Author: Ca11me1ce

Problem 1

a) If the the program is executed without pipeline, the lower bound on execution time that is (5+3+3)*m which is based on the math performed.

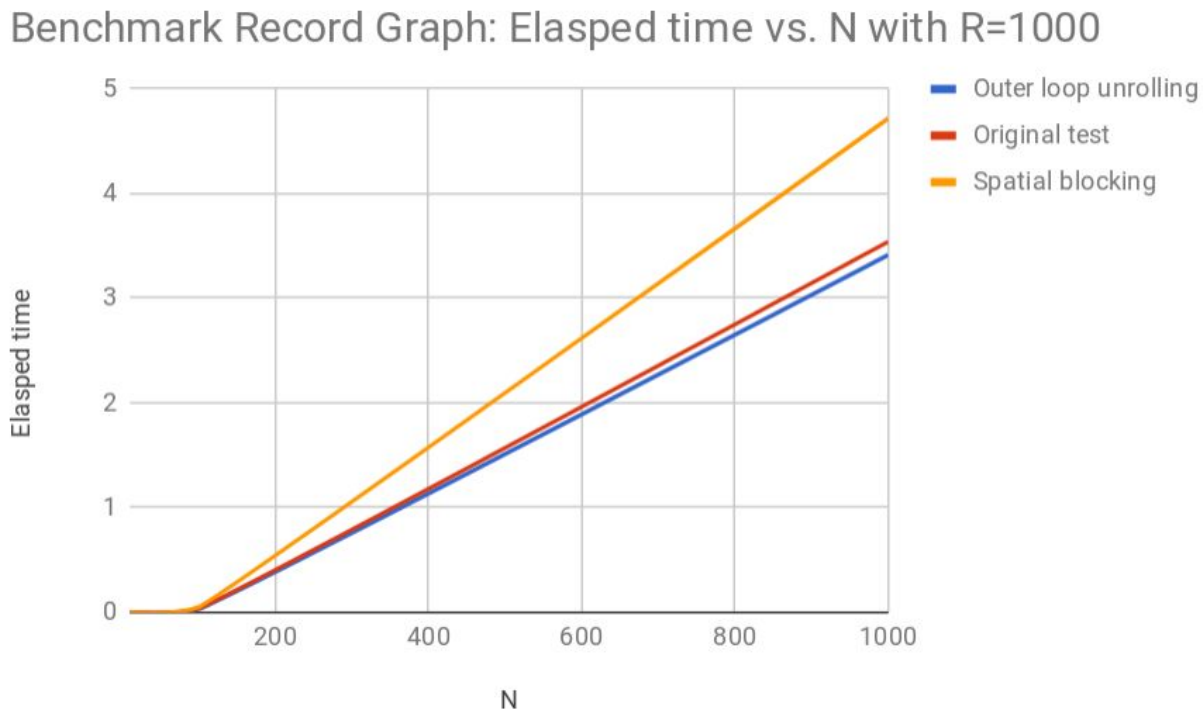
b) int i=0;
A[i] = B[i] * C[i] + D[i] + i;
A[i+1] = B[i+1] * C[i+1] + D[i+1] + (i+1);
A[i+2] = B[i+2] * C[i+2] + D[i+2] + (i+2);
…
…
A[m-1] = B[m-1] * C[m-1] + D[m-1] + (i+m-1);
Use the code above to instead the for-loop of the program.

c) According to the Amdahl's Law S(s)=1/((1-p)+(p/s)), when s is approached to m and m is approached to infinity, the program will run with highest speed, that speed will approached to the time of one operation of one multiply latency and two add latency in the program. So the lower bound is 5+3+3=11 clock cycles.

Problem 2

1. Original code without any change, the elapsed time is 1.391085.
2. Add another parallel expression to compute v_mat[i+1][i+1] in for-loop of myfunc, then change the increment to j+=2 and i+=2, so the elapsed time is 0.613169.
3. Take the declaration of i and j out of for-loop, the elapsed time is 0.594082.
4. Because sin(d_val)*sin(d_val)-cos(d_val)*cos(d_val)=sin^2(d_val)-cos^2(d_val)=(-cos(2*d_val)), therefore I use -cos(2*d_val) to instead the long expression of sin and cos, the elapsed time is 0.234944.
5. Remove round() of d_val, then I used typecasting of integer to instead of that, so the elapsed time is 0.230913.
6. In -cos(2*d_val), I changed (2*d_val) to (d_val+d_val), so the elapsed time is 0.223052.

Problem 3



According to this record graph, the outer loop unrolling has the highest speed among these three tests, that can show the parallel programming is the faster than other designs. Then nested for-loop will reduce the speed of a program. When N increases and R=1000, the result will show more clearly among these three tests.