Home Work 3

Transpose Kernel_1 : Transpose kernel: Use a row in private memory and matrix transpose is performed. There is no method that can be faster than numpy operation. This method of storing a row in private memory is definitely faster than the naïve algorithm. Tiling algorithm does not help speed up the process either.

Kernel_2: Passed one row as global id. In order to optimize the matrix multiplication we store one row in private memory. Then take an entire column in (shared)local memory. Add the columns or matrix B and matrix C and store the result in Matrix B. Then we use the private memory row and the columns stored in local memory, multiply them to accomplish the operation A*(B+C). This result Is faster than the naïve method as it uses efficient

Kernel_3: This Is done using tiling algorithm. Inside the kernel we initialize two variables A_tile and B_tile both having tile width 2-D dimensions. We first store the Tiled values in this variable from the global memory. We use barrier synchronization to make sure that this storing into local memory is completed before performing any other operation. Once the values are copied into the local memory, we will iterate through the dimensions using another for loop to store the and store their multiplication into private memory. We use barrier again and make sure that this is performed first before loading the resultant values into the output matrix.

Conclusions:

1. The tiled matrix multiplication is the fastest Matrix multiplication. There is a significant reduction in computation time as compared to python.
2. The point of inflection, that is , the point where python becomes a lot slower than that of the Two openCL algorithms used is around Matrix size 85-95.
3. Once the sizes of individual matrices cross 100x100, the only real competition is between the Algorithm 1 and Algorithm 2. Algorithm 2(Tiled matrix multiplication) is faster.
4. If we try for higher and higher sizes, amongst the two algorithms Tiled would be faster. There is also a possibility of getting out of resources error.

The charts and table are given in the excel sheet.
Sheet 1 : This concerns the operation A*(B + C).Tabulating the respective speeds of the algorithm with respect to matrix sizes. The graph is available to view as well.
Sheet 2 : This concerns the Transpose operation. Tabulation and graph is included.

References: 1. http://www.cedricnugteren.nl/tutorial.php?page=7
2. HW2 solutions provided by TA.
3. http://www.ks.uiuc.edu/Research/gpu/files/upcrc_opencl_lec2.pdf