Parallelization of the Floyd-Warshall algorithm

David Bertoldi – 735213
email: d.bertoldi@campus.unimib.it

Department of Informatics, Systems and Communication

University of Milano-Bicocca

***Abstract*—The well known Floyd-Warshall (FW) algorithm solves the all-pairs shortest path problem on directed graphs. In this work we parallelize the FW using three different programming environments, namely MPI, OpenMP and CUDA. We experimented with multiple data sizes, in order to gain insight on the execution behavior of the parallelized algorithms on modern multicore and distributed platforms, and on the programmability of the aforementioned environments. We were able to significantly accelerate FW performance utilizing the full capacity provided by the architectures used.**

I. Introduction and Background

The FW is a classic dynamic programming algorithm that solves the *all-pairs shortest path (APSP)* problem on directed weighted graphs $G(V, E, w)$, where $V = \{1, \dots, n\}$ is a set of nodes, $E \subseteq V \times V$ are the edges and w is a weight function $E \to \mathbb{R}$ that expresses the cost of traversing two nodes. The number of nodes is denoted by n and the number of edges by m .

The output of the algorithm is typically in matrix form: the entry in the ith row and jth column is the weight of the shortest path between nodes i and j. FW runs in $\Theta(|V|^3)$ time and for this reason is a good choiche when working with dense graph: even though there may be up to $\Omega(|E|^2)$ edges, the computational time is independent from the number of edges.

The FW algorithm is shown in Alg. 1

Algorithm 1: The Floyd-Warshall (FW) algorithm

for $(u, v) \in E$ **do**
    $M_{u,v} \leftarrow w(u, v)$
end
for $v = 1 \to n$ **do**
    $M_{v,v} \leftarrow 0$
end
for $k = 1 \to n$ **do**
    for $i = 1 \to n$ **do**
        for $j = 1 \to n$ **do**
            if $M_{i,j} > M_{i,k} + M_{k,j}$ **then**
                $M_{i,j} \leftarrow M_{i,k} + M_{k,j}$
            end
        end
    end
end

It is easy to notice that that the nested i and j for-loops are totally independent and therefore parallelizable.

II. Methodology