step0

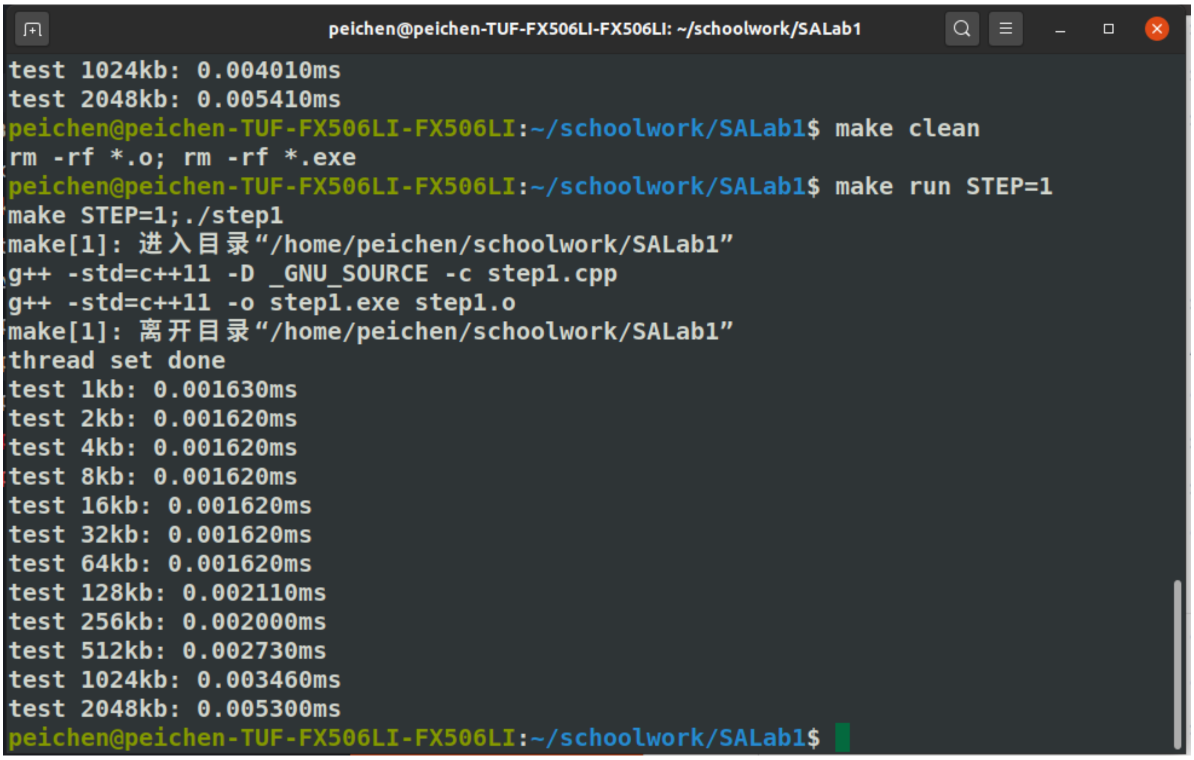
在windows上查找:

cache1:64k

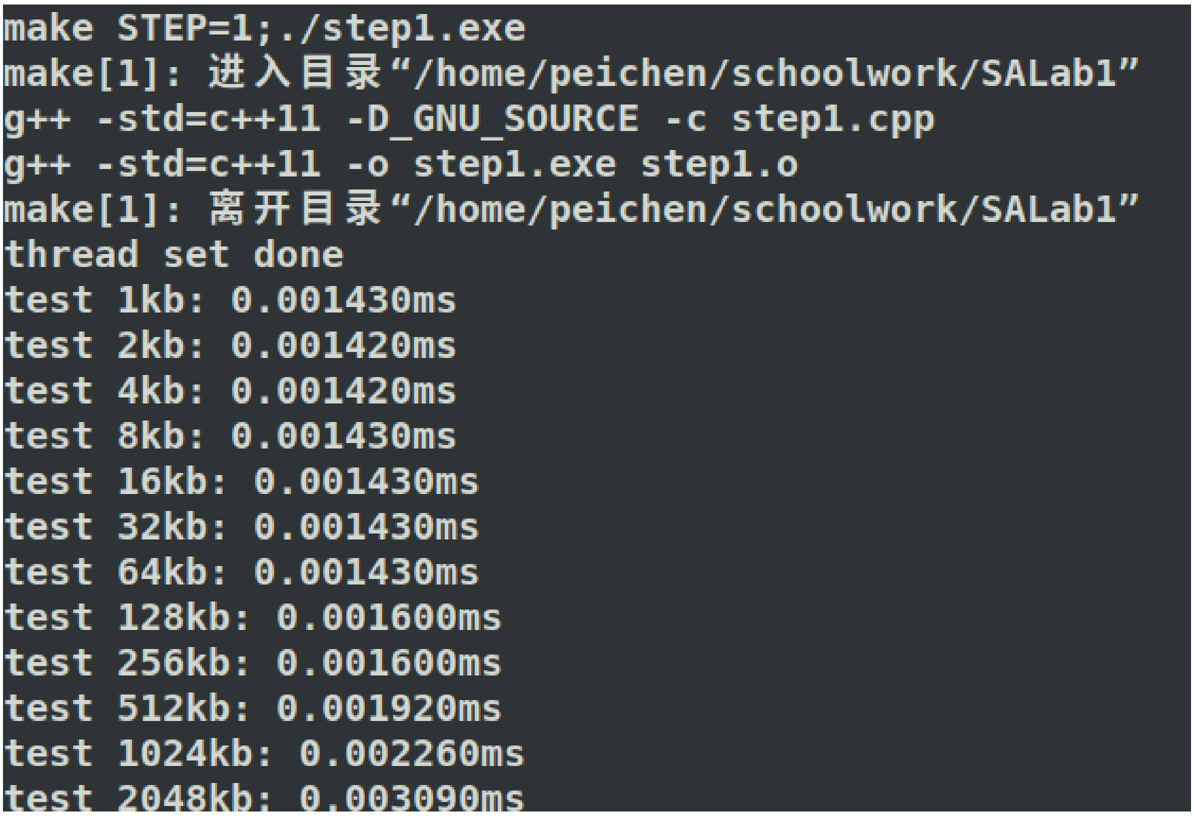
cache2:256k

cache3:1m

step1



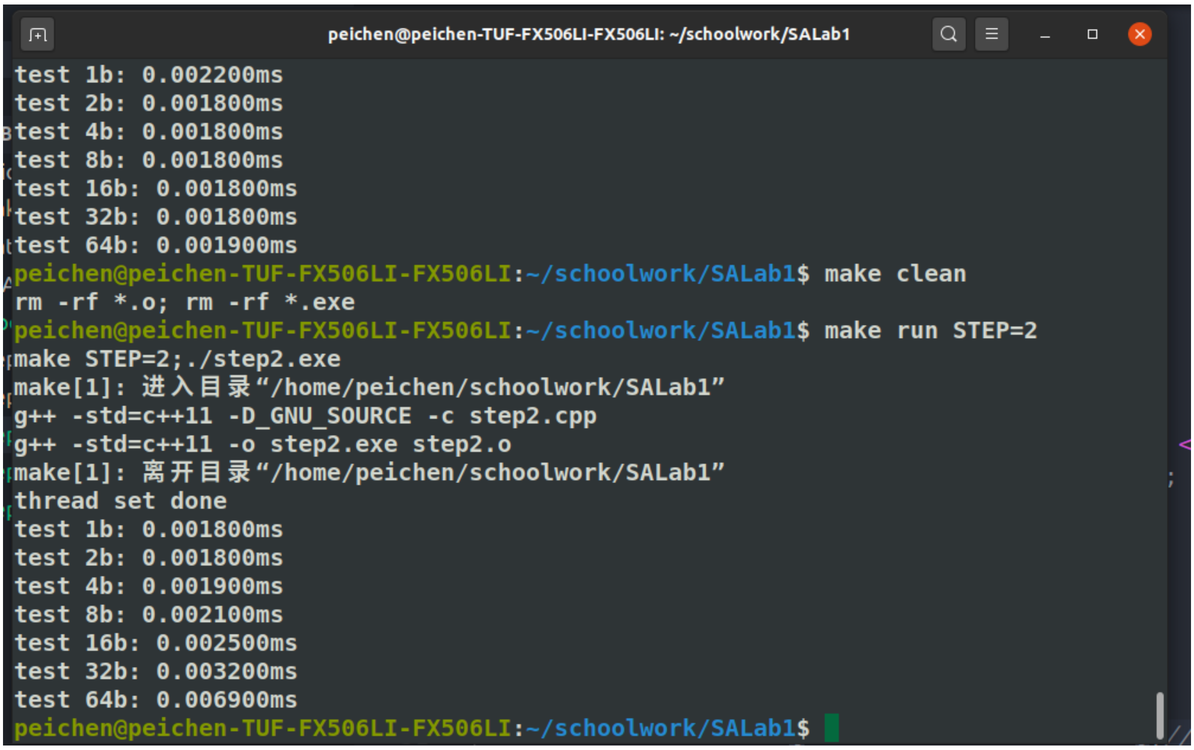
```
test 1024kb: 0.004010ms
test 2048kb: 0.005410ms
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$ make clean
rm -rf *.o; rm -rf *.exe
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$ make run STEP=1
make STEP=1;./step1
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
g++ -std=c++11 -D _GNU_SOURCE -c step1.cpp
g++ -std=c++11 -o step1.exe step1.o
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
thread set done
test 1kb: 0.001630ms
test 2kb: 0.001620ms
test 4kb: 0.001620ms
test 8kb: 0.001620ms
test 16kb: 0.001620ms
test 32kb: 0.001620ms
test 64kb: 0.001620ms
test 128kb: 0.002110ms
test 256kb: 0.002000ms
test 512kb: 0.002730ms
test 1024kb: 0.003460ms
test 2048kb: 0.005300ms
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$
```



```
make STEP=1;./step1.exe
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
g++ -std=c++11 -D_GNU_SOURCE -c step1.cpp
g++ -std=c++11 -o step1.exe step1.o
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
thread set done
test 1kb: 0.001430ms
test 2kb: 0.001420ms
test 4kb: 0.001420ms
test 8kb: 0.001430ms
test 16kb: 0.001430ms
test 32kb: 0.001430ms
test 64kb: 0.001430ms
test 128kb: 0.001600ms
test 256kb: 0.001600ms
test 512kb: 0.001920ms
test 1024kb: 0.002260ms
test 2048kb: 0.003090ms
```

可以看出在64kb和256kb的时候有明显上升。

step2



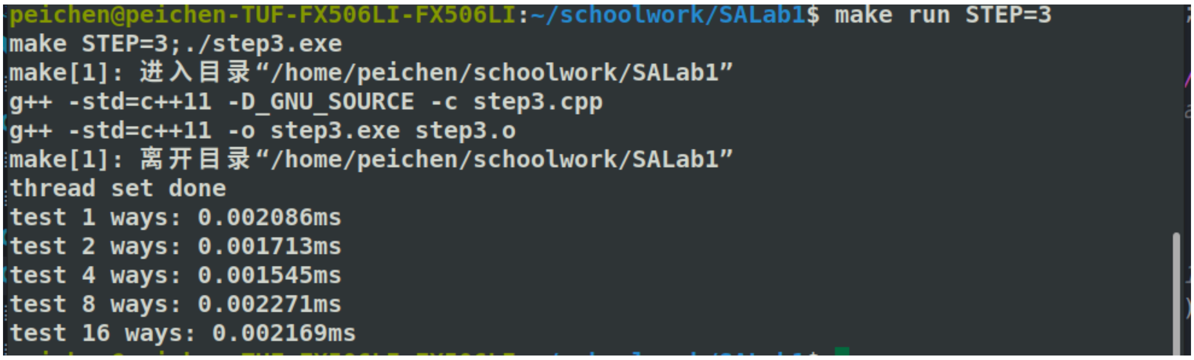
```
test 1b: 0.002200ms
test 2b: 0.001800ms
test 4b: 0.001800ms
test 8b: 0.001800ms
test 16b: 0.001800ms
test 32b: 0.001800ms
test 64b: 0.001900ms
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$ make clean
rm -rf *.o; rm -rf *.exe
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$ make run STEP=2
make STEP=2;./step2.exe
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
g++ -std=c++11 -D_GNU_SOURCE -c step2.cpp
g++ -std=c++11 -o step2.exe step2.o
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
thread set done
test 1b: 0.001800ms
test 2b: 0.001800ms
test 4b: 0.001900ms
test 8b: 0.002100ms
test 16b: 0.002500ms
test 32b: 0.003200ms
test 64b: 0.006900ms
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$
```

可以看到在32b到64b上出现了不正常增长

step3

尝试了三种不同的算法，实验效果都不明显。发现测试后隔一段时间再测会有较好效果，但多次实验n=4时和n=5时的时间差距最明显，因此可能是八路组相连。

算法大同小异，目的是为了以步长大于block size，远小于总大小（不会超出一个组）的方式反复访问一个数组。当访问个数正好是联通度两倍时，每次访问都会是miss，因此时间应该最长。以实验手册的算法为例，因为只访问奇数块，所以最后是块数除4。



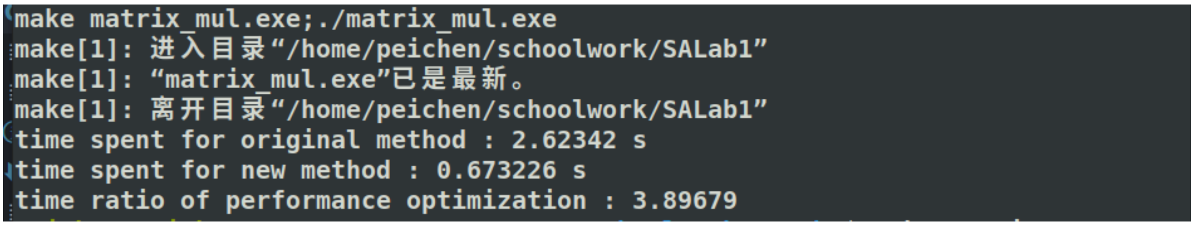
```
peichen@peichen-TUF-FX506LI-FX506LI:~/schoolwork/SALab1$ make run STEP=3
make STEP=3;./step3.exe
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
g++ -std=c++11 -D_GNU_SOURCE -c step3.cpp
g++ -std=c++11 -o step3.exe step3.o
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
thread set done
test 1 ways: 0.002086ms
test 2 ways: 0.001713ms
test 4 ways: 0.001545ms
test 8 ways: 0.002271ms
test 16 ways: 0.002169ms
```

矩阵加速

试了两个办法

重排循环顺序:

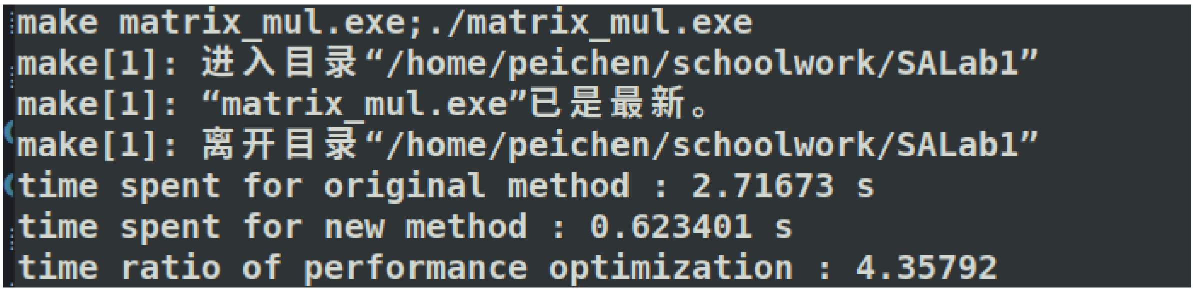
```
for(int register i = 0; i < MAXSIZE; i ++){
        for(int register k = 0; k < MAXSIZE; k ++){
            int register num = a[i][k];
            for(int register j = 0; j < k; j ++){
                d[i][j] += num * b[k][j];
            }
        }
    }
```



```
make matrix_mul.exe;./matrix_mul.exe
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
make[1]: “matrix_mul.exe”已是最新。
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
time spent for original method : 2.62342 s
time spent for new method : 0.673226 s
time ratio of performance optimization : 3.89679
```

分块:

```
for(int register jj = 0; jj < MAXSIZE; jj += blockSize){
        int register jjMaxSize = min(jj + blockSize, MAXSIZE);
        for(int register kk =jj; kk < MAXSIZE; kk += blockSize){
            int register kkMaxSize = min(kk + blockSize, MAXSIZE);
            for(int register i = 0; i < MAXSIZE; i ++){
                for(int register j = jj; j < jjMaxSize; j ++){
                    sum = 0;
                    for (int register k = kkMaxSize; k > kk ; k --){
                        if(k==j){break;}
                        sum += a[i][k] * b[k][j];
                    }
                    //write, do it once
                    d[i][j] += sum;
                }
            }
        }
    }
```



```
make matrix_mul.exe;./matrix_mul.exe
make[1]: 进入目录“/home/peichen/schoolwork/SALab1”
make[1]: “matrix_mul.exe”已是最新。
make[1]: 离开目录“/home/peichen/schoolwork/SALab1”
time spent for original method : 2.71673 s
time spent for new method : 0.623401 s
time ratio of performance optimization : 4.35792
```

虽然最优下分块快，但平均看起来分块比重排要慢，可能是因为矩阵不够大。