5451 LAB1 Ruoyan Kong, Part A, kong0135@umn.edu
(a) Matrix-Matrix products in openMP
Test on phi02.cselabs.umn.edu

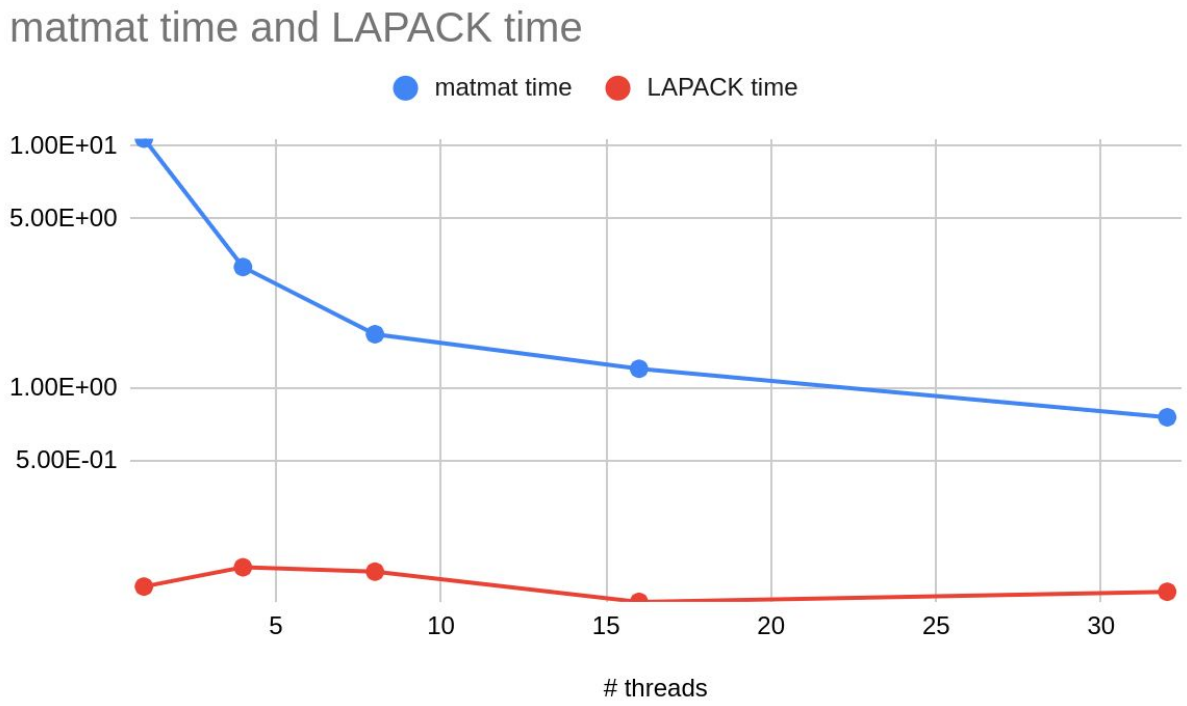
1. Explain what you did to improve performance.
   1) Transfer C matrix initilization to parallel mechanism.
   2) Transfer the outer loop i1 to parallel mechanism.
   3) Transpose B to B_T so B_T will be retrieved in the columnwise order, which is easier for chaching store.

```
int mat_mat_product(int m, int n, int k, double *A, double *B,
                        double *C){
  int nt, i1, i2, i3;
  #pragma omp parallel
  {
    nt = omp_get_num_threads();
  }
  double *B_T;
  B_T = (double*) malloc(k*n*sizeof(double));
  transpose(B, B_T, k, n);  // 3
  printf("Run with %d threads\n",nt);
  /* loop to perform:   C(i,j) = sum_t A(i,t) * B(t,j)
   * Outer loop i1: rows of A
   */
  /*-------------------- set all of C to zero first*/

  /*-------------------- Inner loop, linear comb of rows  of B */
  # pragma omp parallel for
  for( i1 = 0 ; i1 < m*n ; i1 ++) {                                    // 1
    C[i1] = 0.0;
  }
  /*--------------------row i1 of C == lin comb. of rows of B*/

  # pragma omp parallel for
  for( i1 = 0 ; i1 < m ; i1 ++)       {    // 2
     for( i3 = 0 ; i3 < n ; i3 ++ ) {
           for( i2 = 0 ; i2 < k ; i2 ++) {
              C[i3+i1*n] += A[i2+i1*k]*B_T[i2 +i3*n];           // 3
           }
     }
   }
  return 0;
}
```

2. Run your code with 1, 4, 8, 16, 32 threads and plots the curve of the times you get versus the number of threads. On the same figure plot a horizontal line that shows the time you get with Lapack.



3. Calculate the best flops rate you get with your code.
#threades = 32, matmat time=7.53E-01
flop rates (MFLOPS) = (2*m/100.0*n/100.0*k/100.0+m*n/1000000.0+k*n/1000000.0)/matmat time = 2.12E+04 (m=n=k=2000)

4. When you increase the dimension to a larger matrix [double the size] does the performance drop? Can you give a possible explanation as to the reason?
Yes, when I increase dimension to 8000, time spent = 7.999595e+01, the MFLOPS drop to 1.280225e+04 when #threads = 32 (compared to 2.12E+04).
It's because the cache will be missing for more times if we increase the dimension, it will go to memory to retrieve data, which is time-consuming.