PARALLEL TRIANGULAR SOLVES: A TALE OF TWO ALGORITHMS

1. SHELBY LOCKHART AND SAMAH KARIM

PROBLEM

Sparse triangular solves are difficult to parallelize due to their irregular storage structure and the sequential nature of backward and forward substitution algorithms, as seen below.

Sequential Forward Substitution
1: for \( j = 0 \ldots n \) do
2: \( x_j = b_j / l_{jj} \)
3: end for

Approximate Solve

When used in preconditioning, finding an exact solution to the triangular solve becomes less important. Hence an approximate iterative solver can be used to get a “good-enough” solution at every step of the preconditioned Krylov solver. One approach is the Jacobi method. 1

Starting from an initial guess \( x^{(0)} \), compute next iterate as follows:

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

where \( D \) is the matrix consisting of the diagonal of \( L \).

Parallel Direct Solve

Attempts to parallelize triangular solvers come with a great communication cost, due to every process needing access to all components of the solution vector. This translates to a broadcast at every iteration as seen in the Row Fan-Out algorithm. 2

Parallel 1-D Row Fan-Out Forward Substitution
1: for \( j = 1 \ldots n \) do
2: if \( j \in \text{myrows} \) then
3: \( x_j = b_j / l_{jj} \)
4: end if
5: Broadcast \( x_j \)
6: for \( i \in \text{myrows}, i > j \) do
7: \( b_i = b_i - l_{ij}x_j \)
8: end for
9: end for

With sparse matrices, using a primitive block-row partitioning of matrices will result in less computation, but the communication overhead will be the same as in the dense case.

Parallel Approximate Solve

Jacobi method parallelizes well:

- All components of the current iterate \( x^{(k+1)} \) depend only on components of the previous iterate \( x^{(k)} \)
- They can be updated simultaneously

All basic operations can be done in parallel:

- Row scaling operation with depth = 1
- SpMV with depth = \( \log(n) \)
- saxpy with depth = 1

Strong Scaling Experiments

Experimental Parameters:
- UF Trefethen2000 \( L \) component from LU; \( n = 2000 \)
- UF nd3k lower triangular part; \( n = 9000 \)
- Each approximate solve was 5 Jacobi iterations

Parallel Cost Models

\[ T_p = (\alpha + \beta)(n - 1)b_k + \gamma \left( \frac{n^2 + 2nρ - 2n}{2p} \right) \]  

\[ T_p = \alpha a_k + \beta n + \gamma \left( \frac{n^2 + 6n}{2p} \right) \]  

\[ T_p = \alpha a_k + \beta n + \gamma \left( \frac{3n + mnz}{p} \right) \]

Machine Specs

2x Broadwell-EP 12-core Xeon
256 GiB of 2133 DDR4 RAM

References