Q3

We use the kernel function matmul to do the gpu version multi threads matmul. In the kernel function, we specify N * N threads, each for the calculation of 1 item in matrix c. So in the host codes, we specify dim3 variable grid_size and block_size to implement gpu multi threads processing. Before kernel execution, we copy data from host to device for cpu calculation. After kernel execution, we copy data from device to host for result check. Here is the comparison between cpu openblas matmul and gpu matmul.

```
[zhang.yam@login-00 Question3]$ cat slurm-32511731.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 12.889376 ms
[zhang.yam@login-00 Question3]$ cat slurm-32512705.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 133.346896 ms
starting sparse matrix multiply
A result -1.11183e+07
The total time for matrix multiplication with sparse matrices = 125.870446 ms
The sparsity of the a and b matrices = 0.750977
```

Both the multiplication runs 10 times, but the running time of cpu is nearly 11 times more than gpu version.

Let's do some strong scaling here on the gpu version code with increasing numbers of threads.

Grid_size 128*128 Block_size 2*2

```
[zhang.yam@login-01 Question3]$ cat slurm-32527408.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 14.740091 ms
```

Grid_size 64*64 Block_size 4*4

```
[zhang.yam@login-01 Question3]$ cat slurm-32527417.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 5.469203 ms
```

Grid_size 32*32 Block_size 8*8

```
[zhang.yam@login-01 Question3]$ cat slurm-32527440.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 4.742480 ms
```

Grid_size 16*16 Block_size 16*16

```
[zhang.yam@login-01 Question3]$ cat slurm-32527446.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 4.946898 ms
```

According to the running time, we could tell that as the number of threads increases, the running of gpu matmul becomes faster.

Now we test the running on different types of gpu and different numbers of gpus on each node.

v100-pcie: 2 GPU each node

```
[zhang.yam@login-00 Question3]$ cat slurm-32512821.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 9.417927 ms
```

p100: 4 GPU each node

```
[zhang.yam@login-00 Question3]$ cat slurm-32512827.out
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 13.927795 ms
```

k40m: 1 GPU each node 32513866

```
[zhang.yam@login-00 Question3]$ cat slurm-32512794.out
starting dense matrix multiply
a result 0
The total time for matrix multiplication with dense matrices = 1.056890 ms
```

k80: 8 GPU each node

```
[zhang.yam@login-00 Question3]$ cat slurm-32512843.out
starting dense matrix multiply
a result 0
The total time for matrix multiplication with dense matrices = 1.325790 ms
```

NOTE: The result of using k40m and k80 gpu is obviously wrong, which means that there might be some issue on the nodes that are equipped with k40m and k80 gpu.

The running time of v100-pcie is 30% faster than p100. From searching on Google, we know that Tesla V100 is the fastest NVIDIA GPU available on the market. V100 is 3x faster than P100. If you primarily require a large amount of memory for machine learning, you can use either Tesla P100 or V100. So it makes sense that v100 runs faster than p100.

REFERENCE:

1. https://www.e2enetworks.com/blog/tesla-v100-vs-p100-do-you-know-the-differences