Report Assignment 2

Verardo Thomas

18/2/2022

Introduction

In this report, there is written how a KD-Tree is implemanted and how it can be builded in serial or in a parallel way. A KD-Tree is a data structure presented originally by Friedman, Bentley and Finkel in 1977 to represent a set of k-dimensional data in order to make them efficiently searchable. A KD-Tree is a tree whose basic concept is to compare against 1 dimension at a time per level cycling cleverly through the dimensions so that to keep the tree balanced. At each iteration i, it bisects the data along the chosen dimension d, creating a "left-" and "right-" sub-trees for which the 2 conditions $\forall x \in sub-tree, x_i < p_i$ and $\forall x \in sub-tree, x_i > p_i$ hold respectively, where $p \in D$ is a point in the data set and the sub-script i indicates the component of the $i-th$ dimension.

To build the tree, two assumptions were made:

- The dataset, and hence the related KD-Tree, it's immutable;
- The data points are homogeneously distribuited in all the k dimension, i.e. the data are taken pseudo-randomly from the function *rand()* of the *random* library of C++.

In the serial part, the algorithm is costructed as a "normal" program, without keeping in consideration how the processor will works. In the parallel part, it will be consider how the processor can distribuite the amout of work among processes or threads. To implement this part, it was used two library for parallel programming, that are **OpenMP** and **Open MPI**.

Algorithm

since you can assume that your data are homogeneously distributed in every dimension, a simple and good choice is to pick the median element along each dimension.

Implementation

C++11 The implemented KD-Tree is written to work with the number of k-dimension > 0 ($N\_DIM > 0$).

#Performance model and scaling

Discussion

aaa