

Scientific Computing and Machine Learning on Multi- and Manycore Architectures

Exercise 4

林澤佑
Tse-Yu Lin

D06946003@ntu.edu.tw

Data Science Degree Program

Analysis

In this exercise, compressed sparse row format is implemented. A special type of matrix is used here.

Let $A = [a_{ij}]_{0 \leq i, j \leq n-1}$ be a n -by- n matrix defined by

$$a_{ij} = \begin{cases} i + j & , \text{ if } i + j \equiv 1(\text{mod } 2) \\ 0 & , \text{ otherwise.} \end{cases}, 0 \leq i, j \leq n - 1 \text{ !0-based!}$$

In this case, each row i contains $\frac{n}{2}$ nonzero elements. So, the result can be easily verified.

For instance, in the case $n = 3$, we have

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix}$$

with CSR format:

$$\begin{array}{lcl} \text{value} & = & [1 \quad 1 \quad 3 \quad 3] \\ \text{colidx} & = & [1 \quad 0 \quad 2 \quad 1] \\ \text{rowptr} & = & [0 \quad 1 \quad 3 \quad 4] \end{array}$$

For $n = 4$, we have

$$\begin{array}{lcl} \text{value} & = & [1 \quad 3 \quad 1 \quad 3 \quad 3 \quad 5] \\ \text{colidx} & = & [1 \quad 3 \quad 0 \quad 2 \quad 1 \quad 3] \\ \text{rowptr} & = & [0 \quad 2 \quad 4 \quad 6] \end{array}$$

For each size of matrix, there are 4 time are collected. They are time of row non-zero element computing, partial sum counting, value and column index collecting, and all procedure including memory between CPG and GPU.

As we can see in figure 1, first two step has linear computational time.

Note that when size around 1700 and 3500, the time of column index collecting step have a jump. The reason of this situation is still unknown.

In this exercise, two setting of block setting are considered, where their total number of threads are the same. For the first one, one block is used, and for the second, 256 blocks is used,

The result of second setting is shown in figure 2. Note that as the size of matrix getting larger, the first two steps become unstable compared with the result using only one block. As size higher than 1,000, system returns fail.

Run the code

For each program in this exercise, please use

```
nvcc ex4.cu -o ex4 -lm
```

to compile, where $x = 1, 3$ indicates the LU decomposition without and with pivoting.

To see computational time, please type

```
./ex4 N
```

where N is the size of matrix. The result would look like:

```
runtime  nz[s]: 0.000038
runtime psum[s]: 0.000015
runtime CSR[s]: 0.000016
runtime ALL[s]: 0.000241
```

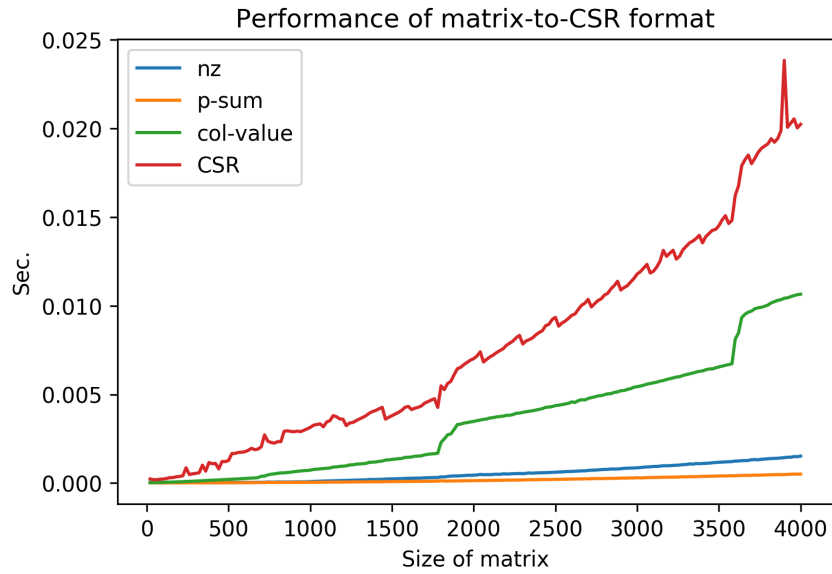


Figure 1: Performance of CSR format under different matrix size. (block = 1)

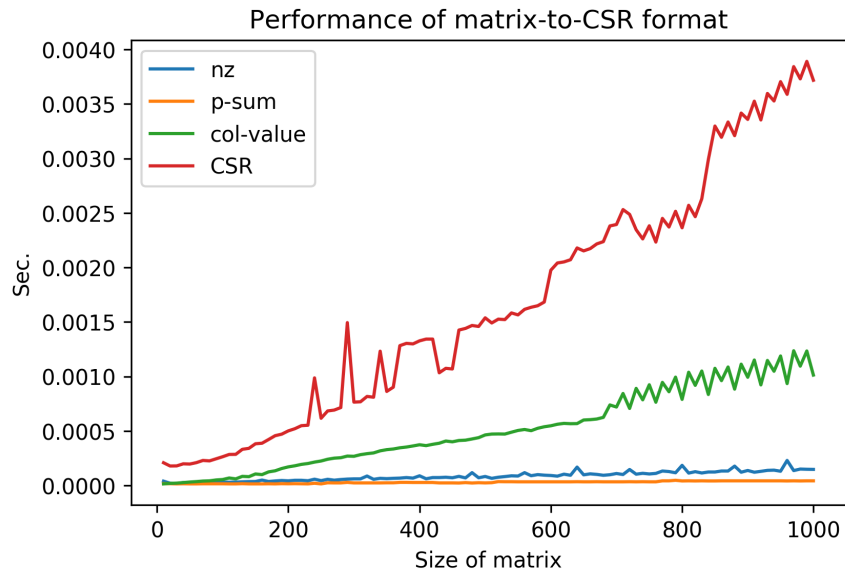


Figure 2: Performance of CSR format under different matrix size. (block = 256)