Quiz 4

×

Karan Vora (username: kv2154)

Attempt 1

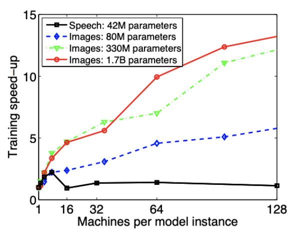
Written: May 4, 2023 5:20 PM - May 4, 2023 5:48 PM

Submission View

Released: May 3, 2023 2:20 PM

Question 1 **0.5 / 2 points**

Observe the figure below from downpour SGD paper about scaling with Model Parallelism. In this figure, the number of machines per model instance is increased and the speed-up in time take per iteration is shown. Based on the figure select all the correct observations.



➡ ✗ ☐ Slow machines can also impact the performance with model parallelism as it may result in many machines waiting for the single slowest machine to finish a given phase of computation.

➡ ✓ ☐ Image models due to their local-connectivity are able to benefit from using many more machines per model replica.

✗ ☐ A model with more parameters will always have better speed-up with increasing machines compared to a model with lesser number of parameters.

✗ ☐ The fully-connected speech model with 42M parameters did not benefit with increasing number of machines per instance beyond 8 machines as communication overhead is dominant and the amount of compute per machine remains the same.

Question 2 **0.8 / 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,

1. When K=P, K-batch sync is same as Sync _______
2. When K=P, K-sync is same as K-batch sync_______
3. When K=P, K-batch async is same as Async_______
4. When K=1, K-batch async and K-async both behave same as Async_______
5. When K>1, stale gradients problem can happen both in K-batch sync and K-batch async_______

Answer for blank # 1: True ✗ **(False)**

Answer for blank # 2: False ✓(20 %)

Answer for blank # 3: True ✗ **(False)**

Answer for blank # 4: True ✓(20 %)

Answer for blank # 5: True ✗ **(False)**

Question 3 **0 / 1 point**

Suppose you are maintaining a deep learning cluster where users come and submit training jobs. All users submit the same type of training job. In your cluster each node has 16 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 16 GPUs. When a job is running on a node and uses less than 16 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 follows:

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

You need to come up with a cost structure (lets call it "best cost") so that the users always prefer to use all the 16 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 GPUs.

When a user provisions 1 GPU for training, the cost you charge is $1.2 per hour. Under the "best cost" cost structure what you should charge (per GPU) when user provisions:

(**Remember**: You answer should be rounded to two places of decimal, e.g., 0.239 is 0.24, 0.233 is 0.23, and 0.2 should be entered as 0.20)

1. 2 GPUs: $ _______ per hour (per GPU cost)

2. 4 GPUs: $ _______ per hour (per GPU cost)

3. 8 GPUs: $ _______ per hour (per GPU cost)

4. 16 GPUs: $ _______ per hour (per GPU cost)

Answer for blank # 1: 1.70 ✗ **(0.85)**

Answer for blank # 2: 2.40 ✗ **(0.60)**

Answer for blank # 3: 3.39 ✗ **(0.42)**

Answer for blank # 4: 4.80 ✗ **(0.30)**

Question 4 **1 / 1 point**

High-performance computing with GPUs is often called heterogeneous computing because:

- ◯ GPUs typically run a variety of kernels over the course of a computation
- ✓ ◯ Computing with GPUs typically also involves CPUs, so there are two different kinds of hardware with different strengths
- ◯ There are very many models of GPUs to choose from when constructing a system

Question 5 **1 / 1 point**

Reproducibility is always guaranteed as long as cudNN routines are executed on the same GPU and cudNN version.

- ◯ True
- ✓ ◯ False

Question 6 **0.333 / 1 point**

Arrange the following distributed training synchronous SGD algorithms with P learners in terms of their **increasing training time** for the same number of epochs. Here 1<K<P.
Enter 1, 2, 3 in the blanks.

1. _______ Fully-sync
2. _______ K-batch sync
3. _______ K-sync

Answer for blank # 1: 3 ✓(33.33 %)

Answer for blank # 2: 2 ✗ (1)

Answer for blank # 3: 1 ✗ (2)

Question 7 **1 / 1 point**

Which of the following correctly describes a GPU kernel

- ◯ A kernel may contain a mix of host and GPU code
- ◯ A kernel is part of the GPU's internal micro-operating system, allowing it to act as in independent host
- ✓ ◯ All thread blocks involved in the same computation use the same kernel

Question 8 **1.5 / 2 points**

Select all that is True about Tensor cores in NVIDIA GPUs.

- ⇨ ✓ ☐ Tensor cores are much faster than cuda cores as they work with reduced precision multiplication
- ⇨ ✓ ☐ The basic role of a tensor core is to perform the following operation on 4x4 matrices: D = A×B + C
- ⇨ ✗ ☐ Tensor cores are much faster than cuda cores as they work with reduced precision multiplication
- ✓ ☐ NVIDIA Tesla V100 has more tensor cores than cuda cores

Question 9 **1 / 1 point**

Which of the following correctly describes the relationship between Warps, thread blocks, and CUDA cores?

- ◯ A warp is divided into a number of thread blocks, and each thread block executes on a single CUDA core
- ✓ ◯ A thread block may be divided into a number of warps, and each warp may execute on a single CUDA core
- ◯ A thread block is assigned to a warp, and each thread in the warp is executed on a separate CUDA core

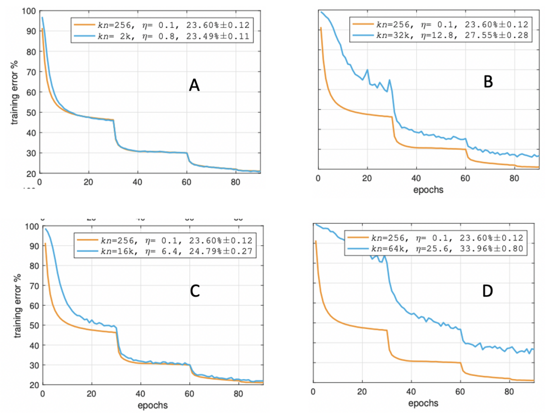
Question 10 **1 / 1 point**

The threads in a thread block are distributed across SM units so that each thread is executed by one SM unit.

- ◯ True
- ✓ ◯ False

Question 11 **1 / 1 point**

Shown below are the experimental results for distributed training with very large minibatch sizes on a cluster with each server having 8 NVIDIA Tesla P100 GPUs. Also shown in each chart is the training error curve for the 256 minibatch baseline. The minibatch size is give by kn, where k is the minibatch size per GPU and n is the total number of GPUs used in the distributed training. For n=8 the minibatch size is 256. The learning rate η is also linearly scaled with minibatch size. Validation error (mean±std of 5 runs) is shown in the legend, along with minibatch size kn and reference learning rate η.



Identify the right number of servers used in distributed training for each of these charts. Your choices are:

8, 64, 128, 256

1. Chart A . _______

2. Chart B. _______

3. Chart C. _______

4. Chart D. _______

Answer for blank # 1: 8	✓(25 %)
Answer for blank # 2: 128	✓(25 %)
Answer for blank # 3: 64	✓(25 %)
Answer for blank # 4: 256	✓(25 %)

Question 12 **0.8 / 2 points**

Select all that is true when systematically benchmarking distributed deep learning systems.

- ➡ ✗ ☐ Given the same network connecting the machines, training using high performance GPUs will have poor scaling efficiency than compared to training using low performance GPUs.
- ➡ ✓ ☐ Training using a neural network with a high compute to communication ratio will give a higher scaling efficiency compared to training using a neural network with lower compute to communication ratio.
- ✗ ☐ Given training throughput with N machines you can calculate the scaling efficiency.
- ✓ ☐ Scaling efficiency is the single most important metric as it gives an indication of training speed-up achievable with using increasing number of machines.
- ➡ ✗ ☐ To show high scaling efficiency one trick is to work with a deep learning framework with higher compute time per iteration.

Question 13 **0.5 / 1 point**

Observe the figure below from Downpour SGD paper about scaling with Downpour SGD using both model and data parallelism. The left figure is time to reach 16% accuracy with different strategies as a function of number of machines whereas right side shows this as a function of number of cores. Based on

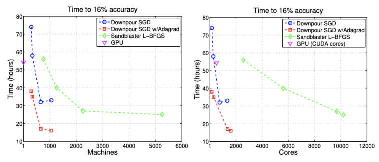


Figure 5: Time to reach a fixed accuracy (16%) for different optimization strategies as a function of number of the machines (left) and cores (right).

the figure select all the correct observations.

- ✗ ☐ The decrease in training time is much steeper with number of machines then with number of cores due to the difference in bandwidth of the communication interface between the cores and the difference in network bandwidth.
- ✓ ☐ This figures show that distributed training with GPUs is slower than distributed training using multiple machines as GPU memory is limited.
- ➡ ✓ ☐ The best tradeoff between cost and performance is with Downpour SGD with Adagrad.
- ✗ ☐ Looking at the graphs the reduction in time is more with number of machines than with number of cores.

Question 14 **0.75 / 1 point**

Consider a distributed deep learning set-up where the learners alternate between parameter server (with a single reducer) and ring topology (using allreduce collective) during the gradient exchange phase. Thus if in one communication round, the gradients are exchanged using parameter server in the next communication round, they are exchanged using ring allreduce. For this distributed training select the right answers from the below for the communication cost. Let N be the total number of model parameters and P be the number of learners.

- ➡ ✓ ☐ The communication cost is linear in the number of learners even though ring allreduce was used in alternate communication round.
- ➡ ✗ ☐ The average communication cost is proportional to N(P-1)(1+1/P)
- ➡ ✓ ☐ Instead of alternating between parameter server and ring allreduce one could just do only ring allreduce and achieve better scalability with number of learners.
- ✓ ☐ The communication cost get practically independent of number of learners for very large P.

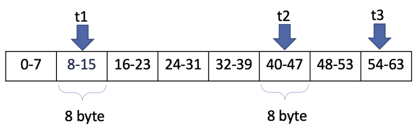
Question 15 **1 / 1 point**

Functions annotated with the `__global__` qualifier may be executed on the host or the device

- ◯ True
- ✓ ◯ False

Question 16 **0 / 1 point**

Three threads (t1, t2, t3) do non- coalesced accesses for a total of 24 bytes as follows:



If the memory transaction is at 64 bytes granularity, what is the effective bandwidth for 24 bytes:

- ◯ 1/2
- ➡ ◯ 3/8
- ◯ 3/4
- ✗ ◯ 1/4

Question 17 **1 / 1 point**

The style of parallelism supported on GPUs is best described as:

- [✓ SIMT - Single Instruction Multiple Thread]
- Data parallelism
- SISD - Single Instruction Single Data
- MISD - Multiple Instruction Single Data

Question 18 **1 / 1 point**

To create an array in shared memory the following code is written:

```
extern __shared__ int s[];
int *integerData = s;                    // nI ints
float *floatData = (float*)&integerData[nI]; // nF floats
char *charData = (char*)&floatData[nF];      // nC chars
myKernel<<<gridSize, blockSize>>>(...);
```

What is the size of array s[] created in shared memory?

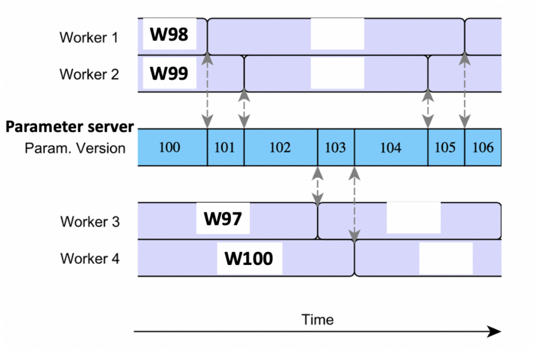
- [✓ Code is incorrect as the shared memory size is not specified when calling mykernel]
- Same as blockSize
- Same as gridSize
- Code is correct and array s size can grow dynamically in the shared memory. No need to specify the size of array s[] when calling the kernel.
- Same as gridSize*blockSize

Question 19 **1 / 1 point**

Which of the following descriptions best captures the characteristics of GPU memory compared to CPU memory?

- [✓ Less cache, longer latency, higher memory bandwidth]
- More cache, longer latency, higher memory bandwidth
- More cache, shorter latency, higher memory bandwidth

Question 20 **0.8 / 2 points**



The picture above shows asynchronous SGD with parameter server. There are four workers and one parameter server. Initially the model version at Worker 1 is 98, Worker 2 is 99, Worker 3 is 97, and Worker 4 is 100. The model version (labeled as parameter version in picture below) at parameter server initially is 100. Fill in the appropriate model version in the weight update equations at the parameter server below. In these equation alpha is the learning rate and grad(w) is the gradient of loss evaluated at w. For each update you also need to calculate and enter the staleness defined as the difference between model version at parameter server and the model version at which the gradient was calculated for that update. The first one is done for you. Note that the staleness in the first update is 100 - 98 =2 (as W100 was the model version at parameter server and W98 was the model version at which gradient was calculated for that update).

a. W101 = W100 - alpha grad(W98) staleness = 2
b. W102 = W101 - alpha grad(1. _______) staleness = 2. _______
c. W103 = 3. _______ - alpha grad(4. _______) staleness = 5. _______
d. W104 =6. _______- alpha grad(7. _______) staleness = 8. _______
e. W105 = W104 - alpha grad(9. _______) staleness = 10. _______
f. W106 = W105 -alpha grad(11. _______) staleness = 12. _______

For this scenario involving 6 weight updates at the parameter server:

g. The maximum value of staleness is 13. _______
h. The minimum value of staleness is 14. _______
i. The mean of staleness values is 15. _______

Answer for blank # 1:	W99	✓	(6.67 %)
Answer for blank # 2:	1	✗	(2)
Answer for blank # 3:	W102	✓	(6.67 %)
Answer for blank # 4:	W97	✓	(6.67 %)
Answer for blank # 5:	5	✓	(6.67 %)
Answer for blank # 6:	W103	✓	(6.67 %)
Answer for blank # 7:	W100	✓	(6.67 %)
Answer for blank # 8:	4	✗	(8)
Answer for blank # 9:	W98	✗	(W102)
Answer for blank # 10:	7	✗	(2)
Answer for blank # 11:	W99	✗	(W101)

Answer for blank # 12: 7 ✗ (4)

Answer for blank # 13: 7 ✗ (5)

Answer for blank # 14: 1 ✗ (2)

Answer for blank # 15: 4.33 ✗ (3)

Question 21 **0.333 / 1 point**

Consider a distributed deep learning experiment with data parallelism and synchronous SGD across multiple servers. Each server has 8 P100 GPUs and batch size per GPU is kept fixed at 64. The training dataset has 131072 images. Scaling efficiency is defined as ratio of per iteration time when training using one 1 server to per iteration time when training using N servers. Following is the result from this experiment, showing how the per iteration time scales as the number of servers.

Number of servers	Time per iteration (secs)
1	0.3
2	0.32
4	0.33
8	0.35
16	0.36
32	0.37
64	0.39
128	0.41
256	0.43

Based on this data provide answer to the following. **Note that for answers with decimal part you need to round to two places of decimal**. No points will be awarded if the answer is incorrectly rounded.
1. Per epoch time (in secs) with 256 GPUs: _______
2. Throughput (images/sec) with 256 GPUs: _______
3. Per epoch time (in secs) with 1024 GPUs: _______
4. Throughput (images/sec) with 1024 GPUs: _______

5. Till 256 GPUs the scaling efficiency is greater than 85%. True or False. _______
6. The scaling efficiency is less than 75% with 1024 GPUs. True or False. _______

Answer for blank # 1: 3.44 ✗ (2.96)

Answer for blank # 2: 38140 ✗ (44281.08)

Answer for blank # 3: 0.94 ✗ (0.82)

Answer for blank # 4: 139440 ✗ (159843.90)

Answer for blank # 5: False ✓(16.67 %)

Answer for blank # 6: True ✓(16.67 %)

Question 22 **0.333 / 1 point**

Arrange the following distributed training asynchronous SGD algorithms with P learners in terms of their **increasing training time** for the same number of epochs. Here 1<K<P.
Enter 1, 2, 3

1. _______ K-batch async
2. _______ K-async
3. _______ Async

Answer for blank # 1: 3 ✗ (2)

Answer for blank # 2: 2 ✗ (3)

Answer for blank # 3: 1 ✓(33.33 %)

Question 23 **1 / 1 point**

Shared memory in CUDA is accessible to:

- ○ All threads associated with a single kernel
- ○ Both the host and GPU
- ✓ ○ All threads in a single block

Attempt Score: 17.649 / 28 - D

Overall Grade (highest attempt): 17.649 / 28 - D

Done