Parallel Jacobi Algorithm

Steven Dong
Applied Mathematics
Overview

- Parallel Jacobi Algorithm
  - Different data distribution schemes
    - Row-wise distribution
    - Column-wise distribution
    - Cyclic shifting
    - Global reduction
  - Domain decomposition for solving Laplacian equations

- Related MPI functions for parallel Jacobi algorithm and your project.
Linear Equation Solvers

- Direct solvers
  - Gauss elimination
  - LU decomposition

- Iterative solvers
  - Basic iterative solvers
    - Jacobi
    - Gauss-Seidel
    - Successive over-relaxation
  - Krylov subspace methods
    - Generalized minimum residual (GMRES)
    - Conjugate gradient
Sequential Jacobi Algorithm

\[ Ax = b \]

\[ A = D + L + U \]

\[ x^{k+1} = D^{-1}(b - (L+U)x^k) \]

D is diagonal matrix
L is lower triangular matrix
U is upper triangular matrix
Parallel Jacobi Algorithm: Ideas

- Shared memory or distributed memory:
  - Shared-memory parallelization very straightforward
  - Consider distributed memory machine using MPI

- Questions to answer in parallelization:
  - Identify concurrency
  - Data distribution (data locality)
    - How to distribute coefficient matrix among CPUs?
    - How to distribute vector of unknowns?
    - How to distribute RHS?
  - Communication: What data needs to be communicated?

- Want to:
  - Achieve data locality
  - Minimize the number of communications
  - Overlap communications with computations
  - Load balance
Row-wise Distribution

- Assume dimension of matrix can be divided by number of CPUs
- Blocks of m rows of coefficient matrix distributed to different CPUs;
- Vector of unknowns and RHS distributed similarly

\[ A \times x = b \]

- \( A \): \( n \times n \)
- \( m = n/P \)
- \( P \) is number of CPUs
 Already have all columns of matrix A on each CPU;  
Only part of vector x is available on a CPU; Cannot carry out matrix vector multiplication directly;  
Need to communicate the vector x in the computations.

\[
x^{k+1} = D^{-1}(b - (L + U)x^k)
\]
How to Communicate Vector X?

- Gather partial vector \( x \) on each CPU to form the whole vector; Then matrix-vector multiplication on different CPUs proceed independently. (textbook)

- Need MPI_Allgather() function call;
- Simple to implement, but
  - A lot of communications
  - Does not scale well for a large number of processors.
How to Communicate X?

- Another method: Cyclic shift
  - Shift partial vector $x$ upward at each step;
  - Do partial matrix-vector multiplication on each CPU at each step;
  - After $P$ steps ($P$ is the number of CPUs), the overall matrix-vector multiplication is complete.

- Each CPU needs only to communicate with neighboring CPUs
  - Provides opportunities to overlap communication with computations

- Detailed illustration …
\[
\begin{align*}
\text{Cpu 0:} & & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \\
\text{Cpu 1:} & & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \\
\text{Cpu 2:} & & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 \\
\text{Cpu 3:} & & a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4
\end{align*}
\]
Overlap Communications with Computations

- **Communications:**
  - Each CPU needs to send its own partial vector $x$ to upper neighboring CPU;
  - Each CPU needs to receive data from lower neighboring CPU

- **Overlap communications with computations:** Each CPU does the following:
  - Post non-blocking requests to send data to upper neighbor to receive data from lower neighbor; This returns immediately
  - Do partial computation with data currently available;
  - Check non-blocking communication status; wait if necessary;
  - Repeat above steps
Stopping Criterion

\[ \| x^{k+1} - x^k \| < \varepsilon \| b \| \]

\[ \| \vec{A} - \vec{B} \| = \sqrt{\sum_i (A_i - B_i)^2} \]

- Computing norm requires information of the whole vector;
- Need a global reduction (SUM) to compute the norm using MPI_Allreduce or MPI_Reduce.
Column-wise Distribution

Blocks of m columns of coefficient matrix A are distributed to different CPUs;

Blocks of m rows of vector x and b are distributed to different CPUs;

\[ x^{k+1} = D^{-1}(b - (L+U)x^k) \]
Data to be Communicated

- Already have coefficient matrix data of m columns, and a block of m rows of vector x;
- So a partial $A \times x$ can be computed on each CPU independently.
- Need communication to get whole $A \times x$;
How to Communicate

- After getting partial $A^*x$, can do global reduction (SUM) using MPI_Allreduce to get the whole $A^*x$. So a new vector $x$ can be calculated.

- Another method: Cyclic shift
  - Shift coefficient matrix left-ward and vector of unknowns upward at each step;
  - Do a partial matrix-vector multiplication, and subtract it from the RHS;
  - After $P$ steps ($P$ is number of CPUs), matrix-vector multiplication is completed and subtracted from RHS; Can compute new vector $x$.

- Detailed illustration …
(1) \[ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \]

b_1 - a_{11}x_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4
b_2 - a_{21}x_1 - a_{22}x_2 - a_{23}x_3 - a_{24}x_4
b_3 - a_{31}x_1 - a_{32}x_2 - a_{33}x_3 - a_{34}x_4
b_4 - a_{41}x_1 - a_{42}x_2 - a_{43}x_3 - a_{44}x_4

(2) \[ \begin{bmatrix} a_{12} & a_{13} & a_{14} & a_{11} \\ a_{22} & a_{23} & a_{24} & a_{21} \\ a_{32} & a_{33} & a_{34} & a_{31} \\ a_{42} & a_{43} & a_{44} & a_{41} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \]

b_1 - a_{11}x_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4
b_2 - a_{21}x_1 - a_{22}x_2 - a_{23}x_3 - a_{24}x_4
b_3 - a_{31}x_1 - a_{32}x_2 - a_{33}x_3 - a_{34}x_4
b_4 - a_{41}x_1 - a_{42}x_2 - a_{43}x_3 - a_{44}x_4

(3) \[ \begin{bmatrix} a_{13} & a_{14} & a_{11} & a_{12} \\ a_{23} & a_{24} & a_{21} & a_{22} \\ a_{33} & a_{34} & a_{31} & a_{32} \\ a_{43} & a_{44} & a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \]

b_1 - a_{11}x_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4
b_2 - a_{21}x_1 - a_{22}x_2 - a_{23}x_3 - a_{24}x_4
b_3 - a_{31}x_1 - a_{32}x_2 - a_{33}x_3 - a_{34}x_4
b_4 - a_{41}x_1 - a_{42}x_2 - a_{43}x_3 - a_{44}x_4

(4) \[ \begin{bmatrix} a_{14} & a_{11} & a_{12} & a_{13} \\ a_{24} & a_{21} & a_{22} & a_{23} \\ a_{34} & a_{31} & a_{32} & a_{33} \\ a_{44} & a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} x_4 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \]

b_1 - a_{11}x_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4
b_2 - a_{21}x_1 - a_{22}x_2 - a_{23}x_3 - a_{24}x_4
b_3 - a_{31}x_1 - a_{32}x_2 - a_{33}x_3 - a_{34}x_4
b_4 - a_{41}x_1 - a_{42}x_2 - a_{43}x_3 - a_{44}x_4
Solving Diffusion Equation

\[ \nabla^2 f + q = 0 \]

\[ f_{ij} = \frac{1}{4} (f_{i-1,j} + f_{i+1,j} + f_{ij-1} + f_{ij+1} + \Delta x^2 q_{ij}) \]

- How do we solve it in parallel in practice?
- Need to do domain decomposition.
Domain Decomposition

- Column-wise decomposition
- Boundary points depend on data from neighboring CPU
  - During each iteration, need send own boundary data to neighbors, and receive boundary data from neighboring CPUs.
- Interior points depend only on data residing on the same CPU (local data).
Overlap Communication with Computations

- Compute boundary points and interior points at different stages;

Specifically:

- At the beginning of an iteration, post non-blocking send to and receive from requests for communicating boundary data with neighboring CPUs;
- Update values on interior points;
- Check communication status (should complete by this point), wait if necessary;
- Boundary data received, update boundary points;
- Begin next iteration, repeat above steps.
Other Domain Decompositions

1D decomposition

2D decomposition
Related MPI Functions for Parallel Jacobi Algorithm

- MPI_Allgatherv()
- MPI_Isend()
- MPI_Irecv()
- MPI_Reduce()
- MPI_Allreduce()
MPI Programming Related to Your Project

- Parallel Jacobi Algorithm
- Compiling MPI programs
- Running MPI programs
- Machines: www.cascv.brown.edu