Parallel breadth-first search

Alexander Gallauner
Matr.: 1026090
Curriculum: 534
`e1026090@student.tuwien.ac.at`

May 11, 2016

Abstract

Some years ago, to have a continuous progress of the performance of processors, it was necessary to stop trying to increase the clock rate of CPUs, and to concentrate on the development of multicore processors. But with this development another problem was raising up. The programs or rather the algorithmic problems have to be changed to get a speed up with multicore processors. It was the job of the developers to split the work from one core to many cores and to coordinate the communication between these cores. In this scientific paper I will concentrate on algorithms for searching tree or graph data structures and how to parallelize them. To make it specific, I will keep my mind on the breadth-first search, because it is easier to parallelize than the depth-first search. First I will show a sequential algorithm to solve this problem. After that my focus is on a parallel realisation of this sequential algorithm. In this chapter I will demonstrate some improvements to the parallel algorithm to reach a better speed up. To get a feeling how good the solution is, I will populate my solution into a benchmark for supercomputers, the graph500 project. Graph500 establishes a large-scale benchmark for data-intensive supercomputer applications. There exists already a official solution of the benchmark, so I can compare the performance of Jupiter, that is the name of the distributed system of the Vienna University of Technology, one time with my own implemented benchmark, but observing the rules of the graph500 project, and another time with the official solution of the graph500 project. Another thing is that I compare Jupiter with other supercomputers, which are listed in the graph500 ranking.