Parallel Implementation for Gaussian Elimination Using MPI and OpenMP

Author: Danqi Qu | College of Engineering | CMSE822 Supervisor: Sean Couch
Michigan State University

Introduction

In linear algebra, Gaussian elimination is an algorithm for solving systems of linear equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}, \Rightarrow \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1k} \\ 0 & a'_{22} & \cdots & a'_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a'_{kk} \end{bmatrix} \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_k \end{bmatrix}.$$

Row Operations

- Swapping two rows
- Multiplying a row by a nonzero number
- Adding a multiple of one row to another row

Objectives

Overall

Develop a parallel version of Gaussian elimination for machines with multiple cores and processors and test the implement performance.

Tasks

1. Use different parallel programming models: MPI and OpenMP.
2. Explore different parallelization strategies: domain decomposition and task-based.
3. Perform extensive scaling studies: weak scaling, strong scaling, and thread-to-thread speedup.

Reference

1. Victor Eijkhout, Introduction to High Performance Scientific Computing, 2nd edition
2. Victor Eijkhout, Parallel Programming in MPI and OpenMP, 1st edition, 2017
3. M, Cosnard, Parallel Gaussian elimination on an MIMD computer, Parallel Computing, 1988, p275-296
4. S. F. McGinn, Parallel Gaussian Elimination Using OpenMP and MPI, IEEE, 2002
5. Simplice Donfack, A survey of recent developments in parallel implementation of Gaussian elimination, Concurrency and Computation, 2015
6. Jeff Howe, http://cseweb.ucsd.edu/classes/fa98/cse164b/Projects/PastProjects/LU/abstract.html
7. Vardit Cohen, https://www.cs.rutgers.edu/~venugopa/parallel_summer2012/ge.html

Algorithm

Domain Decomposition

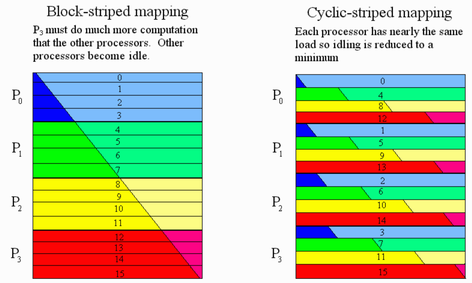
An entire row or group of rows are issued to each processor.

In blocked mapping, a processor is assigned a number of contiguous rows.

In cyclic mapping, distributing the rows to processors cyclically, which can balance the computation load for each processor.

Problem: Load inbalance due to higher numbered rows taking longer to compute than lower numbered rows.

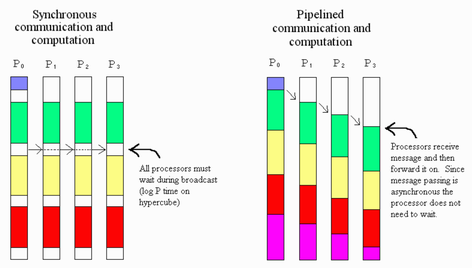
Solution: Use cyclic-striped mapping



Communication and Computation Overlap

Problem: Costly broadcast of row must be performed at each iteration.

Solution: Pipelined communication and computation.



OpenMP Implementation

```
   do pivot = 1, (n-1)
!$omp parallel do private(xmult) schedule(runtime)
     do i = (pivot+1), n
       xmult = a(i,pivot) / a(pivot,pivot)
       do j = (pivot+1), n
         a(i,j) = a(i,j) - (xmult * a(pivot,j))
       end do
       b(i) = b(i) - (xmult * b(pivot))
     end do
!$omp end parallel do
   end do
```

In the broadcast method, when a row has completed the division step, it broadcasts its newly calculate values to all processors.

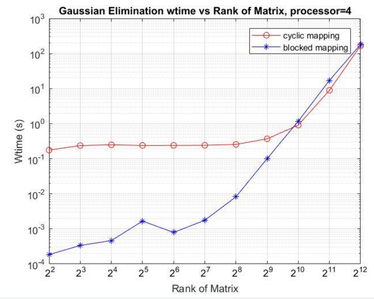
In pipelined communication, a row is sent to a processor who forwards it on to the next processor and it can immediately begin its local computations.

1. The iterations of loop is known.
2. Each iteration is independent to all others.
3. The loop contains no data dependence.

Use OpenMP loop parallelism.

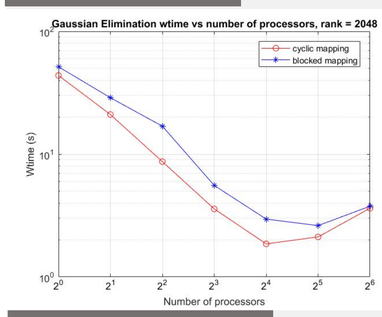
Results

Weak Scaling



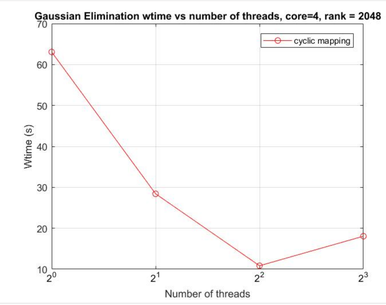
In the weak scaling, the number of processors used are fixed (=4). With the rank of matrix increasing, the total wtime increases.
The blue line represents the performance of blocked mapping method and the red one is cyclic mapping. When the rank of matrix is larger than 1024, the performance of cyclic mapping is better than that of blocked mapping.

Strong Scaling



In the strong scaling, the rank of matrix involved are fixed (=2048). With the number of processors increasing, the total wtime decreases.
At all number of processors, the performance of cyclic mapping is always better than that of blocked mapping.
When the number of processors is over 16, the total run time begins to increase. That's because of the communication time cost between processors.

OpenMP Threads



When the number of threads is below 4, with the number of threads increasing, the total walltime deceases. But when the number of threads equals to 8, the performance is worth than 4 threads.

Conclusion

1. Domain decomposition can implement the performance of parallel Gaussian elimination especially for high rank matrix.
2. Cyclic mapping can make the computation load balanced and is helpful for performance at high rank matrix.
3. With certain number of threads, OpenMP loop parallelism is beneficial to performance.