Claudio Del Valle, Konstantin Zelmanovich

Dr. Nagarajan Kandasamy

ECEC 413 - Introduction to Parallel Computing Architecture

Assignment 3

May 5, 2019

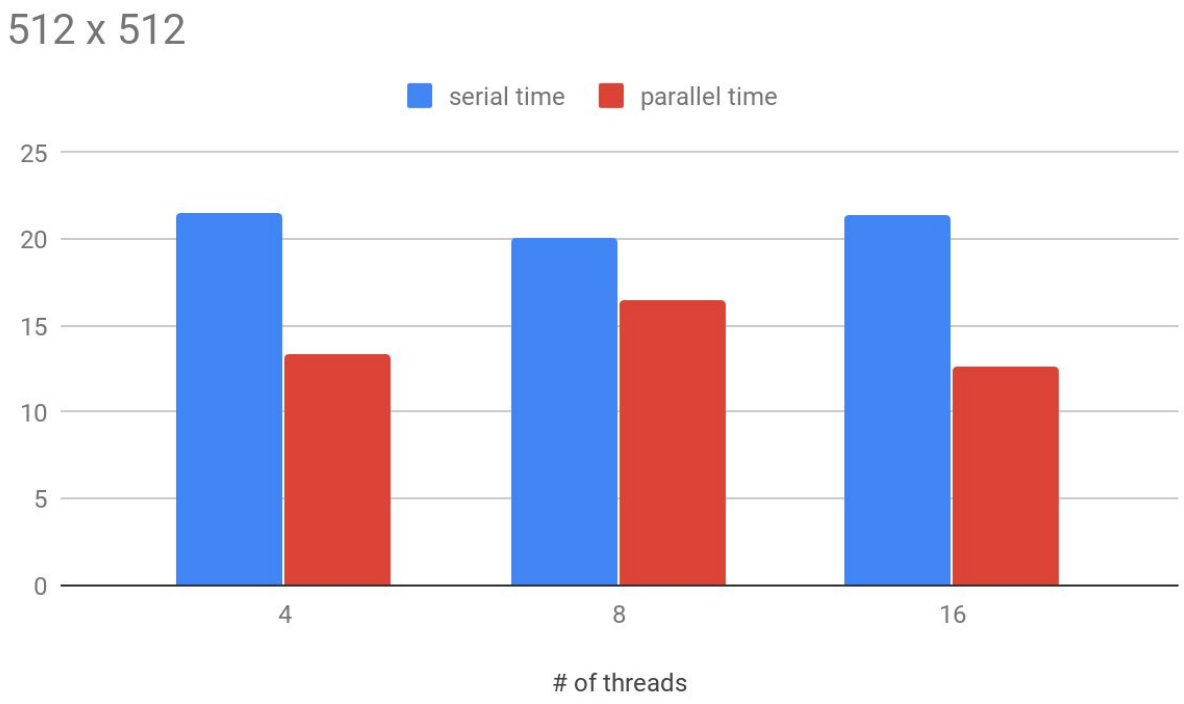
Part 2 - OpenMP: Jacobi Heat Diffusion

```
int
compute_using_omp_jacobi (grid_t *grid, int num_threads)
{
        int num_iter = 0;
        int done = 0;
        int i, j;
        double diff;
        float old, new;
        float eps = 1e-2; /* Convergence criteria. */
        int num_elements = (grid->dim - 2) * (grid->dim - 2);
        grid_t *grid_copy = copy_grid(grid);
        grid_t *temp;

        while(!done) { /* While we have not converged yet. */
                #pragma omp parallel num_threads(num_threads) private(new, old)  shared(grid, grid_copy, temp, diff)
                {
                        double local_diff = 0.0;
                        #pragma omp for schedule(static)
                        for (i = 1; i < (grid->dim - 1); i++) {
                                for (j = 1; j < (grid->dim - 1); j++) {
                                        old = grid->element[i * grid->dim + j]; /* Store old value of grid point. */

                                        new = 0.25 * (grid->element[(i - 1) * grid->dim + j] +\
                                        grid->element[(i + 1) * grid->dim + j] +\
                                        grid->element[i * grid->dim + (j + 1)] +\
                                        grid->element[i * grid->dim + (j - 1)]);

                                        grid_copy->element[i * grid->dim + j] = new; /* Update the grid-point value. */
                                        local_diff = local_diff + fabs(new - old); /* Calculate the difference in values. */
                                }
                        }

                        #pragma omp critical
                        diff += local_diff;


                        temp = grid;
                        grid = grid_copy;
                        grid_copy = temp;
                }

                diff = diff / num_elements;
                /* End of an iteration. Check for convergence. */
                printf ("Iteration %d. DIFF: %f.\n", num_iter, diff);
                num_iter++;

                if (diff < eps)
                        done = 1;
        }


    return num_iter;
}
```

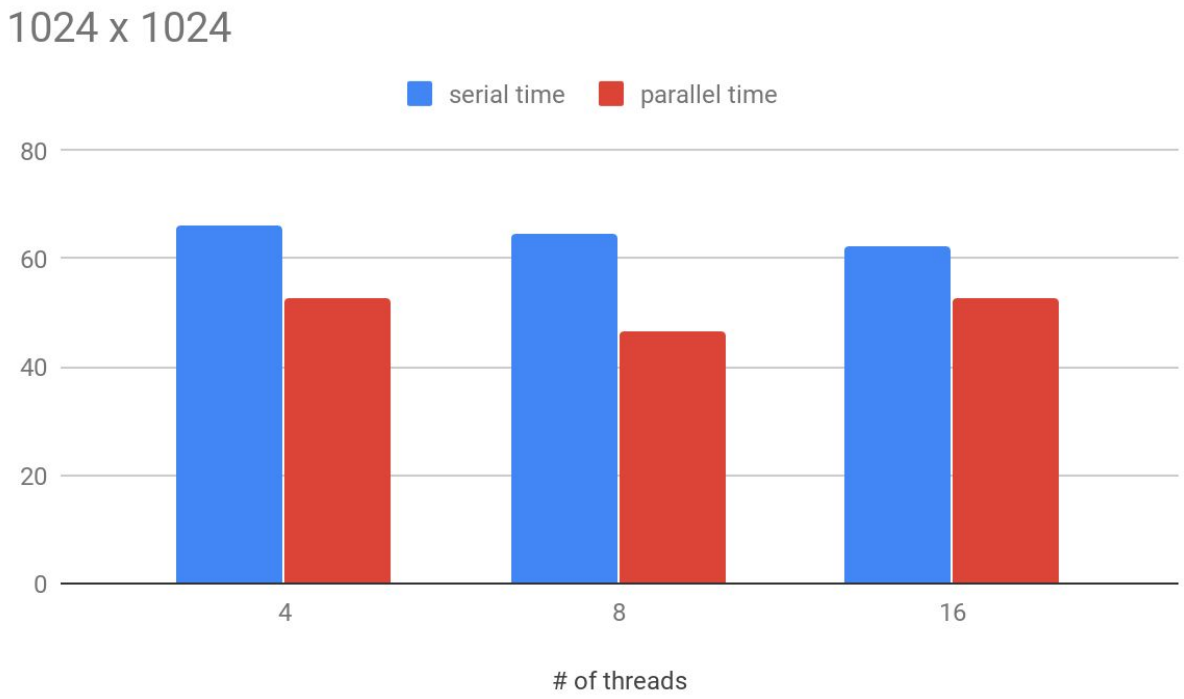
Figure 1. Modified OpenMP Method for Jacobi Heat Diffusion

Results

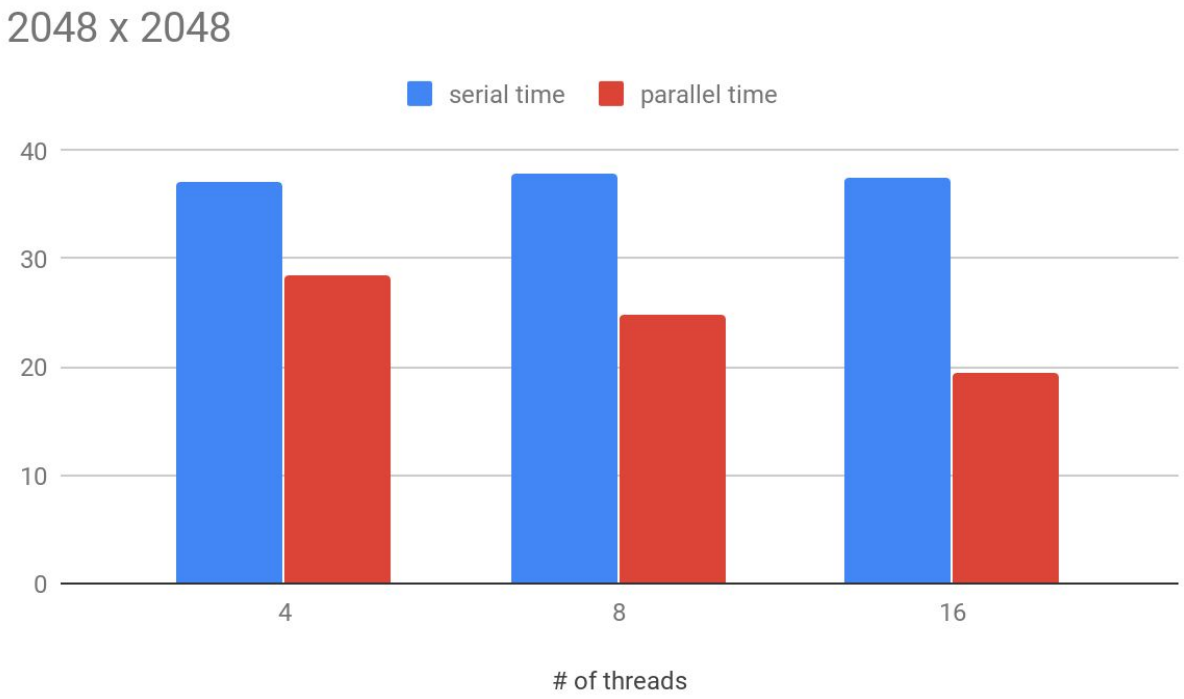
512 x 512			
# of threads	serial time	parallel time	speedup
4	21.52	13.35	1.62
8	20.12	16.43	1.23
16	21.33	12.64	1.69



1024 x 1024			
# of threads	serial time	parallel time	speedup
4	65.94	52.86	1.25
8	64.59	46.45	1.4
16	62.37	52.72	1.19



2048 x 2048			
# of threads	serial time	parallel time	speedup
4	37.1	28.45	1.31
8	37.88	24.72	1.54
16	37.51	19.54	1.92


2048 x 2048
serial time
parallel time
40
30
20
10
0
4
8
16
# of threads