Konstantin Zelmanovich, Claudio Del Valle A

Dr. Nagarajan Kandasamy

ECEC 413 – Introduction to Parallel Computer Architecture

Assignment 2

April 28, 2019

Part 1 - Pthreads: Jacobi Solver

```
int
compute_using_pthreads_jacobi (grid_t *grid, int num_threads)
{
    pthread_t *tids = (pthread_t *) malloc (num_threads * sizeof(pthread_t));
    pthread_attr_t attributes;
    pthread_attr_init (&attributes);

    grid_t *grid_copy = copy_grid(grid);

    int *iterations = (int *) malloc (num_threads * sizeof(int));

    int i;
    thread_params *params = (thread_params *) malloc (num_threads *
sizeof(thread_params));

    if (grid->dim % num_threads) {
        for (i = 0; i < num_threads - 1; i++) {
            params[i].tid = i;
            params[i].chunk_size = grid->dim / num_threads;
            params[i].iterations = iterations;
            params[i].grid_ping = grid;
            params[i].grid_pong = grid_copy;
        }
        params[num_threads-1].tid = num_threads-1;
        params[num_threads-1].chunk_size = grid->dim % num_threads;
        params[num_threads-1].iterations = iterations;
        params[num_threads-1].grid_ping = grid;
        params[num_threads-1].grid_pong = grid_copy;
    } else {
        for(i = 0; i < num_threads; i++) {
            params[i].tid = i;
            params[i].chunk_size = grid->dim / num_threads;
            params[i].iterations = iterations;
            params[i].grid_ping = grid;
            params[i].grid_pong = grid_copy;
        }
    }
```

```
    for (i = 0; i < num_threads; i++)
        pthread_create (&tids[i], &attributes, compute_jacobi, (void *)
&params[i]);

    for (i = 0; i < num_threads; i++)
        pthread_join (tids[i], NULL);

    int total_iterations = 0;
    for (i = 0; i < num_threads; i++)
        total_iterations = total_iterations + iterations[i];

    free ((void *) params);


    return total_iterations;;
}
```

**Table 1**: Timing results for 100-150

| # of Threads | Grid Size | Serial Time (s) | Parallel Time (s) | Speedup |
|---|---|---|---|---|
| 4 | 512 | 0.7 | 0.14 | 5 |
| 4 | 1024 | 0.48 | 0.11 | 4.36 |
| 4 | 2048 | 0.41 | 0.09 | 4.55 |
| 8 | 512 | 0.63 | 0.11 | 5.72 |
| 8 | 1024 | 0.46 | 0.06 | 7.66 |
| 8 | 2048 | 0.39 | 0.14 | 2.78 |
| 16 | 512 | 0.64 | 0.14 | 4.57 |
| 16 | 1024 | 0.44 | 0.06 | 7.33 |
| 16 | 2048 | 0.39 | 0.88 | 0.44 |
| 32 | 512 | 0.73 | 0.11 | 6.63 |
| 32 | 1024 | 0.42 | 0.07 | 6 |
| 32 | 2048 | 0.35 | 0.8 | 0.43 |

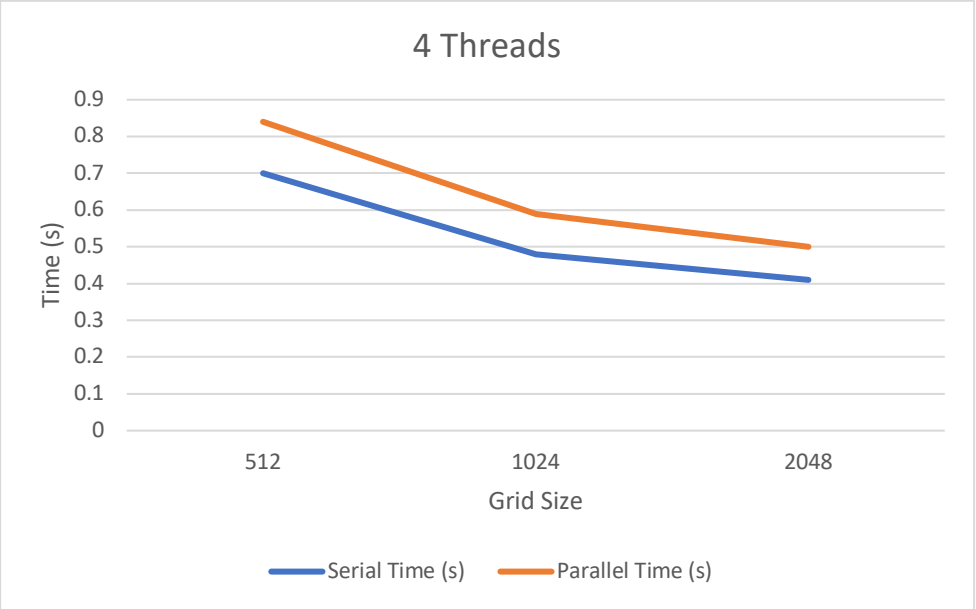


**Figure 1:** Time vs Matrix Size using 4 Threads

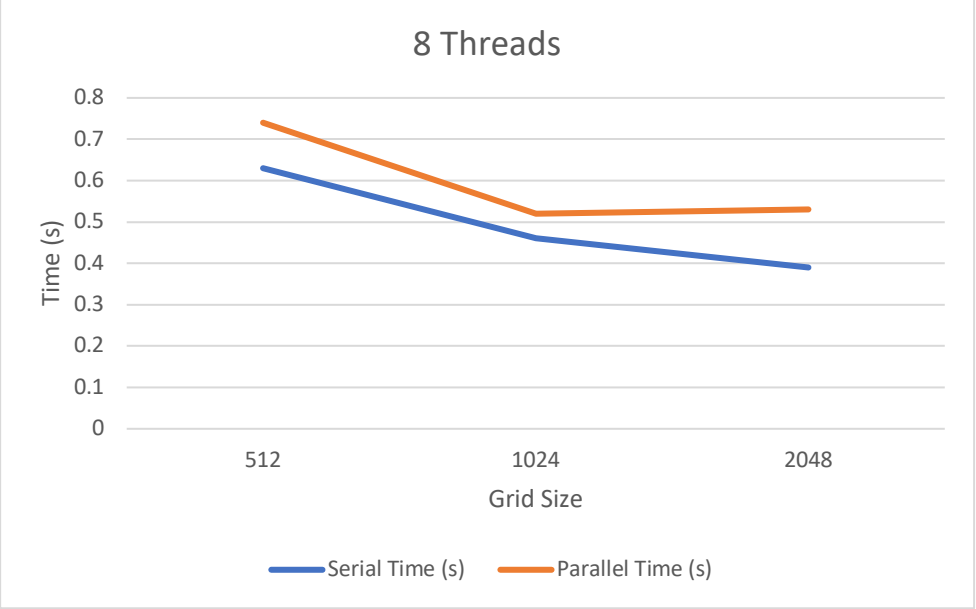


**Figure 2:** Time vs Matrix Size using 8 Threads

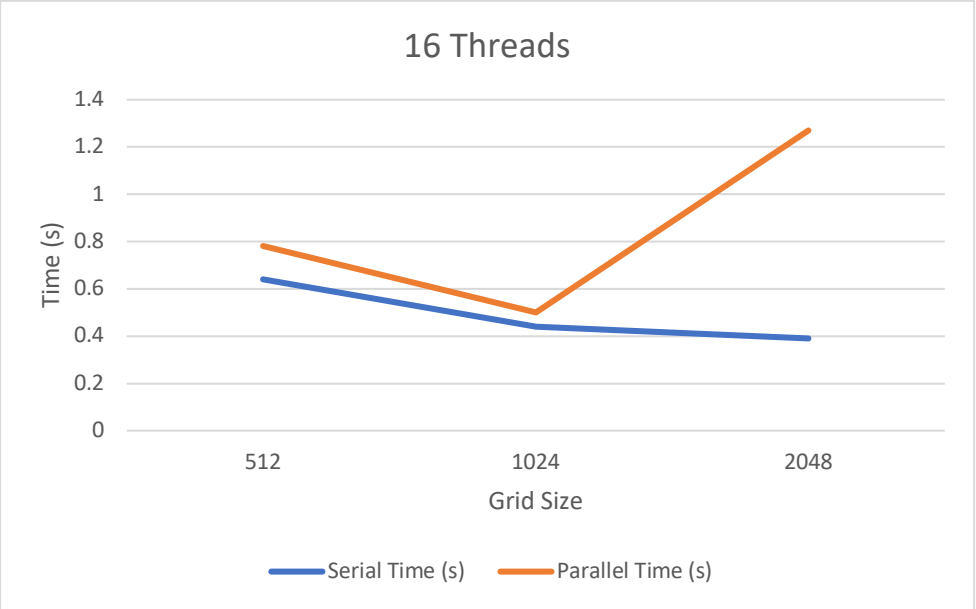


**Figure 3:** Time vs Matrix Size using 16 Threads

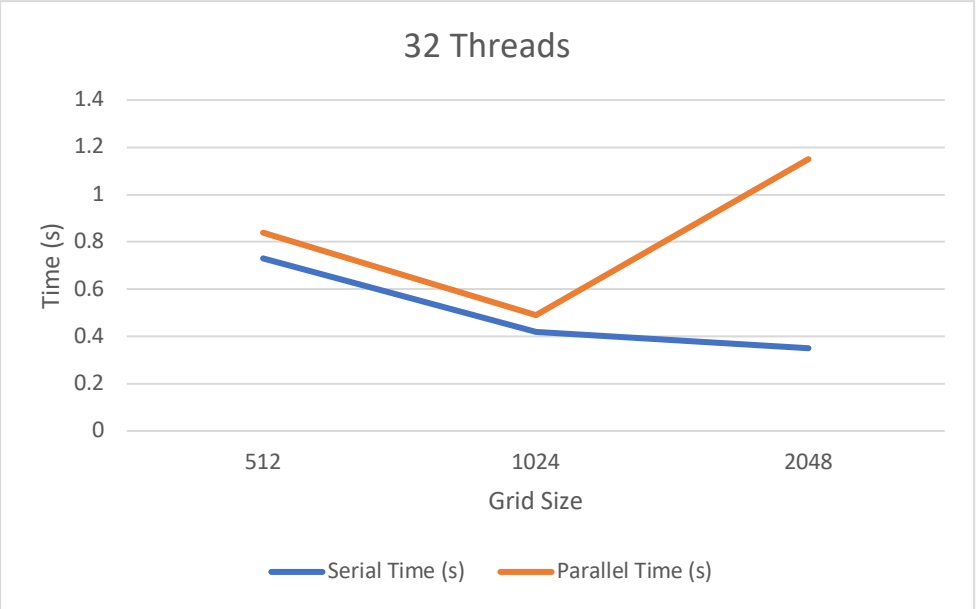


**Figure 4:** Time vs Matrix Size using 32 Threads

**Table 2**: Timing results for 1000-1500

| # of Threads | Grid Size | Serial Time (s) | Parallel Time (s) | Speedup |
|---|---|---|---|---|
| 4 | 512 | 19.98 | 27.55 | 0.72 |
| 4 | 1024 | 63.6 | 18.81 | 3.38 |
| 4 | 2048 | 34.2 | 23.94 | 1.42 |
| 8 | 512 | 20.88 | 28.59 | 0.73 |
| 8 | 1024 | 60.24 | 17.83 | 3.37 |
| 8 | 2048 | 35.56 | 38.72 | 0.91 |
| 16 | 512 | 19.98 | 27.65 | 0.72 |
| 16 | 1024 | 63.12 | 18.69 | 3.37 |
| 16 | 2048 | 33.96 | 39.18 | 0.86 |
| 32 | 512 | 21.21 | 28.48 | 0.74 |
| 32 | 1024 | 60.6 | 18.68 | 3.24 |
| 32 | 2048 | 35.52 | 31.56 | 1.12 |

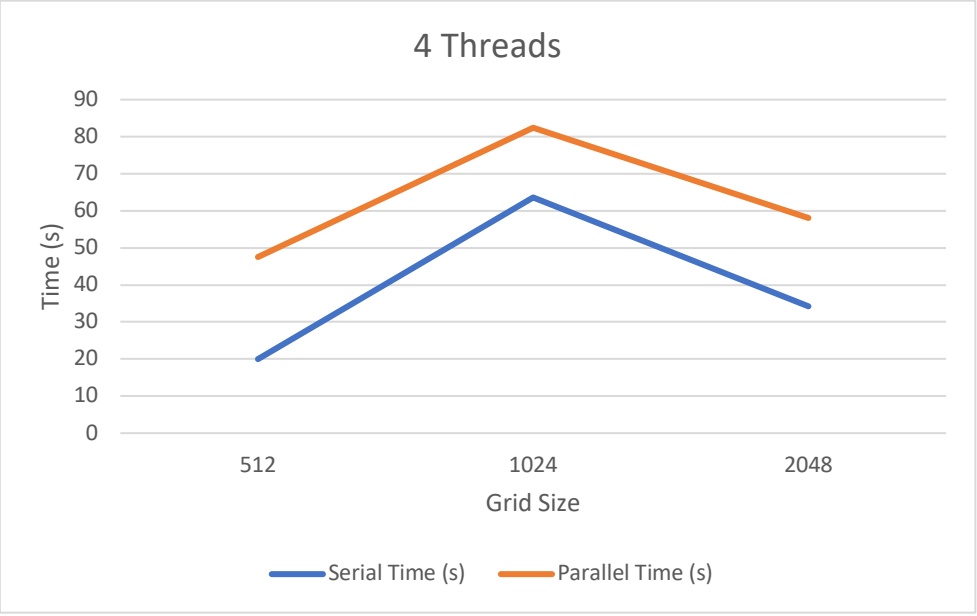


**Figure 5:** Time vs Matrix Size using 4 Threads

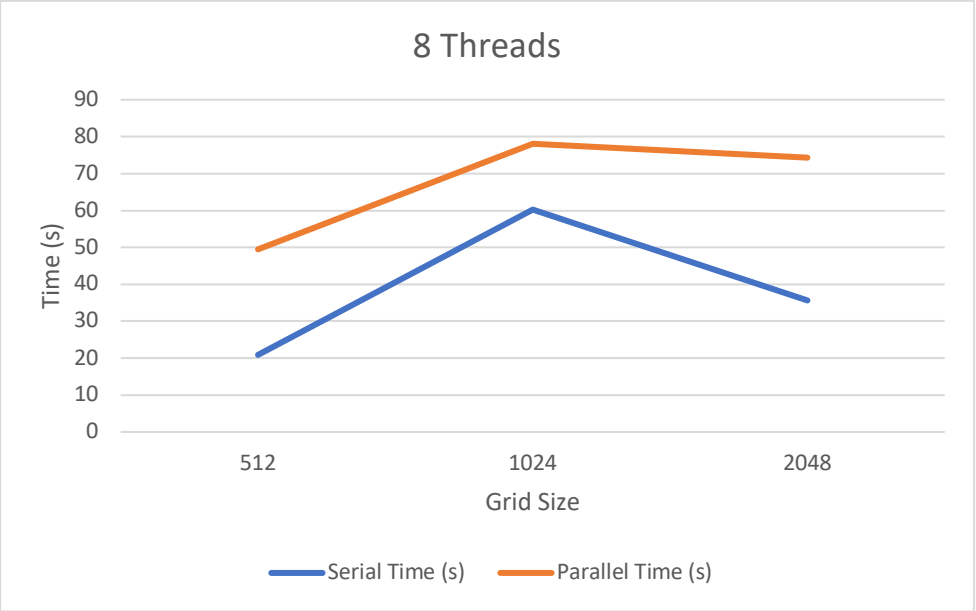


**Figure 6:** Time vs Matrix Size using 8 Threads

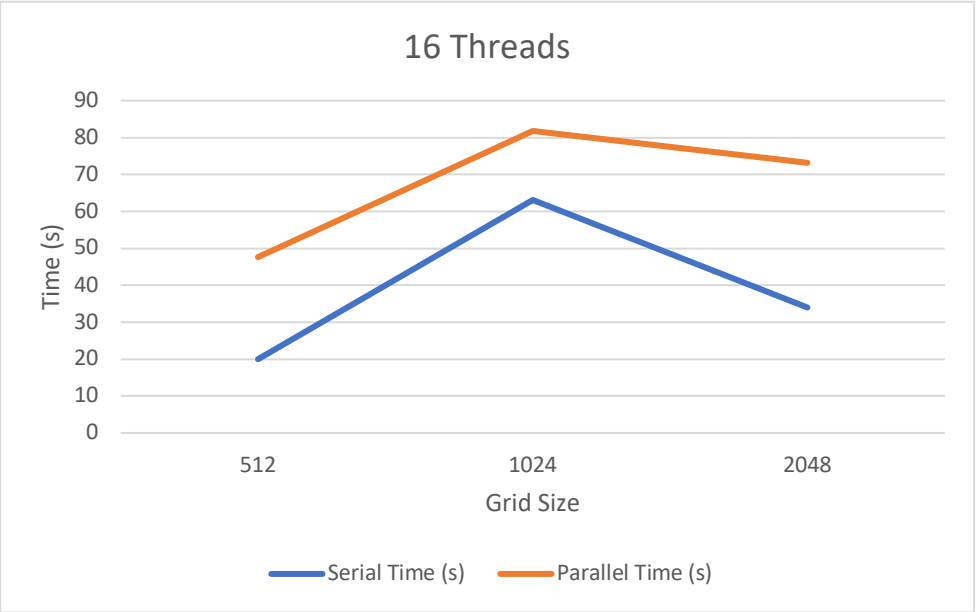


**Figure 7:** Time vs Matrix Size using 16 Threads

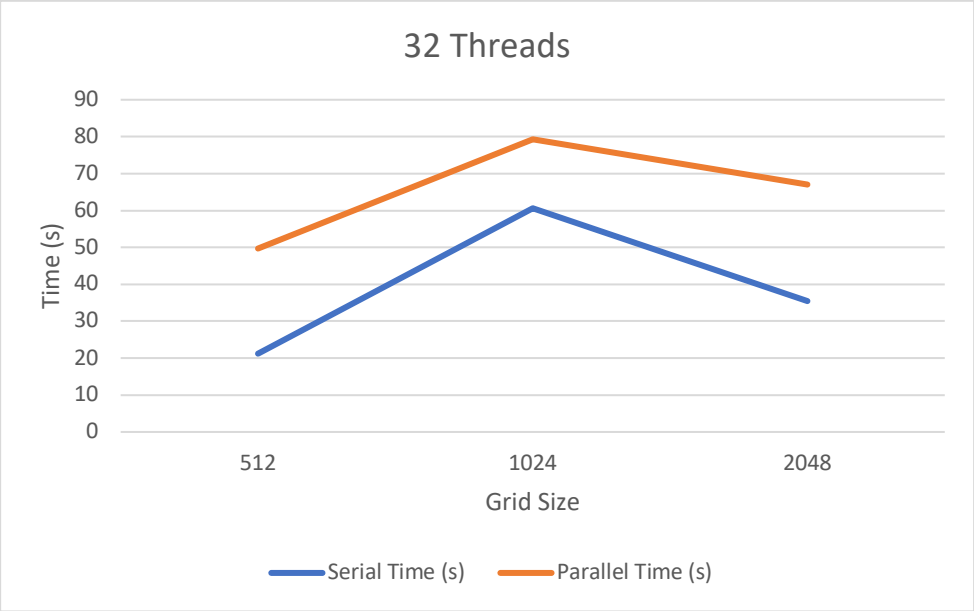


**Figure 8:** Time vs Matrix Size using 32 Threads