CS 5220
Project 3 - Floyd-Warshall Algorithm

Eric Gao (emg222)
Elliot Cartee (evc34)
Sheroze Sheriffdeen(mss385)

November 10, 2015

1 Introduction

The Floyd-Warshall algorithm computes the pair-wise shortest path lengths given a graph with a metric. The computational pattern of this algorithm is very much akin to matrix multiplication. If l_{ij}^s represents the the length of the shortest path from node i to j in at most 2^s steps, [1] then

$$l_{ij}^{s+1} = \min_k \{l_{ik}^s + l_{kj}^s\} \tag{1}$$

2 Design Decisions

2.1 Parallel Tuning

2.2 MPI

In the MPI implementation, each process handles a certain region of the graph. To prevent a master process orchestrating the distance computation, we ideally want each process to only wait for information from the relevant part of the graph. To that end, we take the adjacency matrix on which the Floyd-Warshall algorithm is run and partition the graph by chunks of columns.

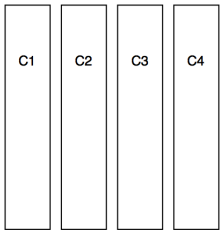

Figure 1: Initial partition of the graph where each C_i is a sequence of columns

Now each sequence of columns owned by a processor can be decomposed into square blocks. To compute the next iteration of the Floyd-Warshall algorithm for a single block, say block number i in processor 2, we need the i^{th} block from all other processors.

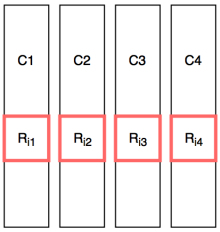

Figure 2: For block R_{i1} we need the i^{th} block from all other processors

Therefore, we do an `MPI_Allgather` operation which gathers the i^{th} block from all the processors and recreates the i^{th} row chunk in all processors. Now, we can update block R_{ij} for all j processors.

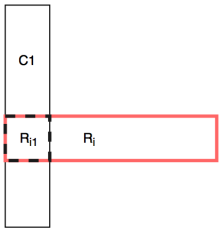

Figure 3: Updating the i^{th} square block by processor 1 using row chunk i

The `MPI_Allgather` is then repeated until all square blocks have completed a step in the Floyd-Warshall algorithm. Each processor individually checks whether an update was made to their sequence of columns. To complete the iteration, we perform a `MPI_Allreduce` operation to check whether any update was made across all processors. If an update was made, we continue the iteration. Otherwise, all processors terminate and the solution is reached.

3 Analysis

3.1 Original Implementation

3.1.1 Profiling

3.1.2 Scaling Study

3.1.3 Strong Scaling Study

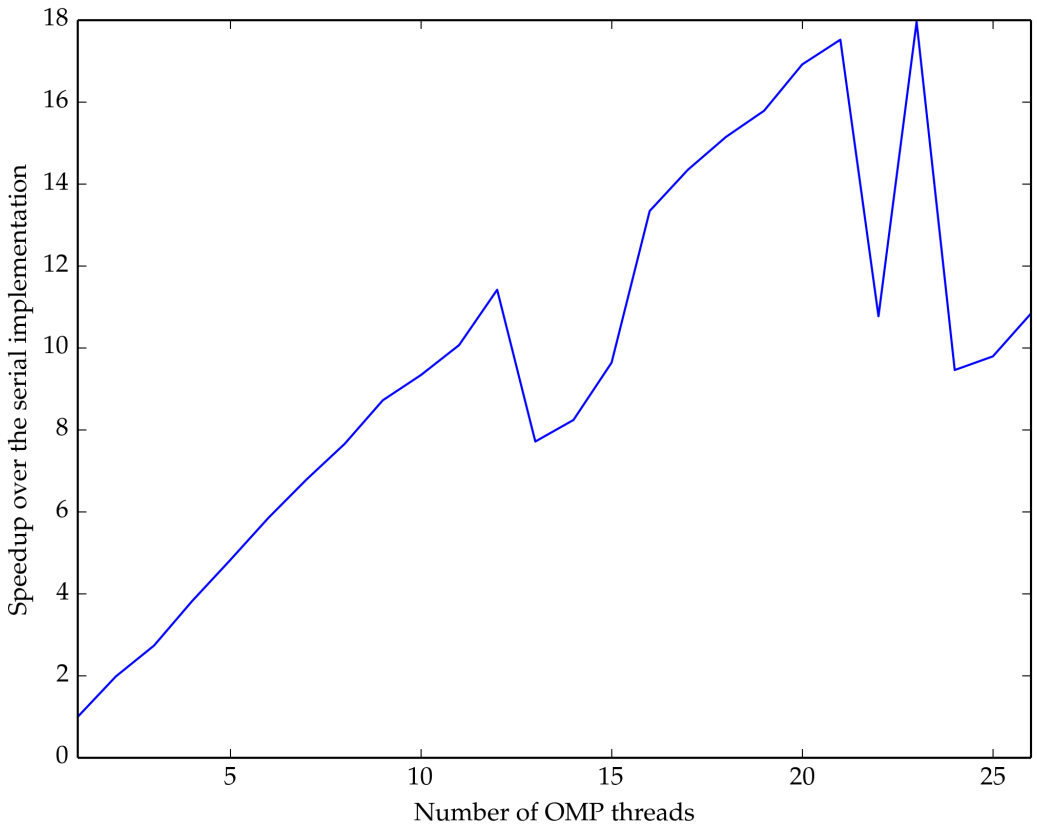

Figure 4: Strong scaling study of the original solution

3.1.4 Weak Scaling Study

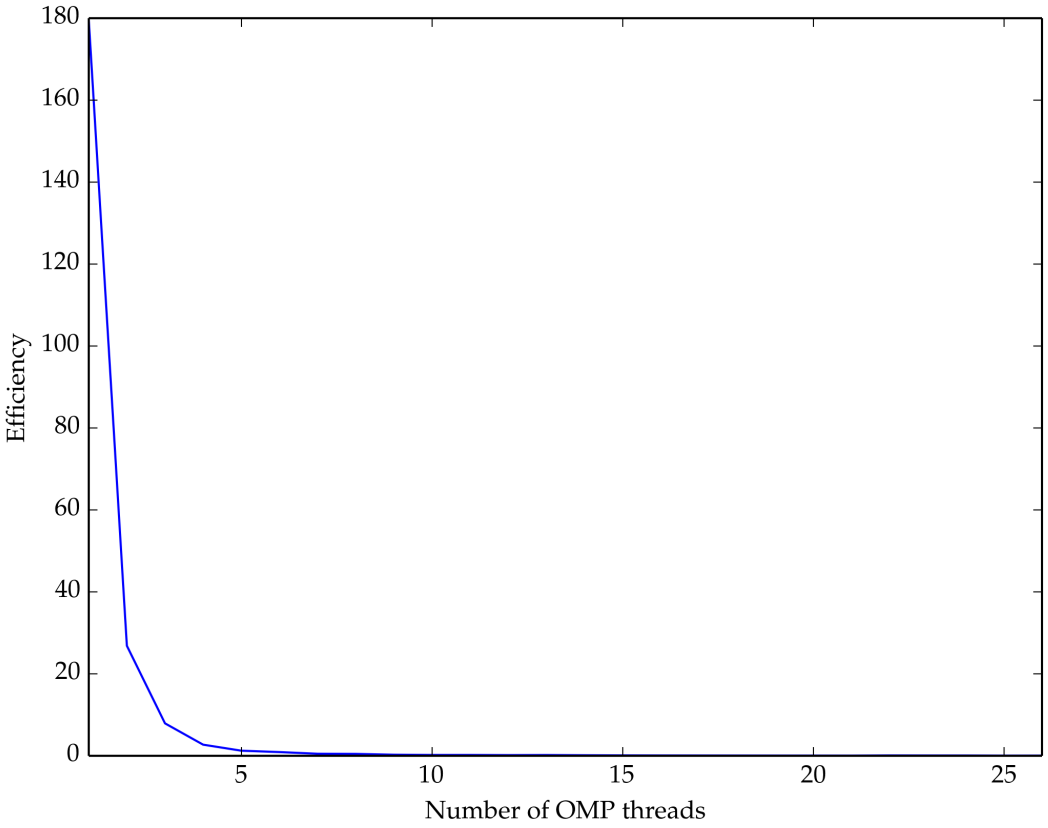

Figure 5: Weak scaling study of the original solution

3.2 Tuned Parallel Implementation

3.2.1 Profiling

3.2.2 Scaling Study

3.2.3 Strong Scaling Study

3.2.4 Weak Scaling Study

3.3 MPI Implementation

3.3.1 Profiling

3.3.2 Scaling Study

3.3.3 Strong Scaling Study

3.3.4 Weak Scaling Study

References

[1] Bindel, D. All-Pairs Shortest Paths. Retrieved November 10, 2015, from `https://github.com/sheroze1123/path/blob/master/main.pdf`