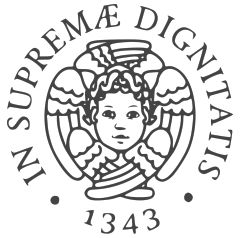
Design and implementation of parallel breadth-first search

Project Report

Academic year 2020-2021
Submitted for the examination of Parallel and distributed systems: paradigms and models

By

Giuseppe Grieco
g.grieco6@studenti.unipi.it



Department of Computer Science
University of Pisa, Italy

July, 2021

Contents

1 Introduction

1.1 Problem statement

The bread-first search is an algorithm visiting the graph in amplitude. It starts from a node, often called the root or source node, and continues visiting all its descendants level by level, whereas the i-th level will contain all the nodes at a distance i from the root. The assumption underlying all the work is that the input to the problem is a direct and acyclic graph. In addition, for the sake of clarity, the notation used is summarized below:

Let $\mathcal{G} = (V, E)$, $|V| = n$ is the number of node and $|E| = m$ is the number of edges. For all the node $v \in V$:

- $\mathcal{N}(v) = \{u : (u, v) \in E\}$ is the neighborhood of v;
- $k_{in}(v) = |\{e : e = (u, v) \in E\}|$ is the in-degree of v;
- $k_{out}(v) = |\mathcal{N}(v)|$ is the out-degree of v;
- $k(v) = k_{in}(v) + k_{out}(v)$ is the degree of v.
- $d(u, v)$ is the distance from u to v, i.e. the minimum path from u to v.

Using the node-focus notation above, it is possible to define general properties for the graph:

- $\bar{k}$ is the average degree;
- $\bar{d}$ is the average distance;
- d_{max} is the diameter.

1.2 Proposed solution

Before entering in the algorithmic details. let's first introduce the data structure used. There are many ways to represent a graph among these the main ones are adjacency list and adjacency matrix. The choices among them is mainly a matter of the usage of the adjacency information and the expected nature of the graph. As for every node v of the graph, induced by the root, it will be necessary to go through each node $u \in \mathcal{N}(v)$, the adjacency list is way more efficient since for each node v the listing of $\mathcal{N}(v)$ takes a time proportional to $k_{out}(v)$, which is thus optimal. Moreover, if the expected input of the algorithm are "real" graphs than since they are very sparse, the representation as a adjacency list is way more efficient in terms of space complexity.

1.2.1 Sequential version

The sequential version

1.2.2 Parallel version

2 Analysis and measurements

3 Conclusion