Homework 10

Qinglei Cao

1. Dense-to-CSR

First, the matrix value was set by that if a random number is dividable by DIV (e.g. 1000), then set its value nonzero (simple set to 2.0). Then, the macro GPU, CPU and CORRECT is to run on GPU, CPU and check the correctness of the result (should be used by combining GPU and CORRECT to compare results of CPU and GPU, which should be the same), respectively.

1.1. Correctness

Use two way to check the correctness: verified by small matrix and compared CPU result with GPU result.

Verified by small matrix:

```
[qcao3@saturn hw11]$ srun -N 1 -w d15 ./dense2csr 5
matrix size: 5
The matrix is:
 0.000000 0.000000 0.000000 0.000000 0.000000
 2.000000 0.000000 2.000000 0.000000 0.000000
 0.000000 0.000000 0.000000 2.000000 2.000000
 2.000000 0.000000 2.000000 2.000000 0.000000
 2.000000 0.000000 0.000000 0.000000 0.000000

rowptr is:
0 0 2 4 7 8
colidx is:
0 2 3 4 0 2 3 0
Result is correct!
```

We can see the result is correct.

Compared CPU result with GPU result:

Define GPU and CORRECT at the beginning, then it will compare results (both rowptr and colidx) of GPU with that of CPU.

1.2. Performance

Environment: d15.descartes of Saturn cluster. DIV control the matrix pattern (dense of sparse).

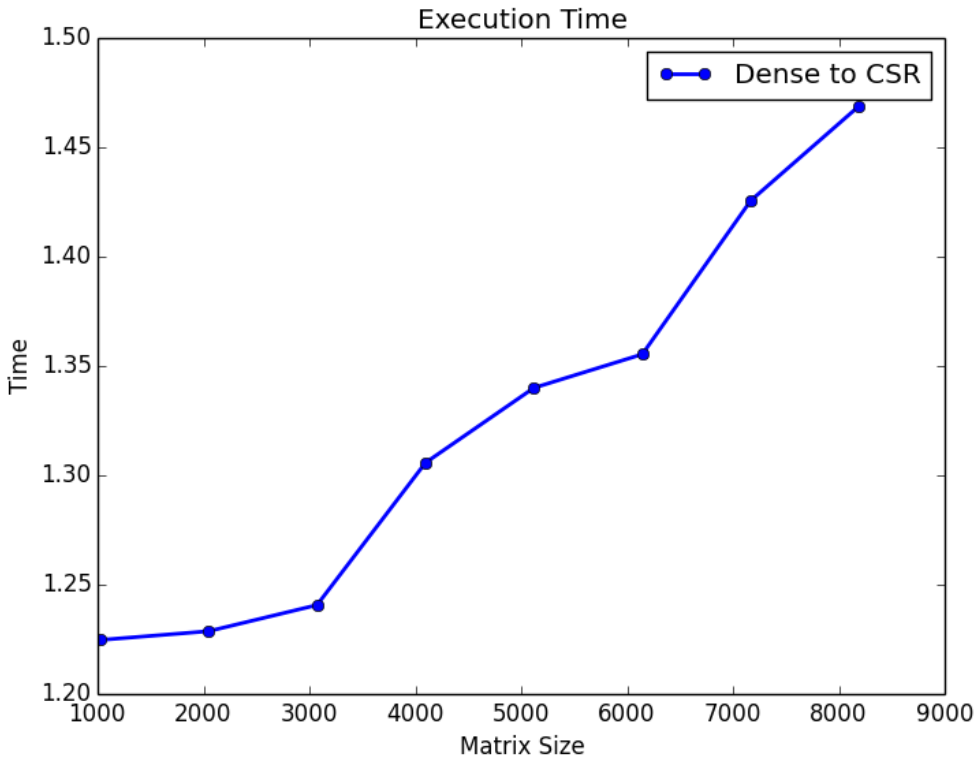

Fig.1 Execution Time

We can see the runtime increases about from 1.23 to 1.47, as matrix size increases from 1024 to 8192. Then set the total operations is Matrix_size * Matrix_size to see the performance trend (Matrix_size * Matrix_size / time), which is shown in Figure 2.

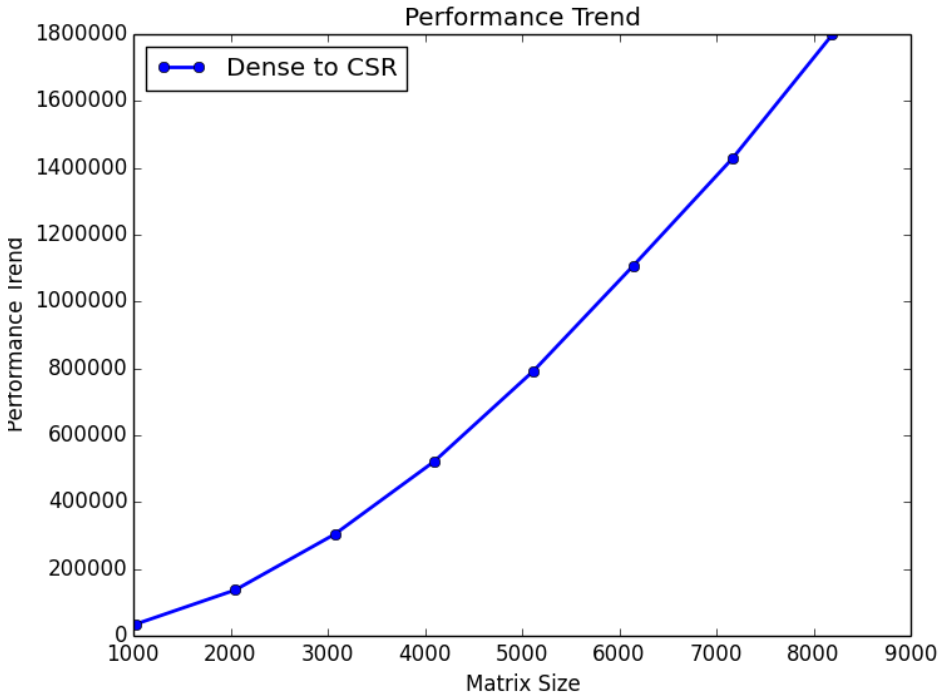

Fig.2 Performance Trend

2. SpMV

First, the matrix value was set by that if a random number is dividable by DIV (e.g. 1000), then set its value nonzero (simple set to 1.0); and simply set values in x to 1.0. **Then**, use the Dense to CSR in PART 1 to get CSR of matrix. **Finally**, the macro GEMV, SPMV and CORRECT is to run GEMV, SPMV and check the correctness of the result (should be used by combining SPMV and CORRECT to compare results of GEMV and SPMV, which should be the same), respectively.

2.1. Correctness

Use two way to check the correctness as before: verified by small matrix and

compared SPMV result with GEMV result.

Verified by small matrix:

```
[qcao3@saturn hw11]$ srun –N 1 –w d15 ./SpMV 5
matrix size: 5 matrix is:
 1.000000 0.000000 0.000000 1.000000 0.000000
 0.000000 0.000000 0.000000 1.000000 0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000 0.000000 1.000000
 0.000000 0.000000 0.000000 1.000000 0.000000

x is:
1.000000 1.000000 1.000000 1.000000 1.000000
result y is:
2.000000 1.000000 0.000000 1.000000 1.000000
row count is:
2 1 0 1 1
```

We can see the result is **correct**, which means row count is equal to y, as just set value to 1.0.

Compared SPMV result with GEMV result:

Define SPMV and GEMV at the beginning, then it will compare result of SPMV to that of GEMV.

2.2. Runtime

Environment: d15.descartes of Saturn cluster. DIV control the matrix pattern (dense of sparse).

Setting:

DIV = 1: matrix is dense, **all the value is nonzero**

DIV = 10000: **matrix is sparse**

TRID: **matrix is tridiagonal**

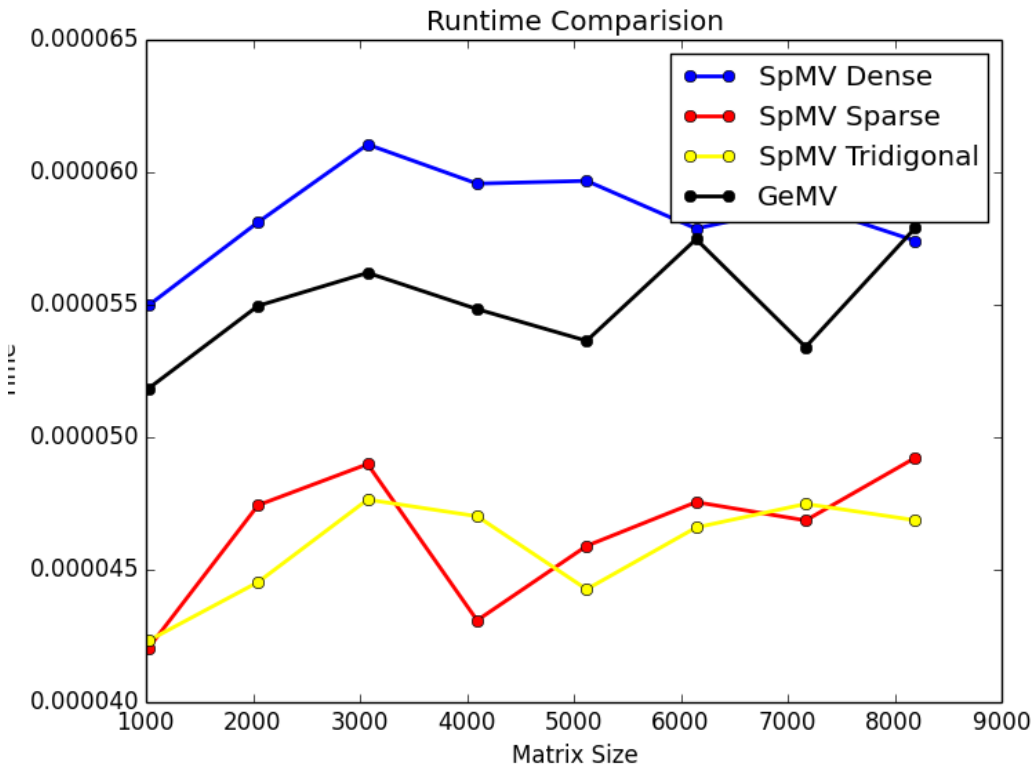

Fig.3 Runtime Time Comparison

From Figure 3 above, we can see, if the matrix is spare, SpMV is faster than GeMV, which makes sense. However, if the matrix is dense (without zero), GeMV is a little faster than SpMV, maybe because there is overhead to set the index and operations are the same.