Exercise 3

Shruti Shivakumar (sshivakumar9)

September 1, 2017

Code listing

```
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>
#include <sys/time.h>
#include <string.h>

double get_walltime(void)
{
  struct timeval tp;
  gettimeofday(&tp, NULL);

  return (double) (tp.tv_sec + tp.tv_usec/1.e6);
}

int main(int argc, char** argv)
{
        int i, j, N=10000, x=1, p;
        double x_x[10000] = {(double)rand()/(double)RAND_MAX},
        x_y[10000] = {(double)rand()/(double)RAND_MAX},
        x_z[10000] = {(double)rand()/(double)RAND_MAX};
        double x_xs[10000], x_ys[10000], x_zs[10000];
        memcpy(x_xs, x_x, 10000*sizeof(double));
        memcpy(x_ys, x_y, 10000*sizeof(double));
        memcpy(x_zs, x_z, 10000*sizeof(double));
        double avg[11], sum;
        avg[0] = 0;
        double a = pow(0.0002,0.5);
        double time = -get_walltime();

        for(j=1;j<=5000;j++)
        {
                #pragma omp parallel for
                for(i=0;i<N;i++)
                {
                        x_x[i] = x_x[i] +
                a*(-1. + 2.*((double)rand()/(double)RAND_MAX));
                        x_y[i] = x_y[i] +
                a*(-1. + 2.*((double)rand()/(double)RAND_MAX));
                        x_z[i] = x_z[i] +
                a*(-1. + 2.*((double)rand()/(double)RAND_MAX));
                }

                if(j%500 == 0)
```

```
                {
                        // printf("x: %d\n", x);
                        #pragma omp parallel for reduction(+:sum)
                        for(p=0;p<N;p++)
                                sum += pow(pow((x_x[i] - x_xs[i]),2) +
                        pow((x_y[i] - x_ys[i]),2) +
                    pow((x_z[i] - x_zs[i]),2),0.5);
                        avg[x] = sum/=N;
                        x++;
                        memcpy(x_xs, x_x, 10000*sizeof(double));
                        memcpy(x_ys, x_y, 10000*sizeof(double));
                        memcpy(x_zs, x_z, 10000*sizeof(double));
                }
        }
        printf("Average distance moved: \n");
        for(i=0;i<11;i++)
                printf("%f\n", avg[i]);
        time += get_walltime();
        printf("Time taken : %g\n", time);
        return 0;
}
```

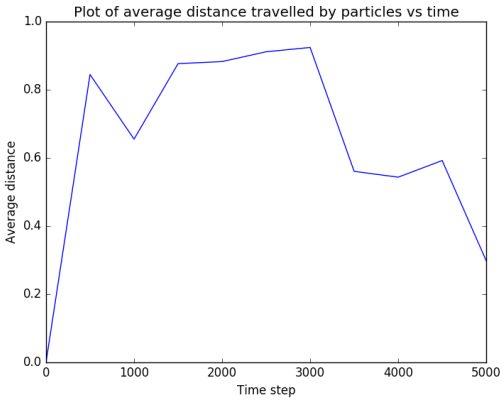

Figure 1: Plot of average distance travelled vs time step

Execution time (secs)	Number of threads	Loop scheduling option
8.053	3	static
15.249	4	static
9.547	3	dynamic
7.441	3	guided

Table 1: Table of run time for different loop scheduling options with best number of threads