Lab09 - GPU Programming - Matrix Multiplication Report

Họ tên: Lê Khắc Quang Huy

MSSV: 24020158

1. Naive + Coalescing:

    1. Phương pháp: copy dữ liệu từ CPU sang GPU dạng block 2 chiều => Nhân bình thường theo cách index của ma trận 2 chiều, tất cả dữ liệu được load và tính toán trên Global Memory => chậm nhất, tuy nhiên đã có tối ưu nhẹ bằng cách ở mỗi phép tích chấm, cộng dồn vào 1 biến temp => lưu sau cùng -> giảm được truy cập vào ma trận kết quả C

    2. Performance: 1274.143 GFLOP/s

2. Block Tiling

    1. Phương pháp tối ưu: sử dụng kĩ thuật Block Tiling:

        1. Vẫn load ma trận theo block 2 chiều vào Global Mem như cách làm đầu tiên

        2. Ở cách làm đầu tiên, mỗi lần tính toán phải đi lấy dữ liệu từ Global Mem => chậm, vậy nên ta sẽ khai thác Shared Mem, có tốc độ nhanh hơn

        3. Chia quá trình nhân ma trận thành nhiều module, mỗi module tính toán 1 lượng nhỏ phép tính trực tiếp trên Share Mem. Kích thước mỗi ma trận là lcM, lcN, lcK - đại diện cho M N K ở ma trận gốc, ở đây đang config lcM, lcN, lcK = 32 để fit vào 1 wrap (1 wavefront trong AMD = 64, tuy nhiên khi config = 64 perf không cải thiện, chưa giải thích được), khi đó quá trình lấy bộ nhớ sẽ được map theo wrap => tối ưu hơn

    2. Performance: 2635.926025 GFLOP/s

3.1D Register Tiling

1. Phương pháp: Ở phương pháp 2, mỗi thread chỉ tính kết quả của 1 phần tử kết quả => số phép tính thực hiện trên 1 thread thấp => không hiệu quả trong việc che latency => 1 thread sẽ tính kết quả cho 1 vector kích thước TM phần tử, việc tính toán kết quả cho C sẽ được “Register Cached”

2. Performance: 3055.34 GFLOP/s

4. 2D Register Tiling

1. Phương pháp: ý tưởng giống với 1D, tuy nhiên cải tiến để 1 thread tính được kết quả cho 1 ma trận kích thước TM * TN => tăng Occupancy

2. Performance: 3652.47 GFLOPS

5. Vectorized

1. Phương pháp: cải tiến việc load dữ liệu

2. Performance: 5733.50 GFLOPS

6. Warp Tiled + CUDA Core: Ý tưởng theo em hiểu là sử dụng kĩ thuật Parallel reduction, tuy nhiên em chưa hiểu cách cài đặt