Songyuan Cui, Sowmya Yellapragada
cui20@illinois.edu, sowmyay2@illinois.edu

CS 420: Project Proposal
Term: Spring 2023

Project Title: Parallel Cholesky factorization with shared and distributed memory parallelism

Members: Songyuan Cui, Sowmya Yellapragada

Problem Statement
Symmetric positive-definite (SPD) linear systems are common in physical and mathematical problems, and efficient algorithms for solving SPD systems have been thoroughly studied. One such method is the Cholesky factorization which, when applicable, is approximately $2\times$ faster than generic LU factorization. In this project, we aim to parallelize it via share (OpenMP) and distributed (MPI) memory parallelism and show performance.

Methodology
The Cholesky factorzation can be summarized as decomposing an SPD matrix into the quadratic form of a upper / lower triangular matrix

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T. \tag{1}$$

The lower triangular matrix can be obtained by first copying the lower triangle of $\boldsymbol{A}$, and then apply the algorithm

for $k = 0 : n-1$ **do** ▷ Loop over rows
  $a_{kk} = \sqrt{a_{kk}}$
  for $i = k+1 : n-1$ **do** ▷ Scale current column
    $a_{ik} = a_{ik}/a_{kk}$
  end for
  for $j = k+1 : n-1$ **do** ▷ Subtract multiple of current column
    for $i = j : n-1$ **do**
      $a_{ij} = a_{ij} - a_{ik}a_{jk}$
    end for
  end for
end for

This method, by virtue of the SPD nature of the original matrix, has multiple advantages:

1. By symmetry, only half of the matrix needs to be stored / accessed.
2. SPD matrices have a full set of real, positive eigenvalues and orthogonal eigenvectors, so no pivoting is required for numerical stability.

For row-first Cholesky factorization, shared memory parallelism can be implemented by parallelizing the inner 'for' loops within the algorithm, as they are operations upon entire columns. Distributed memory parallism can be achieved by mapping rows of the lower triangular matrix to multiple ranks in a circular fashion.

Deliverable

1. Implement Cholesky Factorization on a randomly generated SPD matrix
2. Parallelize the algorithm using MPI for distributed memory, and OpenMP for multithreading
3. Show strong scaling on a single node with shared memory parallelism
4. Shown strong and weak scaling on multiple nodes with distributed memory parallelism
5. (Optional) Solve linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ using forward / backward substitution