SeanCooke8892@gmail.com

Multithreaded Linear Equation Solver

README

Purpose

Gauss-Jordan elimination is an algorithm used to solve a system of linear equations that can be parallelized. We will be creating speedup curves, recording speedup and recording efficiency for parallel solutions to Gauss-Jordan using OpenMP and Cilk.

A speedup curve is a graph of program runtime versus number of threads used. The steeper the negative gradient of the speedup curve, the better the parallelization of the algorithm. The data point for number of threads used equals 1 is the runtime of the most efficient sequential algorithm.

Let T(1) be the runtime of the most efficient sequential algorithm and T(p) be the runtime of the parallel algorithm on a p processor machine. The formula for Speedup S is shown below:

$$S = T(1) / T(p)$$

The formula for efficiency E is shown below:

$$E = S / p$$

Experiment

Descriptions of the four parallel Gauss-Jordan elimination algorithms written are shown below:

Algorithm	Scheduling	Chunk Size
OpenMP Cyclic	Static	3
OpenMP Blocked	Static	10
OpenMP Dynamic	Dynamic	Number of Processors (72)
Cilk	Dynamic	Number of Processors (72)

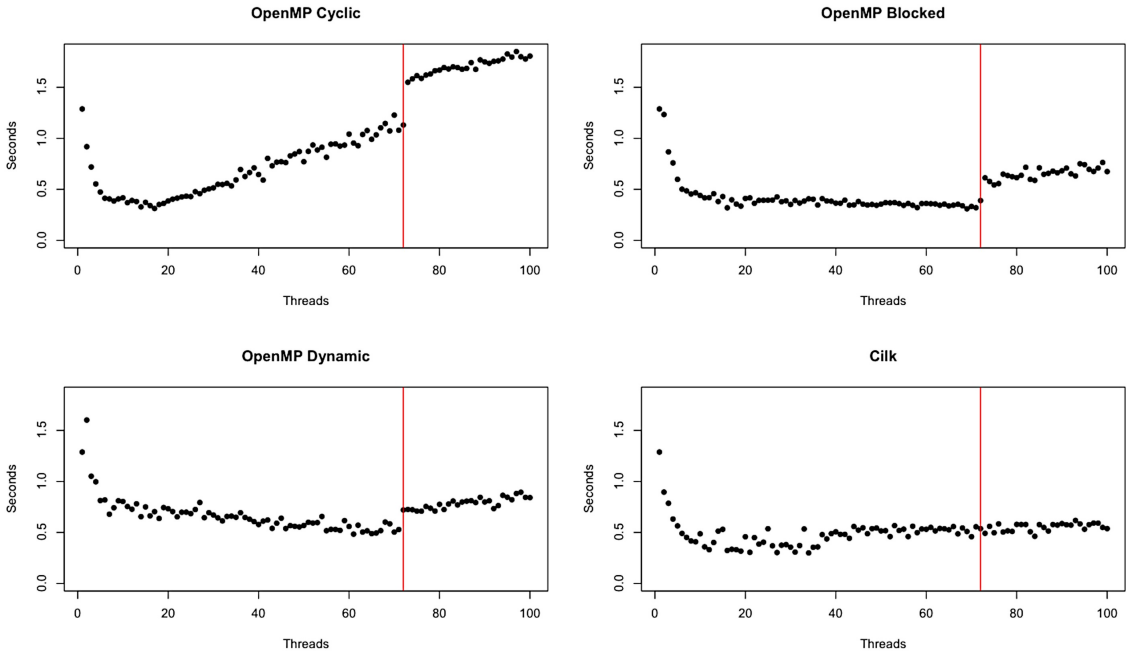
When static scheduling is used, each thread is allocated a "chunk size" number of contiguous matrix rows. In OpenMP Cyclic, each thread is allocated all rows adjacent to one row (3) and in OpenMP Blocked each thread is allocated 10 contiguous matrix rows.

When dynamic scheduling is used, each thread is allocated an equal number of matrix rows. When a thread finishes computing its rows, it will take rows from

threads that have not yet finished, decreasing runtime. Cilk uses dynamic scheduling by default.

Results

The results after running gauss on a 2,000x2,000 matrix on node2x18a.csug.rochester.edu (72 processors), varying t between 1 and 100, are shown below. The red line indicates where the number of threads is equal to the number of processors.



Algorithm	Speedup	Efficiency
OpenMP Cyclic	1.139603	0.01582782
OpenMP Blocked	3.304924	0.04590173
OpenMP Dynamic	1.788172	0.02483572
Cilk	2.397222	0.03329475

Conclusions

Since all results were taken on the same machine, there is a direct relationship between speedup and efficiency.

The most efficient algorithm was OpenMP Blocked. This is because allocating a contiguous chunk of matrix rows of appropriate size benefited by decreasing cache misses and therefore decreasing runtime. Different performance for OpenMP Blocked could be seen by varying "chunk size".

The least efficient algorithm was OpenMP Cyclic. Despite also being an OpenMP algorithm with static scheduling, significantly less speedup is seen than with OpenMP Blocked. This is because "chunk size" for OpenMP Cyclic was too small and suffered from increasing cache misses and therefore increasing runtime.

The middle two algorithms in terms of efficiency were OpenMP Dynamic and Cilk both of use dynamic scheduling. This is a simpler implementation for the programmer as "chunk size" need not be fine tuned as all threads are allocated an equal number of matrix rows. Cilk's implementation of dynamic scheduling is more efficient than Open MP's implementation of dynamic scheduling.