Assignment Part B

Implementation:

I have implemented the checkered matrix multiplication by dividing the first matrix into two matrices, one containing only even rows and one containing only odd rows. For the second matrix I have divided into two parts as well, one containing the required elements to multiply with the even matrix and another one with the required elements to multiply with the odd matrix.

In this way, we can have our whole checkered matrix multiplication problem divided into two subproblem of matrix multiplication. After obtaining the result from each matrix multiplication, we can append our output to the output array given even matrix output stays at even row and odd matrix output at odd row.

Now, in order to implement the matrix multiplication, I have implemented using cuda with cuda 9.1 and gcc-6.5 and simulated it with gpgpusim 4.0. With BLOCK_SIZE 32 the gpu threads are used to calculate the block of data to get the matrix multiplication.

For the calculated matrices, I have made the second matrix into column major to access the elements with more locality of references.

Performance Monitor:

For the First matrix multiplication, the several information are given below:

```
gpu_sim_cycle = 46253
gpu_sim_insn = 3178496
gpu_ipc =     68.7198
L2_BW  =      3.9120 GB/Sec
L2_BW_total  =      3.9120 GB/Sec
L1D_total_cache_accesses = 82432
L1D_total_cache_misses = 68096
L1D_total_cache_miss_rate = 0.8261
averagemflatency = 198                (Average memory fetch latency)
Time elapsed on matrix multiplication of 64x128 . 128x64 on GPU: 82000.000000 ms.
```

For the Second matrix multiplication, the several information are given below:

```
gpu_sim_cycle = 45303
gpu_sim_insn = 3178496
gpu_ipc =     70.1608
L2_BW  =      3.9940 GB/Sec
L2_BW_total  =      3.9526 GB/Sec
L1D_total_cache_accesses = 164864
L1D_total_cache_misses = 136192
L1D_total_cache_miss_rate = 0.8261
```

averagemflatency = 198 (Average memory fetch latency)

Time elapsed on matrix multiplication of 64x128 . 128x64 on GPU: 80000.000000 ms.

Analysis:

As the calculated matrices have a structure where the first matrix is accessed in row major order and for multiplication the second matrix in column major order, the access of L1D cache is quite high. The miss ratio being 0.82 which is quite low, we can say that our code is quite optimized in the CPU part.

Now for the runtime, the first matrix multiplication takes about 82s and second taking 80s in the gpgpu. With the gpu cycles being 46253, 45303, we can say that it is being calculated in a lot less time.

Now the average memory latency denoting the data transfer from L2 to gpu memory being 198, it is showing quite fast. Also the L2 bandwidth is about 3.9 Gbps.