HW6

Q1:

```
Output from main_q1
----------------
[==========] Running 5 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 5 tests from gtestTrain
[ RUN      ] gtestTrain.small1
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
Time for Sequential Training: 0.0271744 seconds
Time for Parallel Training: 0.00452041 seconds
[       OK ] gtestTrain.small1 (167 ms)
[ RUN      ] gtestTrain.small2
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
Time for Sequential Training: 0.00900169 seconds
Time for Parallel Training: 0.0115804 seconds
[       OK ] gtestTrain.small2 (34 ms)
[ RUN      ] gtestTrain.small3
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
Time for Sequential Training: 0.0709044 seconds
Time for Parallel Training: 0.0911635 seconds
[       OK ] gtestTrain.small3 (176 ms)
[ RUN      ] gtestTrain.medium
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
Time for Sequential Training: 0.0507266 seconds
Time for Parallel Training: 0.0408668 seconds
[       OK ] gtestTrain.medium (134 ms)
[ RUN      ] gtestTrain.large
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
Time for Sequential Training: 0.33907 seconds
Time for Parallel Training: 0.139403 seconds
[       OK ] gtestTrain.large (708 ms)
[----------] 5 tests from gtestTrain (1222 ms total)

[----------] Global test environment tear-down
[==========] 5 tests from 1 test suite ran. (1222 ms total)
[  PASSED  ] 5 tests.
```

Q2:

```
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from gtestTrain
[ RUN      ] gtestTrain.custom
Rank 0/1 started on node hpcc-gpu-5-3 [myGPU = 0, nGPUs = 4]
batch_size=3200; num_procs=1; hidden_size=512;
Time for Sequential Training: 0.933764 seconds
Mean time for Parallel Training: (rank 0; repeats = 10): 0.0877465 seconds
Std. deviation of time for Parallel Training (repeats = 10): 0.00722945 seconds
[       OK ] gtestTrain.custom (3573 ms)
[----------] 1 test from gtestTrain (3573 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (3573 ms total)
[  PASSED  ] 1 test.
```

Q2.1

Forward Pass FLOPs

Forward Pass FLOPs

1. First Layer (Input to Hidden Layer):
    - Matrix multiplication: W1 x X with dimensions H x D and D x B
    - FLOPs for matrix multiplication: 2 x H x D x B
    - FLOPs for adding bias b1: H x B
2. Activation (Sigmoid):
    - For each element: 1 exponentiation, 1 addition, 1 reciprocal
    - FLOPs for sigmoid on a1: 3 x H x B
3. Second Layer (Hidden to Output Layer):
    - Matrix multiplication: W2 x a1 with dimensions C x H and H x B
    - FLOPs for matrix multiplication: 2 x C x H x B
    - FLOPs for adding bias b2: C x B
4. Activation (Softmax):
    - For each element: 1 exponentiation, 1 sum for normalization (per column), and 1 division
    - FLOPs for softmax on z2: 3 x C x B

Combining these, the total FLOPs for the forward pass is:
Total FLOPs for forward pass = 2HDB + HB + 3HB + 2CHB + CB + 3CB

= 2HDB + 4HB + 2CHB + 4CB

Backward Pass FLOPs

1. Loss Gradient (Cross-Entropy Loss):
    - FLOPs for loss gradient computation: 3 x C x B
2. Second Layer (Output to Hidden Layer):
    - Matrix multiplication delta2 x a1^T with dimensions C x B and B x H
    - FLOPs for matrix multiplication: 2 x C x B x H
    - FLOPs for updating dW2: 2 x C x H
    - FLOPs for updating db2: 2 x C x B
3. Activation Gradient (Sigmoid Backprop):
    - For each element: 1 multiplication, 1 subtraction, 1 multiplication (from sigma'(x) = sigma(x)(1 - sigma(x)))
    - FLOPs for sigmoid backprop on a1: 3 x H x B
4. First Layer (Hidden to Input):
    - Matrix multiplication delta1 x X^T with dimensions H x B and B x D
    - FLOPs for matrix multiplication: 2 x H x B x D
    - FLOPs for updating dW1: 2 x H x D
    - FLOPs for updating db1: 2 x H x B

Combining these, the total FLOPs for the backward pass is:

Total FLOPs for backward pass

= 3CB + 2CHB + 2CH + 2CB + 3HB + 2HBD + 2HD + 2HB

= 5CB + 2CHB + 2CH + 5HB + 2HBD + 2HD

Total FLOPs

Summing the FLOPs for forward pass and backward pass:

Total FLOPs per batch

= forward + backward

= (2HDB + 4HB + 2CHB + 4CB) + (5CB + 2CHB + 2CH + 5HB + 2HBD + 2HD)

= 4HDB + 9HB + 4CHB + 9CB + 2CH + 2HD

- D = 784
- H = 512
- B = 3200
- C = 10

Total FLOPs = 5219405056

Effective Compute Utilization

CUDA_VISIBLE_DEVICES=0 mpirun -n 1 ./main_q2

CUDA_VISIBLE_DEVICES=0,1 mpirun -n 2 ./main_q2

CUDA_VISIBLE_DEVICES=0,1,2,3 mpirun -n 4 ./main_q2

num_procs=1, nGPUs=1: 0.0886158 seconds

utilization = 5219405056 / 0.0886158 / (16.3 x 10^12) = 0.36%

num_procs=2, nGPUs=2: 0.0844259 seconds

utilization = 5219405056 / 0.0844259 / (16.3 x 10^12) / 2= 0.19%

num_procs=4, nGPUs=4: 0.102481 seconds

utilization = 5219405056 / 0.102481 / (16.3 x 10^12) / 4 = 0.08%

It's a bit surprising that the utilization rate is so low. The more GPUs resources we allocate, the lower the utilization rate is. This suggests that we hard significantly under-utilizing the GPU's computational power. And more GPUs can lead to more communication overhead, which also reduces the utilization rate.

Q2.2

Approximate the number of bytes moved per batch

Consider the following data movements:

1. Input data X to GPU
    - Size: D x B x sizeof(float)
2. Labels y to GPU
    - Size: C x B x sizeof(float)
3. Weights W0 and W1
    - Size: (H x D + C x H) x sizeof(float)
4. Gradients:

    MPI_Allreduce will involve moving gradients across all processes. Each process moves its gradients to others:
    - Total bytes moved per process: (H x D + C x H + C x B + H x B) x sizeof(float)