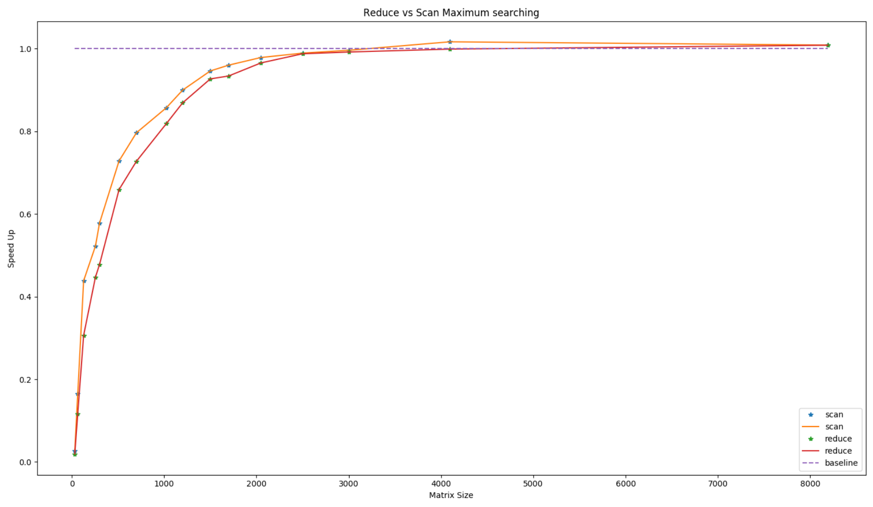
Parallel Computing - Project 2

Zhanhao Chen
Oct 16, 2017

Abstract

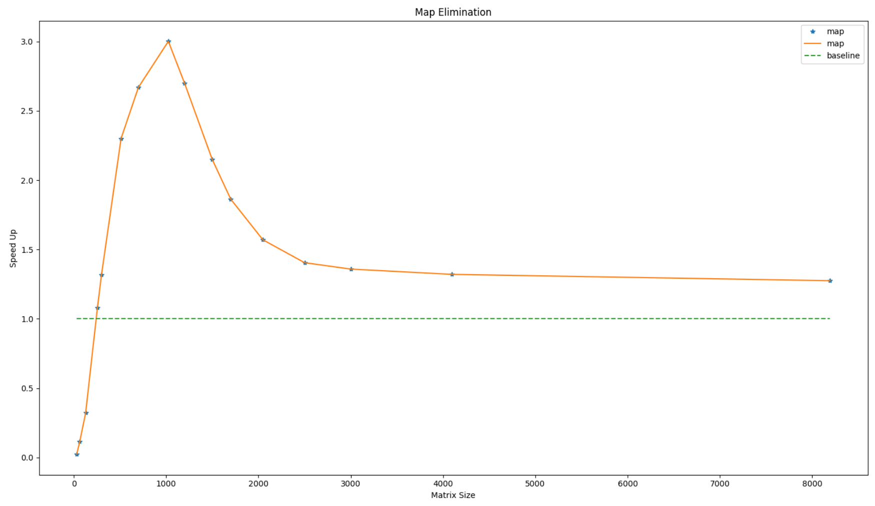
In this project, I have tried to speed up the program by parallelizing three components: Maximum Searching, Elimination and Substitution. The patterns I used is drawn in the flowchart file. Besides, I tried to modify the data structure to optimize the locality. I have designed 10 experiments with tbb to test these strategies. The details of these experiments is listed in the README file. All experiments of this project is done in sunlab machine (8 cores).

Maximum Searching



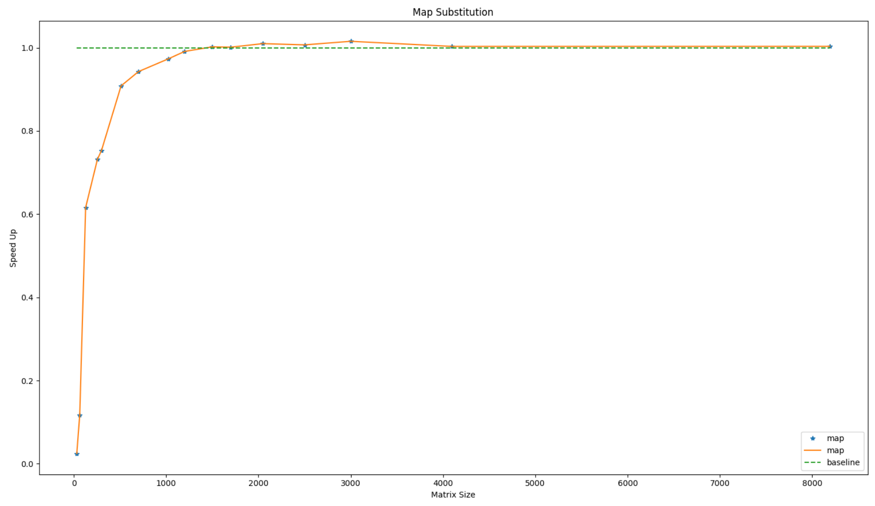
When the dimension of the matrix is smaller than 4000, both pattern failed to defeat baseline. And scan could achieve better performance that reduce.

Elimination



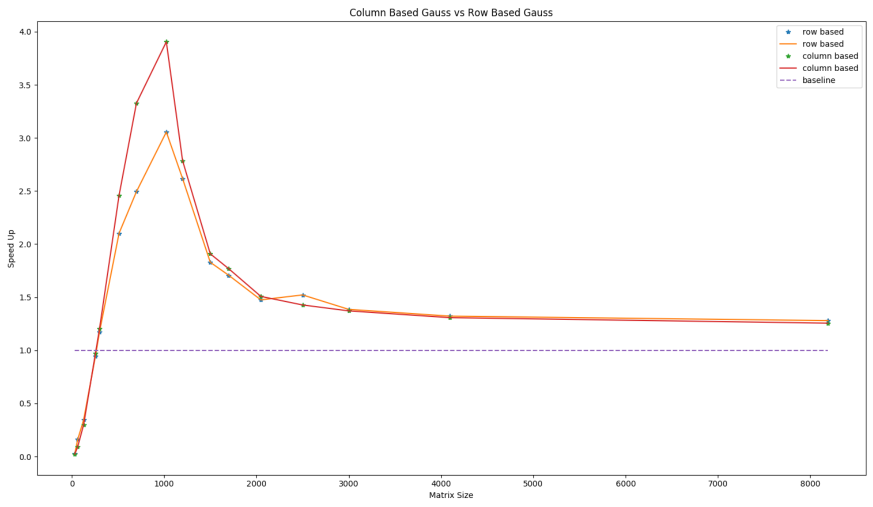
Use map to parallelize the elimination could get the maximum speedup (over 200%) when the dimension of matrix is 1024. And it reach over 27% speedup for the largest matrix.

Substitution

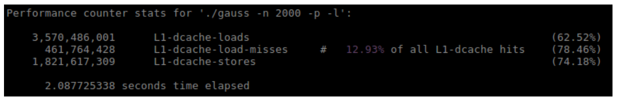


Use map to parallelize the substitution could get the positive speedup for matrix larger than 1200, though the speedup is not significant. The speedup for largest matrix is less than 1%.

Column Based Gauss



Use column based data structure to save the matrix could improve the locality because it will achieve higher hit rate for maximum searching in the same column. The experiment shows that column based matrix could achieve high maximum speedup (about 290%) for 1024 dimension matrix. But the final speedup of column based matrix is slightly lower (24% vs 27%).

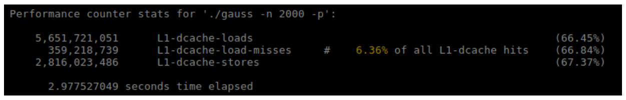


```
Performance counter stats for './gauss -n 2000 -p -l':

    3,570,486,001      L1-dcache-loads                                               (62.52%)
      461,764,428      L1-dcache-load-misses     #   12.93% of all L1-dcache hits    (78.46%)
    1,821,617,309      L1-dcache-stores                                              (74.18%)

      2.087725338 seconds time elapsed
```

Figure 1: Locality of Column Based Gauss



```
Performance counter stats for './gauss -n 2000 -p':

    5,651,721,051      L1-dcache-loads                                               (66.45%)
      359,218,739      L1-dcache-load-misses     #    6.36% of all L1-dcache hits    (66.84%)
    2,816,023,486      L1-dcache-stores                                              (67.37%)

      2.977527049 seconds time elapsed
```

Figure 2: Locality of Row Based Gauss

I used the perf tool to measure the real hit rate of L1 cache. The above figures show the results. It shows that column based gauss could get 2 times higher hit rate to calculate 2000 dimension matrix compared to row based gauss (12..93% vs 6.36%).