Recursive Strassen algorithm focusing on memory space reuse

2110192 RUIZHI LI

Matrix multiplication is part of basic linear algebra and plays an important role in numerical computations in various scientific and technical fields such as machine learning and image processing. Recently popular technologies such as deep learning and edge computing have especially large amounts of data,requiring high precision,and there are more and more opportunities that require large-scale matrix multiplication. Therefore, the Strassen algorithm, which speeds up large-scale matrix multiplication, is attracting attention.

The Strassen algorithm is an algorithm that can calculate large-sized matrices faster than the standard matrix multiplication algorithm. For a square matrix $C = A \times B$, size $M \times M$, where M is even, C, A, and B are first divided into four submatrices each of size $\frac{M}{2}$, and the result is obtained by performing 7 multiplications and 15 additions/subtractions using the Strassen algorithm improved with Winograd. As M increases, the number of operations required by the Strassen algorithm becomes smaller than that of standard multiplication, making it faster. Rectangular matrices with A's number of rows and B's number of columns can be considered in the same way, and M can also be handled when odd using padding.

However, there is a memory usage problem when applying Strassen's algorithm to GPUs. Previous studies have used temporary matrices to save intermediate results during calculations, which require memory allocation using Malloc. This poses a challenge for GPUs with limited global memory space to apply the advantage of Strassen's algorithm for large-scale matrix multiplication. This study focuses on the NVIDIA GPU Tesla A100 and compares it with V100 and P100 to investigate ways to save memory space.

To save memory space, this study proposes a method that avoids memory allocation using Malloc in recursive Strassen's algorithm. Instead, the unused matrices at higher levels are provided to lower levels, and a space is provided to save intermediate results as support matrices. This allows the entire Temporary matrix allocated by Malloc to be removed. To address the problem of insufficient reusable memory space, a Memory space usage analysis algorithm was used to search available memory space at lower levels, while adjusting the calculation order. Experimental results showed that the proposed method could save up to 18.11% of global memory in Level-4 calculations compared to memory-focused previous studies. This revealed optimization level issues and further acceleration issues.

To address the issue of depth of recursive execution, which affects execution speed the most, this study proposes an equation to judge the performance of

GPU multiplication and addition/subtraction functions using measurements. The results of the experiments showed 2-7% error, but the relationship between the boundary of each level was consistent with theory.

Furthermore, to explore further acceleration possibilities, this study proposes a parallel processing technique using partial synchronization, focusing not only on multiplication but also on parallel addition and subtraction based on previous studies. Experimental results showed that the proposed method achieved up to 1.451 times speedup compared to the standard matrix multiplication in CUBLAS11.7 for size 40,960 on A100.