node of 《传统CUDA GEMM 不完全指北》from 知乎

zxp

August 3, 2024

1 background

This article is about the optimization of matrix multiplication without using Tensor Core. The speed achieved by this article is close to cublas. The theme and content of this article are similar to that of a blog by a researcher from the University of California, but slightly less, and the effect is also similar. What is relatively novel is the calculation of occupancy. By adding the –ptxas-options=-v parameter at compile the code and placing the obtained results into the official provided table, so that can get the number of threads that can be switched. (I have also see this method before, but as long as implicit matrix multiplication is used, the actual adjustable amount is limited). The insight we gained is to reduce unnecessary register requests, but necessary registers cannot be avoided. Everything else is similar to what I have seen before. A thread block operates with 32 threads running simultaneously, doing similar tasks (using implicit matrix multiplication, each thread does approximately the same thing). A notable difference is the hassle of transposition encountered by the author in the article, which we do not face because we only load suitable data into matrices (the researcher from the University of California also does not encounter this, because he transposes during the loading process, and I am not sure why the blogger of this article faces this issue). Additionally, changing the global read order is another consideration. Our tensors cannot easily modify the global order, which pertains to data layout. The rest is quite standard: block the matrix, change the loop order, use double buffering, vectorization, and column-major storage.

2 Feelings

Suitable for those who are just starting to write CUDA, the blog from the University of California is better and more comprehensive. It does not solve the problem we face now.