High Performance Computing for Data Science Project

Prof. Fiore Sandro Luigi

Guidolin Davide
davide.guidolin@studenti.unitn.it

Zanolli Giacomo
giacomo.zanolli@studenti.unitn.it

***Abstract*—This is the report for the High Performance Computing for Data Science course project.**

The goal of the project was to implement a parallel solution for the *closest pair of points* problem and evaluate it on the HPC@UniTrento cluster.

In the following sections we will present the problem and we will describe some serial solutions for it. Then we will present how we implemented a parallel solution and finally we will present and discuss the evaluation of the parallel solution on the HPC cluster.

I. Introduction

The *parallel closest pair of points* problem consists in finding the smaller distance between two points in the plane. We will focus on the 2-dimensional problem, however, solutions for the d dimensional space exist.

The naive solution to this problem would be to take each point in the plane and calculate the distance between that point and all the other points. The time complexity for this solution is $O(N^2)$ where N is the number of points in the space. Better solutions have been proposed, in particular in 1976 Rabin proposed a randomized algorithm with an expected run time of $O(N)$ and in 1979 Fortune and Hopcroft proposed a deterministic $O(NloglogN)$ solution [1] assuming that the floor operation takes constant time.

In this project we will explore a divide and conquer approach with $O(N(logN)^2)$ time complexity. We chose it because divide and conquer is highly parallelizable so it will be easier to exploit the computational power of the HPC cluster.

II. Problem analysis

Serial implementation

We will now focus on the analysis of the divide and conquer algorithm and its serial implementation.

A. Divide

The first step of the algorithm is the divide operation, i.e. split the original problem in sub problems.

The original set of points will be split in two subsets and each subset will be split in two subsets and so on. This process will be repeated until we have three or less points in each subset.

B. Find the closest pair

To find the closest pair in each subset we only have to perform at most three comparisons and select the pair of points with the smallest distance in the subset.

C. Merge

The merge operation is a crucial part in the algorithm. The first thing to do is to take two adjacent subsets and compare the distances found in each subset and take the pair of points with smaller distance. Then we need to look at the boundary between the two subsets and check the distances of the pairs where one point belongs to one subset and one point belongs to the other subset. To check these points we first select the strip of points with x distance smaller than the current smaller distance, then we sort the selected points by the y coordinate. Finally, for each point in the strip we calculate the distance between that point and the 7 points around it and we compare this distance with the smaller distance found so far. It can be proven matematically that 7 is the maximum number of points to check since, if there would be closer points, a smaller distance would have been found in the previous step. (Decidere se spiegare meglio e/o aggiungere l'immagine)

D. Implementation

To implement the serial algorithm we started by implementing the `closest_points_divide` function that takes an array of points already sorted by the x coordinate and split the array in two halves and recursively computes the distances of each half using the `closest_points_rec` function. After the distances computation we use the smallest distances to select the points near the boundary. Then we use the `band_update_result` to sort these points by the y coordinate and check the distance between each point and the next 6 points.

III. Main steps towards parallelization

- Design of the parallel solution
- Implementation
- Benchmark on the HPC@UniTrento cluster

To move from serial to parallel we decided to split the ordered points between all the processes, then each process will perform the serial algorithm on the subset and will find its local best distance and finally the distances will be merged and the boundary points will be checked

IV. Final discussion

Conclusions

References

[1] S. Fortune and J. Hopcroft, "A note on rabin's nearest-neighbor algorithm," *Information Processing Letters*, vol. 8, no. 1, pp. 20–23, 1979.