Parallelizing Floyd's Algorithm

A distributed memory approach

Javier Jorge Cano

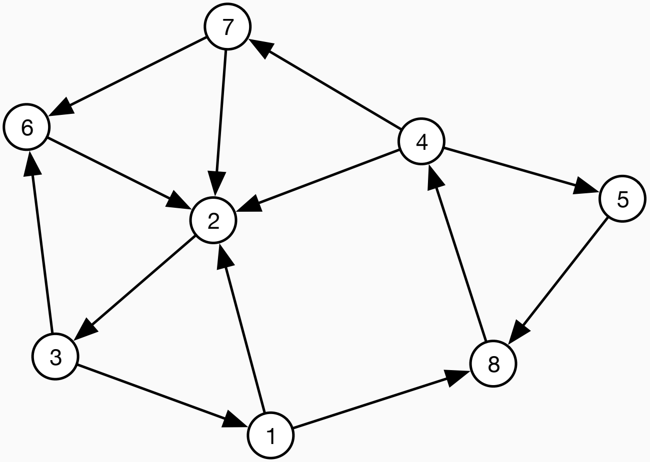
January 24, 2018

Master's Degree in Parallel and Distributed Computing

Table of contents

Problem statement

All Pairs Shortest Path



Problem

Find all pair-wise shortest path during one execution

Graph representation

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} \end{pmatrix}$$

Sequential algorithm

Sequential algorithm

Algorithm 1 Floyd-Warshall Algorithm

procedure FLOYD-WARSHALL[1]
  Let $dist$ be a $|V| \times |V|$ matrix of min. distances initialized to ∞
  for each vertex v **do**
    $dist[v][v] \leftarrow 0$
  for each edge (u, v) **do**
    $dist[u][v] \leftarrow w(u, v)$ ▷ the weight of the edge (u, v)
  for k **from** 1 **to** $|V|$ **do**
    for i **from** 1 **to** $|V|$ **do**
      for j **from** 1 **to** $|V|$ **do**
        if $dist[i][j] > dist[i][k] + dist[k][j]$ **then**
          $dist[i][j] \leftarrow dist[i][k] + dist[k][j]$

[1] home.iitk.ac.in/s̃hubhoj/

Example

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} \end{pmatrix}$$

Example

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} \end{pmatrix}$$

Parallel approach

Parallel approach

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8} \end{pmatrix}$$

Parallel approach

$$A = \left(\begin{array}{cc|cc|cc|cc}
a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & a_{2,8} \\
a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & a_{3,8} \\
a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} \\
\hline
a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} \\
a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} \\
a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} \\
a_{8,1} & a_{8,2} & a_{8,3} & a_{8,4} & a_{8,5} & a_{8,6} & a_{8,7} & a_{8,8}
\end{array}\right)$$

Scattering the matrix

$$
\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{bmatrix}
\begin{bmatrix} a_{1,3} & a_{1,4} \\ a_{2,3} & a_{2,4} \\ a_{3,3} & a_{3,4} \\ a_{4,3} & a_{4,4} \end{bmatrix}
\begin{bmatrix} a_{1,5} & a_{1,6} \\ a_{2,5} & a_{2,6} \\ a_{3,5} & a_{3,6} \\ a_{4,5} & a_{4,6} \end{bmatrix}
\begin{bmatrix} a_{1,7} & a_{1,8} \\ a_{2,7} & a_{2,8} \\ a_{3,7} & a_{3,8} \\ a_{4,7} & a_{4,8} \end{bmatrix}
$$
$$
\begin{bmatrix} a_{5,1} & a_{5,2} \\ a_{6,1} & a_{6,2} \\ a_{7,1} & a_{7,2} \\ a_{8,1} & a_{8,2} \end{bmatrix}
\begin{bmatrix} a_{5,3} & a_{5,4} \\ a_{6,3} & a_{6,4} \\ a_{7,3} & a_{7,4} \\ a_{8,3} & a_{8,4} \end{bmatrix}
\begin{bmatrix} a_{5,5} & a_{5,6} \\ a_{6,5} & a_{6,6} \\ a_{7,5} & a_{7,6} \\ a_{8,5} & a_{8,6} \end{bmatrix}
\begin{bmatrix} a_{5,7} & a_{5,8} \\ a_{6,7} & a_{6,8} \\ a_{7,7} & a_{7,8} \\ a_{8,7} & a_{8,8} \end{bmatrix}
$$

Motivation for our proposal

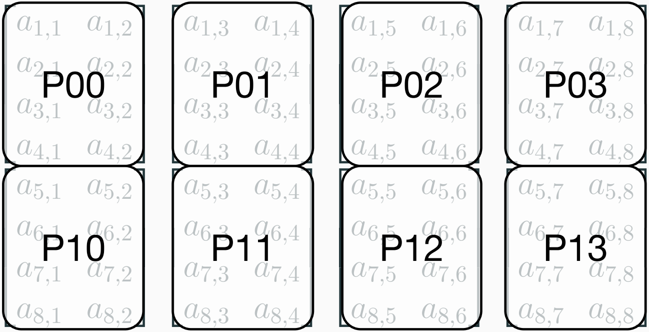


Figure 1: Final distribution after scattering the matrix

Scattering the matrix

Algorithm 2 Floyd-Warshall Algorithm

procedure FLOYD-WARSHALL
    for each block $B_{i,j}$ **do**
        if $P_{0,0}$ **then**
            sends $B_{i,j}$ to $P_{i,j}$ $\triangleright$ $P_{0,0}$ stores its block
        else
            receives $B_{i,j}$
    Compute the solution
    for each block $B_{i,j}$ **do**
        if $P_{0,0}$ **then**
            receives $B_{i,j}$ from $P_{i,j}$ $\triangleright$ $P_{0,0}$ has its block
        else
            sends $B_{i,j}$

Motivation for our proposal

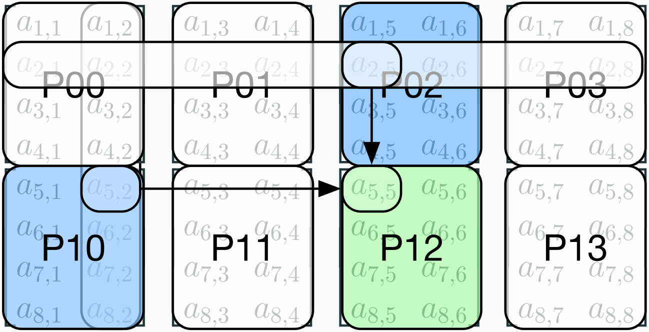


Figure 2: Example of iteration $k = 2$, where $P_{1,2}$ needs the row and the column that $P_{0,2}$ and $P_{1,0}$ have locally.

Motivation for our proposal

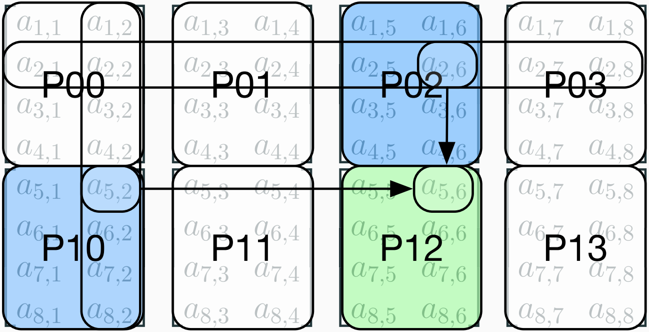


Figure 3: $P_{1,2}$ has all the information to compute its block.

Parallel approach

Algorithm 3 Floyd-Warshall Algorithm

procedure FLOYD-WARSHALL
    Sharing the matrix
    for k **from** 1 **to** $|V|$ **do**
        if $k \in rows(P_{i,j})$ **then**
            Broadcats $B_{k,chunk}$ to my col. communicator
        if $k \in cols(P_{i,j})$ **then**
            Broadcats $B_{chunk,k}$ to my row communicator
        for i **from** 1 **to** $chunk$ **do**
            for j **from** 1 **to** $chunk$ **do**
                if $dist[i][j] > dist[i][k] + dist[k][j]$ **then**
                    $dist[i][j] \leftarrow dist[i][k] + dist[k][j]$
    Gathering the matrix

Experiments and results

Experiments and results

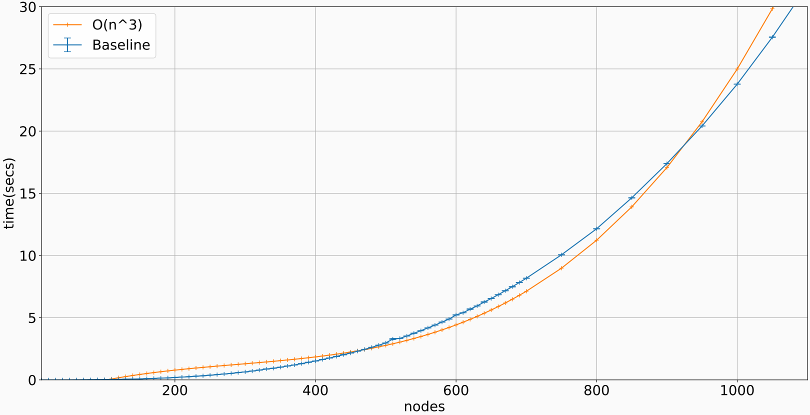


Figure 4: Sequential implementation.

Experiments and results

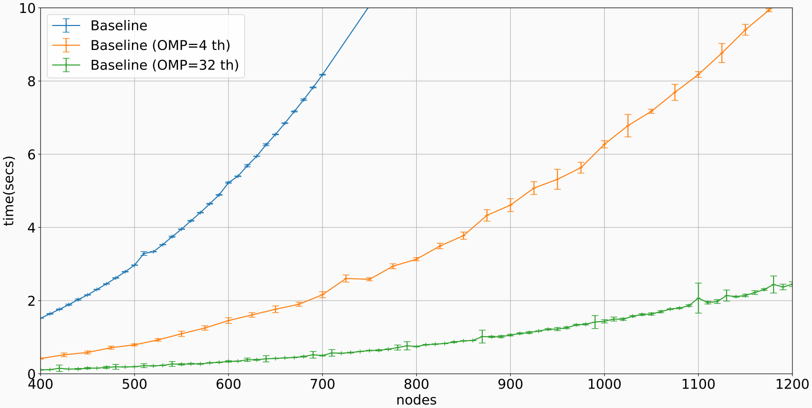


Figure 5: Straightforward parallelization with OpenMP.

Experiments and results

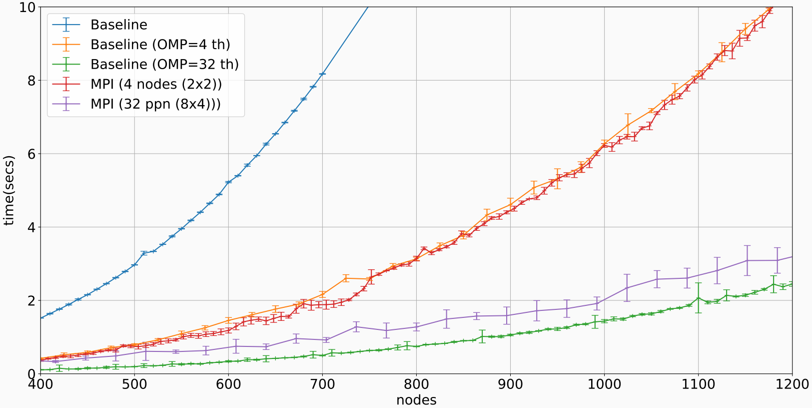


Figure 6: Parallelization with MPI, using the same number as procs.(MPI) and threads (OMP)

Experiments and results

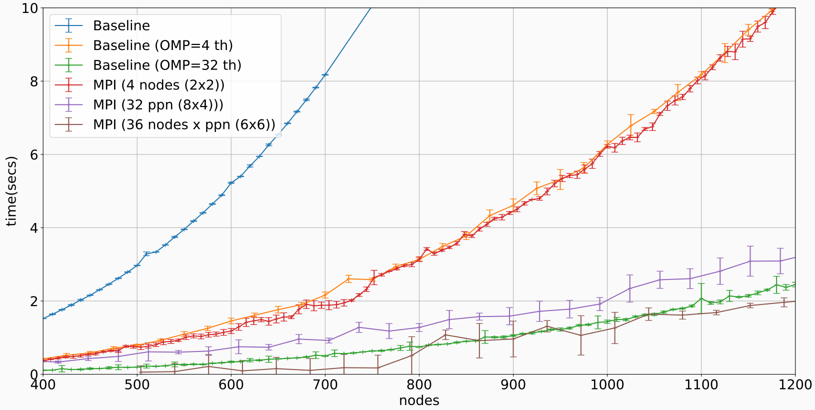


Figure 7: Parallelization with MPI: Maximum mapping.

Experiments and results

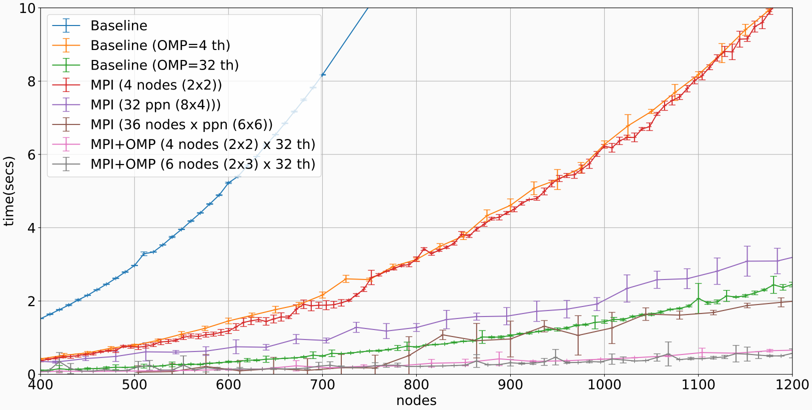


Figure 8: Mix Parallelization: MPI + OMP.

Experiments and results

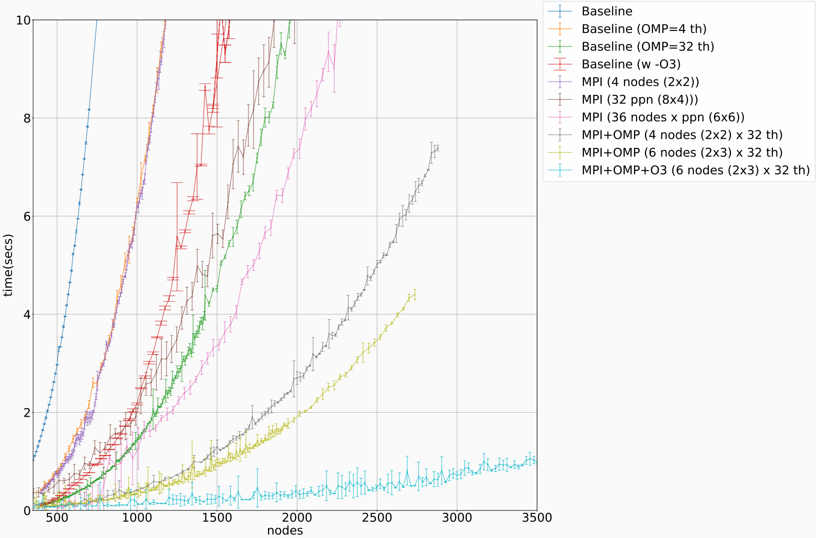


Figure 9: Final comparative.

Experiments and results

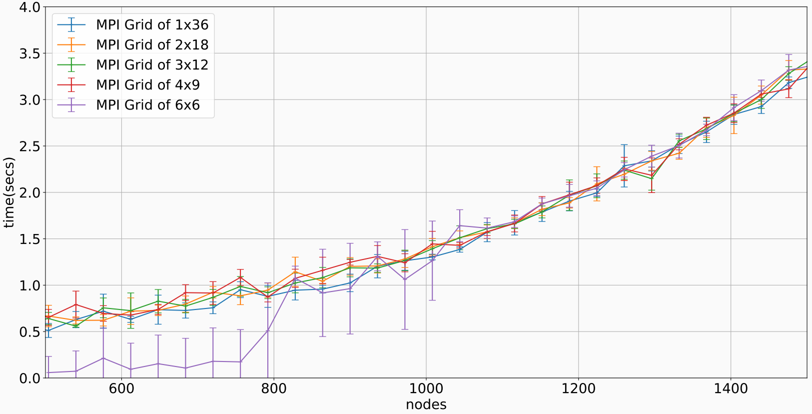


Figure 10: Comparative of different 2D Grid's distributions.

Conclusions

Conclusions

- An incremental solution has been provided to parallelize Floyd's algorithm, using a Grid distribution with MPI.
- As a Matrix-wise problem, plain OpenMP worked well.
- In this problem, the overhead of creating threads is equivalent to the communication's overhead.
- The best performance was achieved with a mixture parallelization model.
- The mistery of the `-O3` flag remains unknown.

Questions?