Proposed a sparse matrix storage format applying non-zero element position information dictionary compression to improve sparse matrix dense vector product on GPU.

2210193 MURAKAMI Shun

In recent years, numerical simulations have become increasingly complex and large-scale. Considering this, there is a need to fast solution of simultaneous linear equations with millions of rows coefficient matrices. And graph analysis, partial differential equations discretised by the finite element method and among others are expressed as simultaneous linear equations with a sparse coefficient matrix whose elements are mostly zero. To solve them, Direct and iterative methods are used. However, the direct method is computationally expensive, and when LU decomposition is performed for a sparse matrix, the memory usage increases due to the generation of fill-ins. Therefore, when solving simultaneous linear equations consisting of large and sparse coefficient matrices, iterative methods that do not involve transformation of the coefficient matrices are used.

Sparse Matrix Vector products (SpMV) is the major computation that consumes the solution time of iterative methods. SpMV is memory intensive with only one multiply-accumulate operation for each matrix element. For this reason, speed-up has been achieved by using GPUs (Graphics Processing Units), which have a higher memory bandwidth than CPUs.

The Compressed Sparse Row (CSR) format is often used to store large sparse matrices in the device memory of GPUs, which has high memory efficiency. However, SpMV in CSR format requires stride and reduction memory access. Therefore, sparse matrix storage formats such as SlicedELL and SELL-C-σ formats have been proposed to improve the memory access pattern and enable fast SpMV computation. These formats reduce the memory access penalty caused by stride accesses and bring out device memory bandwidth, but the number of memory accesses must be reduced to further speed up SpMV.

To reduce the number of memory accesses, there are two methods: compressing the values of the non-zero elements and compressing the position information of non-zero elements. Comparing the two, compressing the values of non-zero elements is a strong matrix-dependent aspect, such as real, binary or complex values. Compression of non-zero element position information is easier than compression of the values, since they are integers and the combination of row and column numbers is unique. For these reasons, this study focuses on compression of non-zero element position information.

To compress non-zero element position information, there are delta encoding and dictionary compression methods. CoAdELL has been proposed as a delta

encoding method, which takes the difference of the column numbers of row vectors and reduces the capacity by reducing the number of bits for position information. However, depending on the size of the distance, the number of bits cannot be reduced, and variable-length bits that are not multiples of two bits are difficult to compute on GPUs. On the other hand, dictionary compression methods can be expected to achieve very high compression rates, depending on the non-zero pattern of the sparse matrix. However, the GPU has groups of threads called a warp, and each thread in the warp executes the same instructions. Therefore, if the word length of the dictionary is different, the instructions executed by the threads in the warp will be different. This causes a reduction in execution efficiency called warp divergence.

Therefore, this paper proposes a compressed sparse matrix storage format that enables fast computation of SpMV on GPUs by applying dictionary compression to non-zero element location information and reducing memory access. The proposed method reduces the memory usage by up to 29.5% and accelerates the SpMV computation time by up to 19.6% compared to the CSR format.

Since the sparse matrix storage formats dedicated to these SpMV calculations do not allow easy addition and deletion of non-zero elements, it is common to generate and store sparse matrices in the COO format, which is easier to edit. Therefore, the storage format must be converted in order to use the CSR format, SELL-C-σ and CoD-SELL. In particular, CoD-SELL requires computation for dictionary compression, which is expected to increase the conversion time. Therefore, we propose a fast format conversion for CoD-SELL, which can be converted in a similar computation time to the conversion from COO format to CSR format.