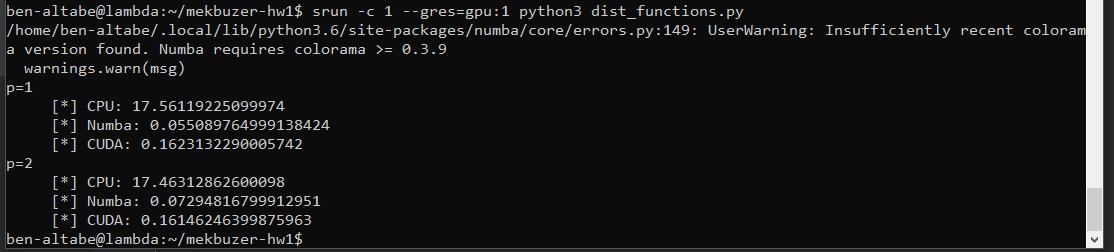
1)

We initilized the kernel with 1000 blocks and 1000 threads per block, where each block is responsible for a certain column, and each thread is responsible for a certain index in that column. For each block we initilized a shared array with one element. All the threads in that block are adding the which is the term in the index that the thread is responsible of. In each block, after every thread added their needed term, there is one thread that is responsible to add the sum of all the terms in his column to C - the output variable that should contain the sum of all the collumns.

And as requested we are doing the final p-root(How to write it lol) in the function dist\_gpu which is executed on the cpu.

You can see here the screenshot from our execution on the server:

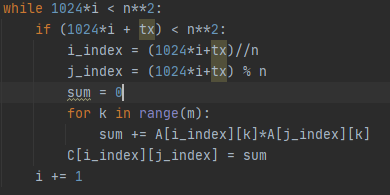


2)

We initialized the kernel with 1 block that contains 1024 threads. We are calculating all of C’s elements where:

* each element is calculated by one thread
* each thread can calculate a number of C’s elements.

The way we are calculating which C’s elements each thread is calculating is by this loop:



One could think of it as a reversable function (I will not prove it although it is quite easy to be convinced)

that is defined by:

and the tx thread is doing the

Since are going through all the numbers(in different threads of course) in we are calculating all of C’s elements.

Note: Each element of C is calculated simply by definition(summing all of the required multiplications)

3)



The gpu matmul is better than Numba because CUDA has 1024 threads that are executing concurrently and Numba will probably has less(depending on the number of available cpu cores on the server).

We didn’t get a significant performance boost because although the extraordinary amount of threads the GPU uses, each thread in the GPU is extremely slow compared to the CPU; note that each thread is calculating [each thread is responsible for roughly C’s elements] operations(adds and multiplications) which is not a very light computation to say the least. Moreover, the data transfer between the CPU and GPU is time consuming as well.

We could improve the performance by adding more threads and use nifty tricks like binary summation(is this the word??)

### delete later…

The Numba is faster than CUDA because although CUDA has 1024 threads that are executing concurrently and Numba will probably has less(depending on the number of available cores on the server), each thread in the GPU is very slow compared to the cpu; note that each thread is calculating [each thread is responsible for roughly C’s elements]operations(adds and multiplications) which is not a very light computation to say the least. Moreover, the data transfer between the CPU and GPU is time consuming as well.