Parallel Scientific Computing

Sean Laguna Hannah Morgan Daniel Reid

The purpose of this project is to implement a block inverse method using parallel prefix (PP) to solve a banded lower triangular matrix. The required time is compared to the time required for solving the matrix using serial forward substitution (FS). It was found that the total time required to solve a problem once with the block inverse parallel prefix method was much greater the time to solve using forward substitution. However much of work done in the PP method is reusable, so the marginal work to solve using the PP method is less than that of FS for large matrices (n ¿ 512).

The algorithm was implemented using three different methods. Sean used pthreads and Daniel used Windows threads. Both methods use a shared memory model. Sean's code will be tested on the 32-core Computer Science Clover server since they can be compiled for Linux. Daniel's will be run on an 8-core Windows machine. The implementations will be timed and compared to the forward substitution method.

The algorithms were tested on a banded lower triangular matrix of bandwidth 2 as shown in Equation ??.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{1}$$

The equation is abbreviated as

$$Lx = f$$

Daniel's Implementation

I implemented the parallel prefix method using Windows threading since my code was already Windows specific. The time to solve the matrix was measured for the parallel prefix and the forward substitution method. The results are shown in Table WHATEVER. Note that forward substitution was done in serial. It can be seen that the PP method takes significantly longer than the FS method to solve the matrix once. However much of the work done on L can be used for multiple X vectors, making the marginal work to solve with the parallel prefix lower than the work for forward substitution for sufficiently large matrices. Table WORK BREAKDOWN shows a breakdown of of the reusable and non-reusable work. Table COMP TO FS compares the marginal time for forward substitution to that of parallel prefix.

Some optimization was done to speed up the process. The speed increases due to each of the optimizations described are shown in Table SPEEDS Initially, C++ Vectors were used to store arrays of matrices such as the Gis, Lis, etc. These arrays are all of fixed and known length, so the code was optimized by replacing the vectors with static arrays. Using static arrays did not noticeably change the runtime for a small number of processors. This is probably because the C++ Vector class allocates some number of slots in initialization. If the final length of the vector is less than that number of slots, the vector will not have to moved in memory. This could change with more processors. I used a custom and mostly unoptimized matrix class, which will like give slower runtimes.

Computing the Gis used a large fraction of the total time to set up a problem. The process was parallelized. This does not affect the marginal work of solving, but reduces the time to set up and solve a single problem by a factor of 2-3. Finally, the Matrix class was optimized by reserving array in advance of pushing the values onto it.