Lokman ABBAS TURKI

13. Batch multiplication of small matrices

The multiplication of large matrices is a classic example in CUDA programming. It is very well presented and explained in the CUDA programming guide. Here, we consider 2 by 2 multiplications of many small matrices.

We have a number $N = 2^{10}$ $(= 1024)$ of $d \times d$ square matrices $(A^n)_{1 \le n \le N}$ and $(B^n)_{1 \le n \le N}$ where ^{n} denotes an index and not a power. We want to define the kernel

```
multiBatch_k(float *A, float *B, int d)
```

of a batch matrix multiplication $(A^n \times B^n)_{1 \le n \le N}$ efficient for each $d = 1, 2, ..., 1024$.

We recall that for two matrices a and b with respective components $(a_{i,j})_{1 \le i,j \le d}$ and $(b_{i,j})_{1 \le i,j \le d}$ where i is the row index and j is the column index, the product $c = a \times b$ gives a matrix of components $(c_{i,j})_{1 \le i,j \le d}$ where each $c_{i,j}$ results from the scalar product of row vector $a_{i,1 \le k \le d}$ with column vector $b_{1 \le k \le d,j}$ i.e. $c_{i,j} = \sum_{k=1}^{d} a_{i,k} b_{k,j}$.

When calling the kernel `multiBatch_k`, we assume that the input pointers `A` and `B` contain respectively the set of matrices $(A^n)_{1 \le n \le N}$ and $(B^n)_{1 \le n \le N}$. We thus assume that the overall memory of the GPU is sufficient. The values are arranged line after line which gives, for example, $A[0] = A^1_{1,1}$, $A[1] = A^1_{1,2}$, ..., $A[d-1] = A^1_{1,d}$, $A[d] = A^1_{2,1}$, ..., $A[d^2 - 1] = A^1_{d,d}$, $A[d^2] = A^2_{1,1}$, ..., $A[N(d^2 - 1)] = A^N_{d,d}$. We will also assume that the result of the multiplication is saved in the memory space pointed to by `A` (overwrites the value of the input `A`).

We distinguish three situations : small d $(1 \le d \le 3)$, medium d $(4 \le d \le 32)$ and large d $(33 \le d \le 1024)$.

1. For small d $(1 \le d \le 3)$, we associate a thread with each matrix multiplication.
  a. Define the kernel `multiBatch_k` which allows this multiplication to be done.
  b. Explain why this solution is not suitable for $(4 \le d \le 32)$.

2. For average d $(4 \le d \le 32)$, we associate d^2 threads with each matrix multiplication.
  a. Define the kernel `multiBatch_k` which allows this multiplication to be done.
  b. Explain why this solution is not suitable for $(33 \le d \le 1024)$.

3. For large d $(33 \le d \le 1024)$, we associate d threads with each matrix multiplication. Define the kernel `multiBatch_k` which allows this multiplication to be done.