CS 211: High Performance Computing Project 2

Zhenyu Yang (862187998)

Fall 2019

Solving Large Linear Systems

1.1 naive GE

The basic idea is using convenience of triangular matrix to solve large linear equations. For basic LU factorization, it requires $\frac{2n^3}{3}$ operations. For two substitutions(forward and backward), it totally requires $2n^2$ operations. The performance (i.e., Gflops) of my naive LU factorization and LAPACK version are shown as follows.

Table 1: **LAPACK** Performance

n	runtime(s)	performance(Gflops)
1002	0.131762	5.10529
2001	0.541297	9.88246
3000	1.712678	10.5204
4002	3.496786	12.2292
5001	6.609571	12.6231

Table 2: **naive LU** Performance

n	runtime(s)	performance(Gflops)
1002	4.064462	0.165503
2001	32.843274	0.162875
3000	111.804068	0.161157
4002	264.094468	0.161922
5001	516.365470	0.161578

1.2 Blocked GEPP

Blocked GEPP using BLAS3 to maximize performance. First, I finished a basic version, and then, I tried some basic techniques to optimize the code. Here is what I did.

1. Skip computations of $LL^{-}1$, directly calculate **A(ib:end , end+1:n)**. Tile the computations for **A(ib:end , end+1:n)** to maximize cache reuse.
2. Tile the **A(ib:n, ib:end)** factorization to maximize cache and registers reuse.
3. Use BLAS3 matrices multiplication to compute **A(end+1:n , end+1:n)**
4. reduce the parameters of mydgemm, and try to make it inline(maybe not work at -O0 which without compiler optimization)

All of this bring me almost **0.1s** performance improvement in **n = 2001**(almost **5%** improvement).
To find out which part has potential to continue to be optimized deeply. I tested different parts of this code. The result shows that the dgemm part is the **bottleneck** of performance, **it costs almost 1.8s when n = 2001(total is 1.84, which means it consumes almost 99% time)**.
the result of optimization Blocked GEPP are shown as follows.

Table 3: **Blocked GEPP** Performance

n	runtime(s)	performance(Gflops)
1002	1.817282	0.370159
2001	14.514987	0.368539
3000	49.464173	0.364264
4002	117.492669	0.363961
5001	228.624235	0.364937

Table 4: **naive LU** Performance

n	runtime(s)	performance(Gflops)
1002	4.064462	0.165503
2001	32.843274	0.162875
3000	111.804068	0.161157
4002	264.094468	0.161922
5001	516.365470	0.161578

Comparing with non-block naive LU version, blocked GEPP improves almost 100% performance.

1.3 try SSE instructions

To improve the dgemm performance, I also tried to use 3x3 SSE instructions computations Unfortunately, the results get worse. The SSE instruction has more latency than c code, **dgemm cost almost 5s when n=2001(worse 3 times)**.