Claudio Del Valle, Konstantin Zelmanovich

Dr. Nagarajan Kandasamy

ECEC 413 - Introduction to Parallel Computing Architecture

Assignment 1

April 21, 2019

Part 1 - Pthreads: Numerical Integration

```c
double
compute_using_pthreads (float a, float b, int n, float h, int num_threads)
{
    pthread_t *tids = (pthread_t *) malloc (num_threads * sizeof(pthread_t));
    pthread_attr_t attributes;
    pthread_attr_init (&attributes);

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

    for (i = 0; i < num_threads; i++) {
        params[i].tid = i;
        params[i].num_threads = num_threads;
        params[i].a = a;
        params[i].b = b;
        params[i].num_traps = n;
        params[i].base_length = h;
        params[i].partial_integral = partial_integral;
    }

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

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

    double integral = 0.0;

    integral = (f(a) + f(b))/2.0;
    for (i = 0; i < num_threads; i++)
        integral += partial_integral[i];

    free ((void *) params);
    free ((void *) partial_integral);

    return integral * h;
}
```

Figure 1. Modified Pthread Method for Numerical Integration

```
void *
compute_integral (void *args) {
    thread_params *params = (thread_params *) args;
    double partial_integral = 0.0;
    int i;

    for (i = 1 + params->tid; i <= params->num_traps - 1; i+=params->num_threads)
        partial_integral += f((params->a + i) * params->base_length);
    params->partial_integral[params->tid] = partial_integral;

    pthread_exit (NULL);
}
```

Figure 2. Method Run by Individual Threads

Explanation of Code

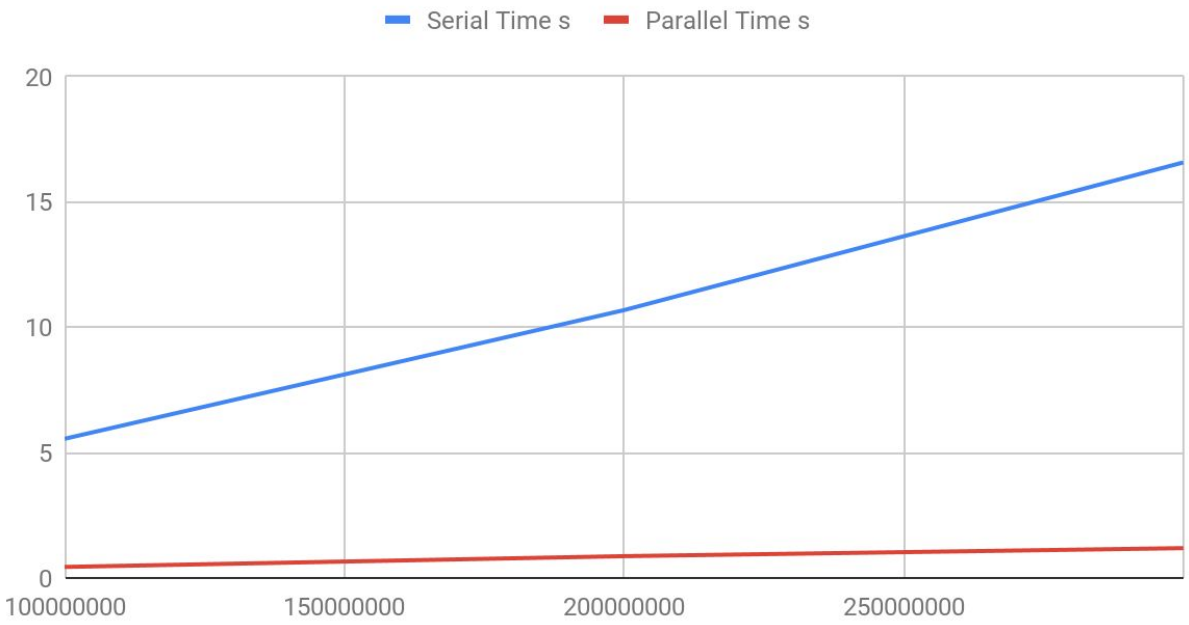
As figure 1 shows, a simple thread parameter structure holds a shared variable ‘partial_integral’ that each thread will access to store their portion of the computed integral. The computation is shown in figure 2. In this case, striding was used so that threads would compute portions of the integral using an offset value based on their thread ID.

Results

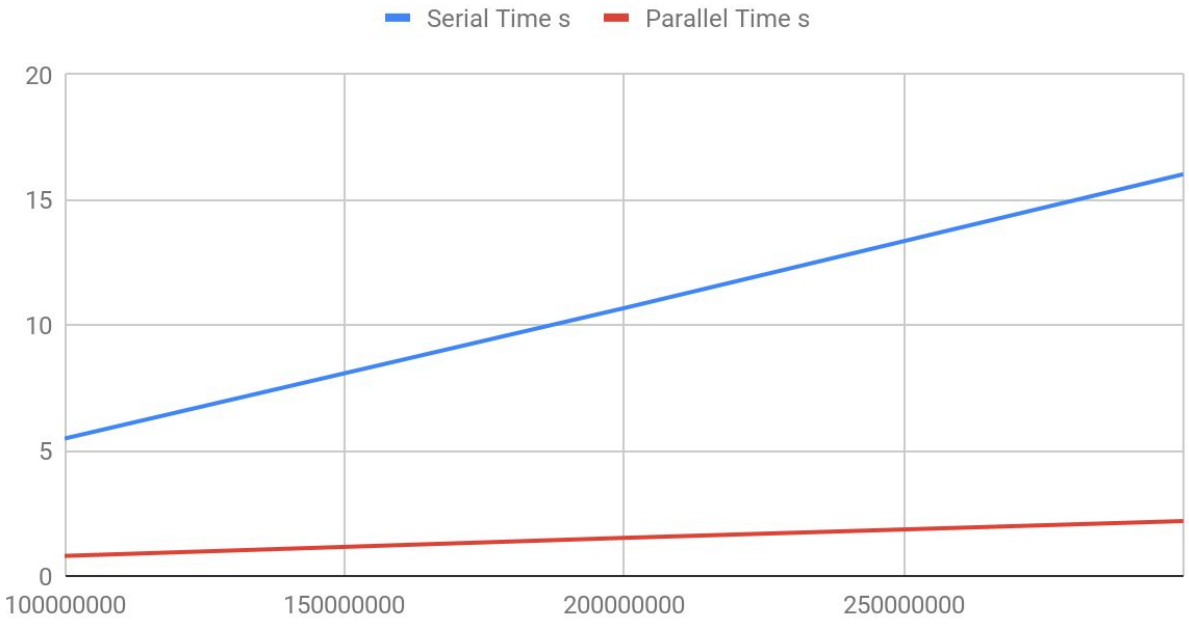
Table 1. Pthread Numerical Integration Results

# of Threads	# of Trapezoids	Serial Time s	Parallel Time s	Speedup
2	100000000	5.35	2.89	1.85
2	200000000	11.13	5.41	2.05
2	300000000	16.02	8.3	1.93
4	100000000	5.48	1.47	3.72
4	200000000	10.68	2.89	3.69
4	300000000	16.4	4.14	3.96
8	100000000	5.48	0.8	6.85
8	200000000	10.68	1.52	7.02
8	300000000	16.02	2.19	7.31
16	100000000	5.55	0.44	12.61
16	200000000	10.68	0.87	12.27
16	300000000	16.57	1.19	13.92

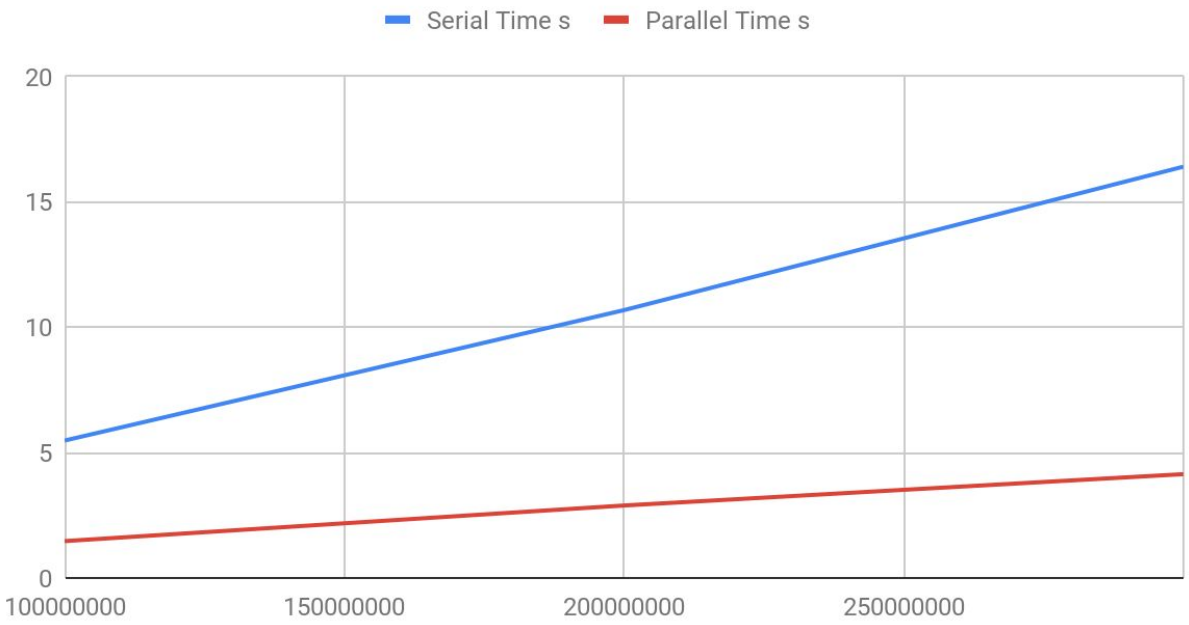
16 Threads



8 Threads



4 Threads



2 Threads