Java Assignment

Jan van der Lugt

1 Introduction

The report discusses my implementation of a distributed Rubik's cube solver based on the Ibis framework.

Section 2 will provide a design overview of my implementation, section 3 will go through the problems that were identified during the conversation from sequential to distributed code and how they were solved and section 4 will discuss performance results.

2 Design overview

My design revolves around a single double-ended queue (deque), which is maintained by the master. The master works on the front of the deque, where it removes a cube and places its children back on the front of the deque. This will cause the cubes with the most work to always be at the end of the deque. In the absence of worker, the master performs the work quite similar to the sequential implementation, with two small differences:

- The master doesn't use recursion, but use a work deque, which incurs a little more overhead, since adding and removing from a deque is more expensive than a function call. In the case of recursion, however, it is very difficult to steal work from the master.
- The children are processed in reverse order, since cubes added to the deque are processed in LIFO order. In microbenchmarks this did not make any difference, which is to be expected because all cubes are explored up to a certain bound anyway.

I chose to let the workers steal cubes instead of have the master push them to the workers, since this scheme has the least bookkeeping. Workers always steal from the end of the deque, since these cubes have the most work left to do.

3 Problems during implementation