DREXEL UNIVERSITY
Electrical and Computer Engineering
College of Engineering

Drexel University

Electrical and Computer Engineering Dept.

ECEC-413

Numerical Integration

Chris Kasper
Prof. Naga Kandasamy
DATE: 04/21/19

Multi-threaded Design

To extend the design of the numerical integration (using the trapezoid rule) to a multi-threaded design, the work has to be divided among the number of threads. This was done as such:

```c
/* Split up work based on number of trapezoids equally across threads */
for (i = 0; i < num_threads; i++) {
    args_thread = (ARGS_FOR_THREAD *) malloc (sizeof (ARGS_FOR_THREAD));

    args_thread->threadID = i;
    args_thread->numThreads = num_threads;

    /* Determine number of trapezoids for each thread */
    if (i != num_threads-1) {
        args_thread->numTrapz = n / num_threads;
    } else {
        args_thread->numTrapz = n - (n / num_threads) * i;
    }

    args_thread->a = startA;
    args_thread->b = startA + (args_thread->numTrapz * h);
    args_thread->h = h;

    args_thread->result = data + i;

    startA = args_thread->b;

    if ((pthread_create (&worker[i], NULL, compute_gold_pthread, (void *) args_thread)) != 0) {
        printf ("Cannot create worker thread \n");
        exit (EXIT_FAILURE);
    }
}
```

Essentially, each thread gets allocated a certain number of trapezoids being used to calculate the integral of the equation. Each thread has a start point, end point, and gets the number of trapezoids (including base of each trapezoid) it has to compute. The end result is stored in a data array structure, which each thread has a pointer to, so they can store their result. Each thread has a different pointer, so thread 0 goes into data[0], thread 1 goes into data[1], etc.

The thread function has certain checks to ensure the computation is done properly. There is different logic for the first thread, middle threads (not the first or last), and the end thread. There is even a special case thread in the case only 1 thread is being used. The logic is structured as such:

```
/* Special case: If there is only 1 thread */
if ((args_thread->threadID == 0) && (args_thread->threadID == (args_thread->numThreads - 1))) {
    integral = (f(args_thread->a) + f(args_thread->b)) / 2.0;

    for (k = 1; k <= args_thread->numTrapz-1; k++) {
        integral += f(args_thread->a+k*args_thread->h);
    }

/* First thread processing */
} else if (args_thread->threadID == 0) {
    integral = f(args_thread->a) / 2.0;

    for (k = 1; k <= args_thread->numTrapz; k++) {
        integral += f(args_thread->a+k*args_thread->h);
    }

/* End thread processing */
} else if (args_thread->threadID == (args_thread->numThreads - 1)) {
    for (k = 1; k <= args_thread->numTrapz-1; k++) {
        integral += f(args_thread->a+k*args_thread->h);
    }

    integral += f(args_thread->b) / 2.0;

/* Middle thread processing */
} else {
    for (k = 1; k <= args_thread->numTrapz; k++) {
        integral += f(args_thread->a+k*args_thread->h);
    }
}
```

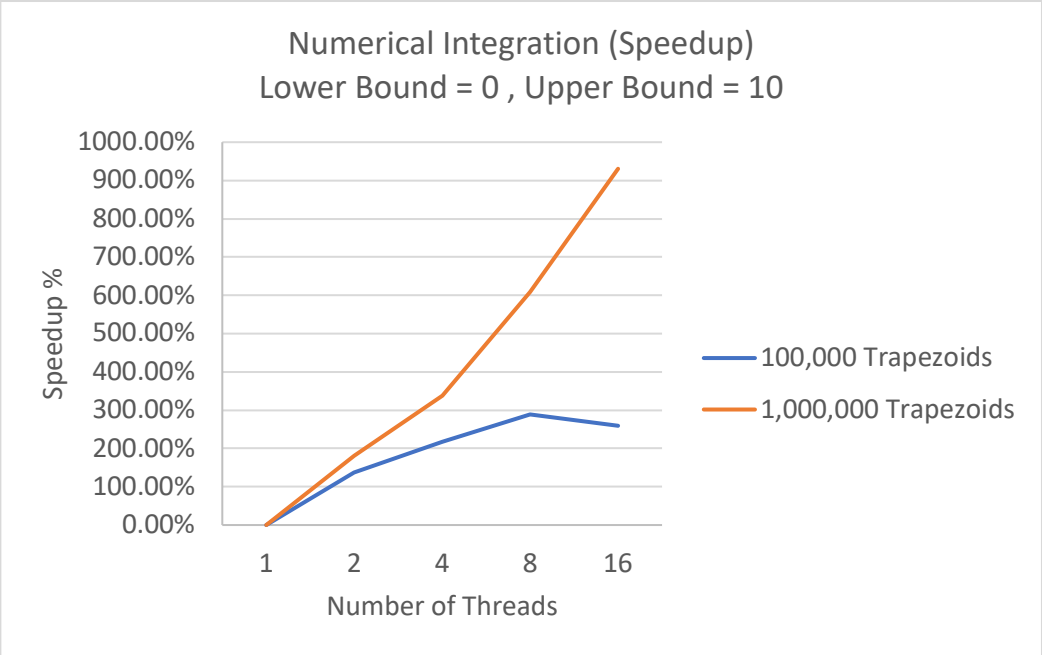
The thread function has certain checks to ensure the computation is done properly. There is different logic for the first thread, middle threads (not the first or last), and the end thread. There is even a special case thread in the case only 1 thread is being used.

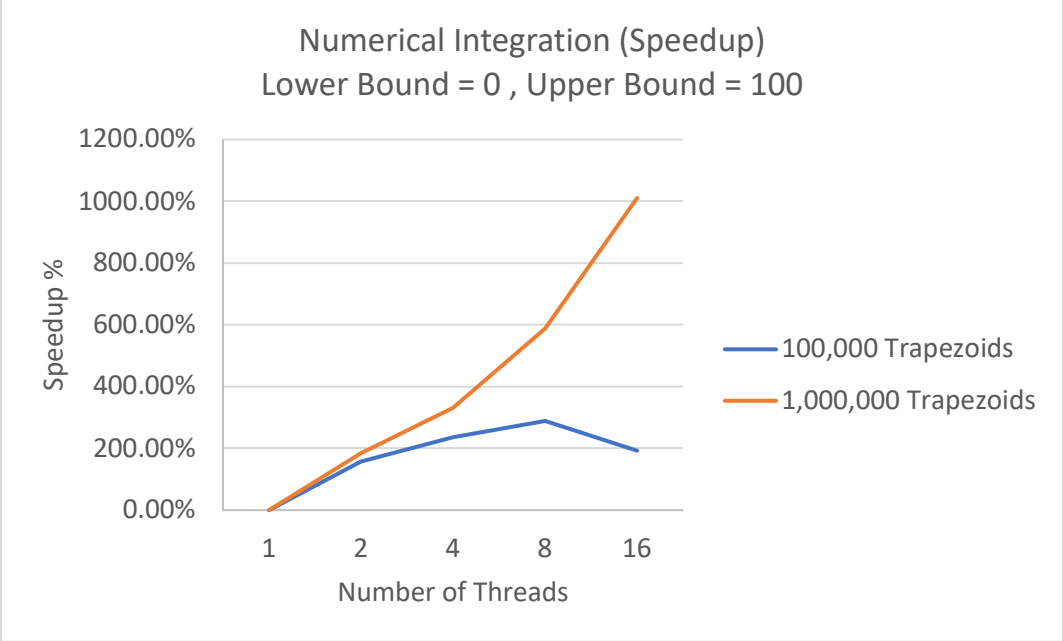
Once all the threads finish, there is a loop over the data structure to calculate the final result.

Speedup

Table 1: Speedup of integration on xunil-05

Number of Trapezoids	**Lower Limit**	**Upper Limit**	Single Thread	2 Threads	4 Threads	8 Threads	16 Threads
100000	0	10	0.0052s	0.0038s	0.0024s	0.0018s	0.0020s
100000	0	100	0.0052s	0.0033s	0.0022s	0.0018s	0.0027s
1000000	0	10	0.0456s	0.0253s	0.0135s	0.0075s	0.0049s
1000000	0	100	0.0465s	0.0251s	0.0140s	0.0079s	0.0046s


Numerical Integration (Speedup)
Lower Bound = 0 , Upper Bound = 10
1000.00%
900.00%
800.00%
700.00%
600.00%
500.00%
400.00%
300.00%
200.00%
100.00%
0.00%
Speedup %
1
2
4
8
16
Number of Threads
100,000 Trapezoids
1,000,000 Trapezoids


Numerical Integration (Speedup)
Lower Bound = 0 , Upper Bound = 100
1200.00%
1000.00%
800.00%
600.00%
400.00%
200.00%
0.00%
Speedup %
1
2
4
8
16
Number of Threads
100,000 Trapezoids
1,000,000 Trapezoids