Parallel Processing Lab 1: OpenMP

Guassian Elimination

Sarah Peachey & Nathan Schomer

February 9, 2018

1 Design

The Gaussian elimination function was initially parallelized by adding OpenMP directives before the first set of for loops and two of the nested for loops. However, these changes did not yield any improvement in run time. Next, the algorithm was implemented using a "ping pong" method as described by Dr. Kandasamy during lecture. Using this method, the dependencies between the elimination and division steps were removed since all calculations are based on values stored in a read-only matrix. These changes provided better performance over the original method. Pseudocode for the "ping pong" method is shown on the next page.

2 Results

Algorithm run time is dependent on the number of threads and current computer workload. This is evident from the speed-up values shown in the table. Speed up was calculated using Equation 1. Values for "Auto" threads were determined during the Eagles parade when the workload on the Xunil server were much lower. Currently running processes were monitored using "top". The remaining values were collected later when many more processes were running on Xunil. The most speed up was achieved when OpenMP determined the number of threads. Otherwise, the parallel version was generally slower than the serial algorithm. As expected, the slowdown was greater with a lower number of threads since the overhead incurred by OpenMP was greater than the additional work throughput gained by additional threads.

$$s = \frac{t_{serial}}{t_{parallel}} \tag{1}$$

Thread Count	1024x1024	2048x2048	4096x4096	8192x8192
Auto	1.46	1.24	1.08	1.08
2	0.31	0.27	0.29	0.31
4	0.55	0.52	0.41	0.55
8	0.63	0.89	0.87	0.89
16	1.20	0.89	0.85	0.90

Result: Ping Pong Gaussian Elimination

initialization;

for *each row in temp_array* **do**

    OMP Directive;

    for *each element temp_array* **do**

        if *first row* **then**

            Perform Division Step

        end

        else

            Perform Elimination Step

        end

    end

    if *Not last Iteration* **then**

        OMP Directive;

        copy U_array to temp_array

    end

end