Project Assignment

Kamilla Faizullina

To run programs, we use NVDIA Tesla V100 Nvlink. Memory bandwidth is 900 GB/s and the prformance is 7.8 teraFLOPS for Double-Precision and 15.7 teraFLOPS for Single-Precision.

1 Formats for Sparse Matrix

A sparse matrix is a matrix that has a lot of zero values. To store such matrix more efficiently than usual two-dimensional matrix, there are special data structures. In this work, we focus only on the Compressed Sparse Row (CSR) format and SELL-C-Sigma approach by Kreutzer (which is extension of ELLPACK structure).

The implementation of naive CSR (scalar CSR) is 3 arrays: pointers on starts of row (+last element - the number of non-zero elements in matrix), indexes on columns and the values. It is seems not bad that we do not store any extra zeros and data are really compressed (see 1). However, the matrix algebra is often used on GPU platform and scalar CSR usually performs not well on GPU. The problems are memory access pattern. It is supposed that a lot of threads would not process enough non-zeroes values. Scalar CSR could be extended to vectorized CSR for which used warp per row. According to Kreutzer et al,it is still could be inefficient (However, the performance also depends on the pattern of original matrices).

The next generalization is ELLPACK which is similar to CSR but includes padding. It is a good approach when the average number of non-zero elements. However, padding is a solve of non coalesced memory accesses of CSR and the other problem at the same time. If one of the rows has too much elements, ELLPACK is inefficient. To overcome this, we can use sliced ELLPACK-S, where for every S rows different number for padding.

The more general approach is SELL-C-SIGMA. It is sliced ELLPACH-C where every SIGMA elements are sorted. When C=1 and SIGMA=1 it is CRS format. However, low C is not efficient values. According to authors 1, this approach is good for GPU as there is no loading of too much extra zeros and each thread run only until the actual row length is reached. However, if number of non-zero element in stored values (β) it is still could be not so efficient (However, there is no perfect data structure for). This approach is quite general due to sorting and padding only every C rows.

Naive ELLPACK is quite easy to implement using Kokkos as there constant number of elements in values.

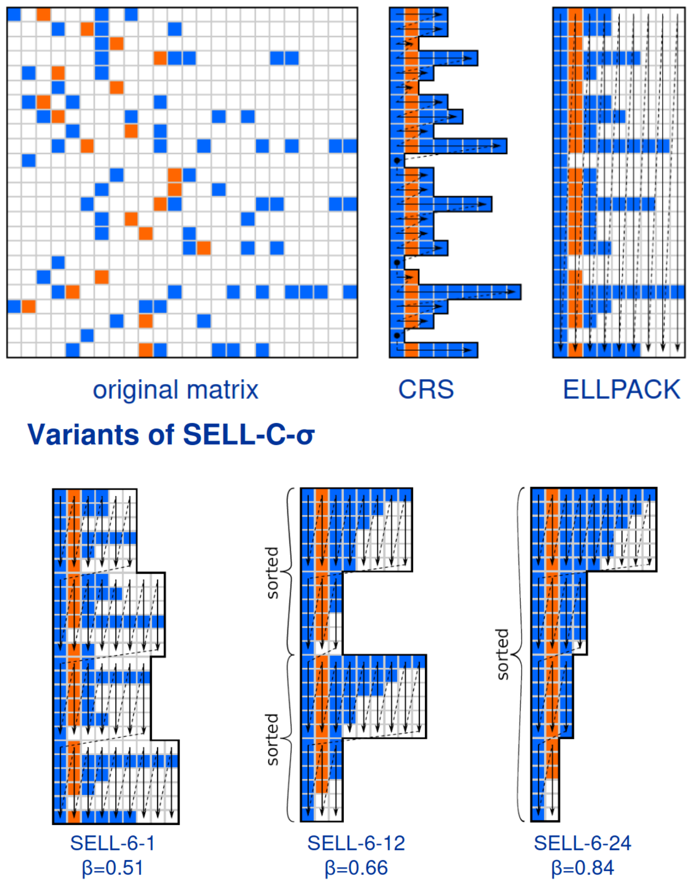


Figure 1: Illustration of matrix formats from 1.

2 Implementation

2.1 CSR

To implement scalar CSR, we use three arrays as described below.

We measure memory bandwidth and computational performance. As expected, the performance values for runnings on GPU are higher than for CPU. Figure 3 illustrates obtained results. Obtained maximum is 452.416 Gb/s which is only half of theoretical value. The times for copy matrix and vectors to devises are higher ≈50 times the computations.

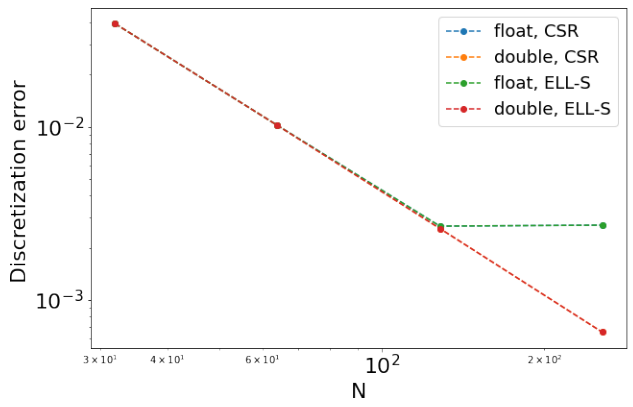
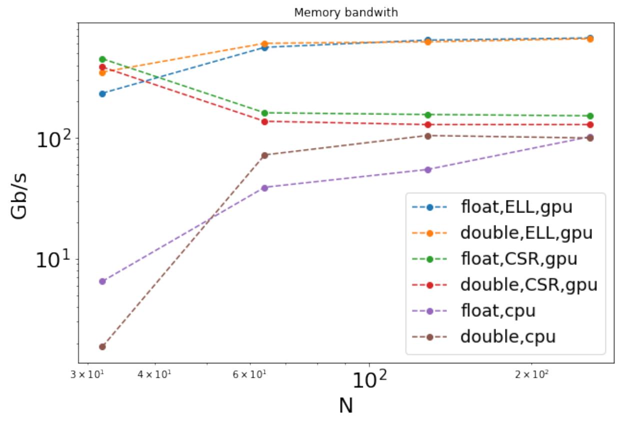


Figure 2: Discretization errors as a function of N. Just for checking the correctness of implementation.

References

[1] Kreutzer, M., Hager, G., Wellein, G., Fehske, H., Bishop, A. R. (2014). A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units. SIAM Journal on Scientific Computing, 36(5), C401-C423.



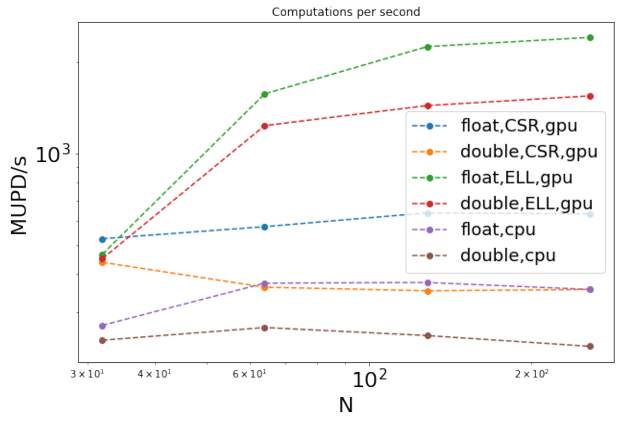


Figure 3: Performance obtained for using CSR format. For CPU, we use 16 processes.