NYC Vehicle Accidents Clustering

Presentation

Igor Iurevici

November 15, 2023

Università degli Studi di Bologna

Overview

Introduction

Implementation

- DBSCAN - Algorithm
- DBSCAN - Criticism
- DBSCAN - Distributed

Evaluation

- Google Cloud Platform (GCP)
- Performances

Conclusions

Introduction

Introduction [1/3]

New York City's streets are constantly busy and counts **hundred of thousands vehicle accidents every year** (∼100k). The record of each crash is crucial to adopt meaningful improvements to the traffic flow.

Introduction [2/3]

Each collision occurence is registered by the Police Department (NYPD) and provided by the **NYC OpenData**. Due to its continously increasing size (**over 2M crash records**), the dataset is becoming hard to analyze in a sequential manner.

Big data engineering

- In order to overcome the **time** and **space** constraints for big data computation, it is crucial to rely on **distributed systems**.

Introduction [3/3]

To explore the potential of the distributed systems on big data, a distributed **DBSCAN algorithm** has been implemented using Scala and Spark, exploiting cloud computing.

Reasons:

- DBSCAN is a clustering algorithm that can be useful to **detect patterns** on vehicle crush geospatial data.
- Time complexity: Naive $O(n^2)$, with Tree structured points $O(nlogn)$

The goal of this work is **Scalalability**.

Implementation

DBSCAN - Algorithm

DBSCAN algorithm **Flowchart**:

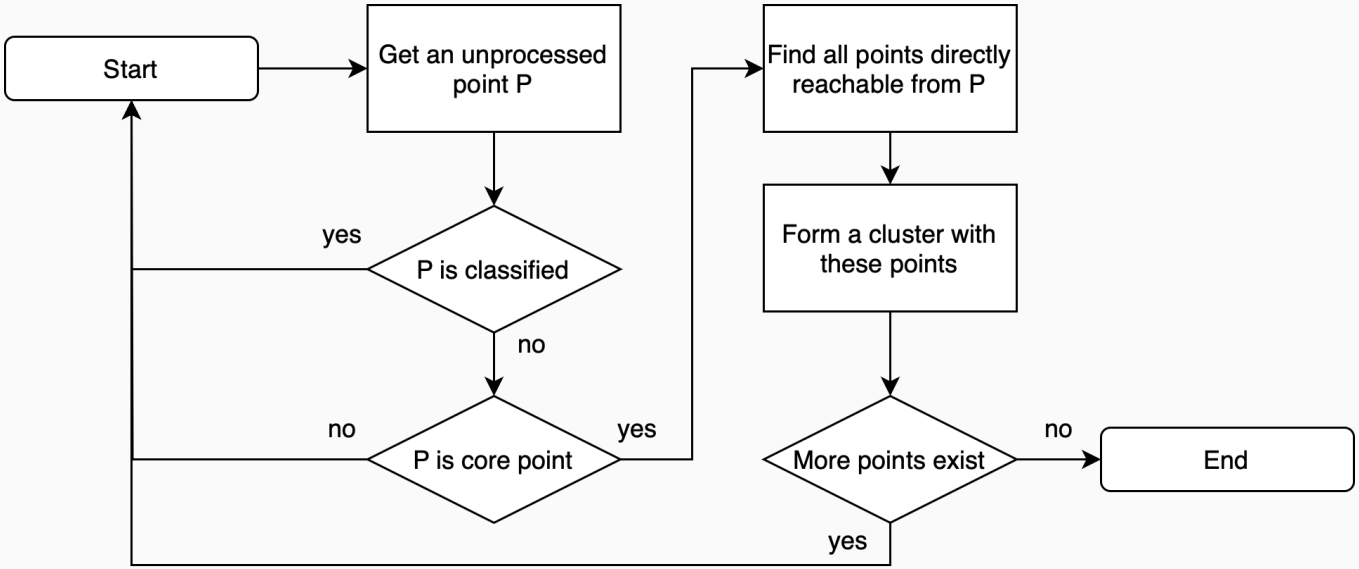

Figure 1: DBSCAN flowchart

DBSCAN - Criticism

Finding reachable points within Epsilon becomes **expensive** with large clusters.

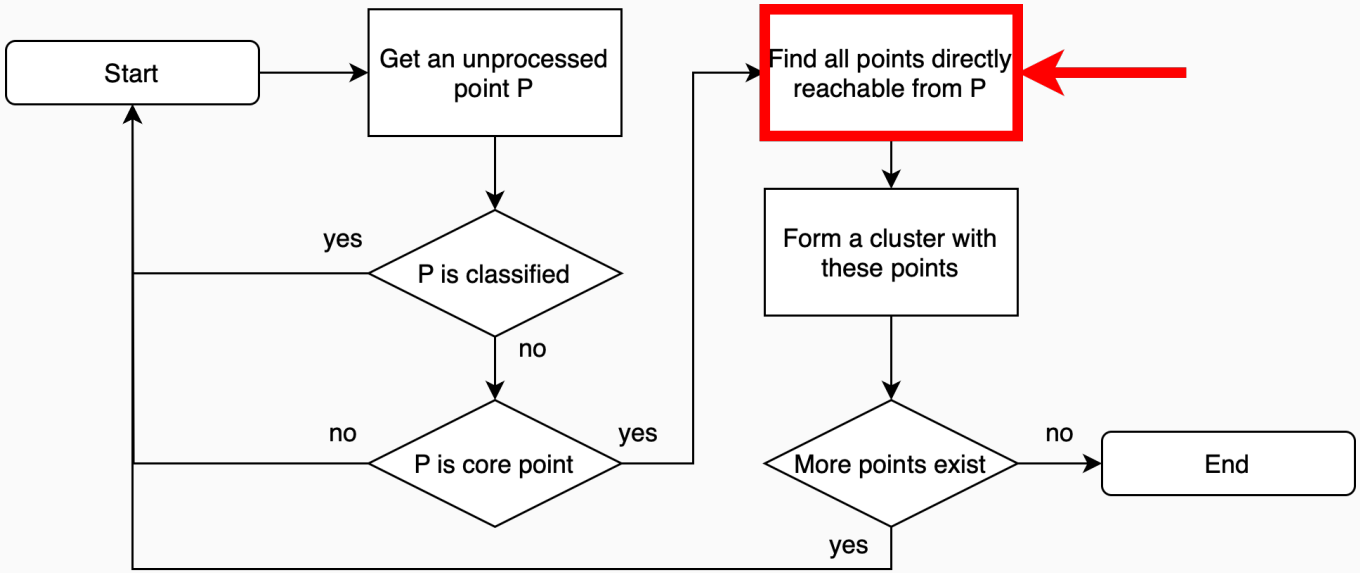

Figure 2: DBSCAN flowchart

DBSCAN - Distributed

For each core point, the **expansion of the cluster is parallelized**.

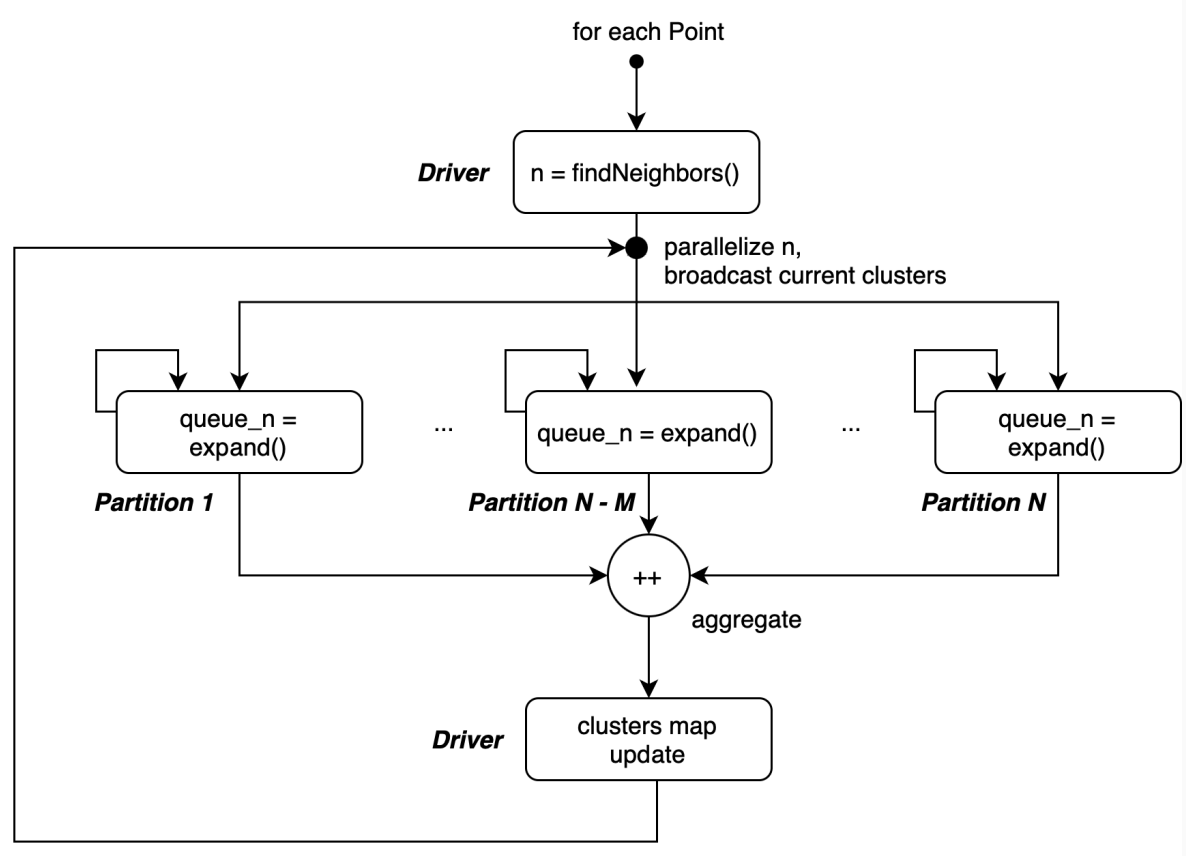

Figure 3: DBSCAN distributed

Evaluation

Google Cloud Platform (GCP) 1/2

The program is computed on a **GCP cluster** provided by **Dataproc** service.
The cluster's machines characteristics used for evaluation are the following:

- **n1-standard** machines (4 vCPUs, 100GB bootsize, Intel Skylake CPU) provided by us-west1 region;
- 1 Master and 5 Workers (total of **24 vCPUs**);
- Image version 2.1-debian11 (**Scala 2.12** and **Spark 3.3**).

Google Cloud Platform (GCP) 2/2

Both JAR and the dataset are stored in a **GCP Bucket**.
The program's result is visualized through a python script.

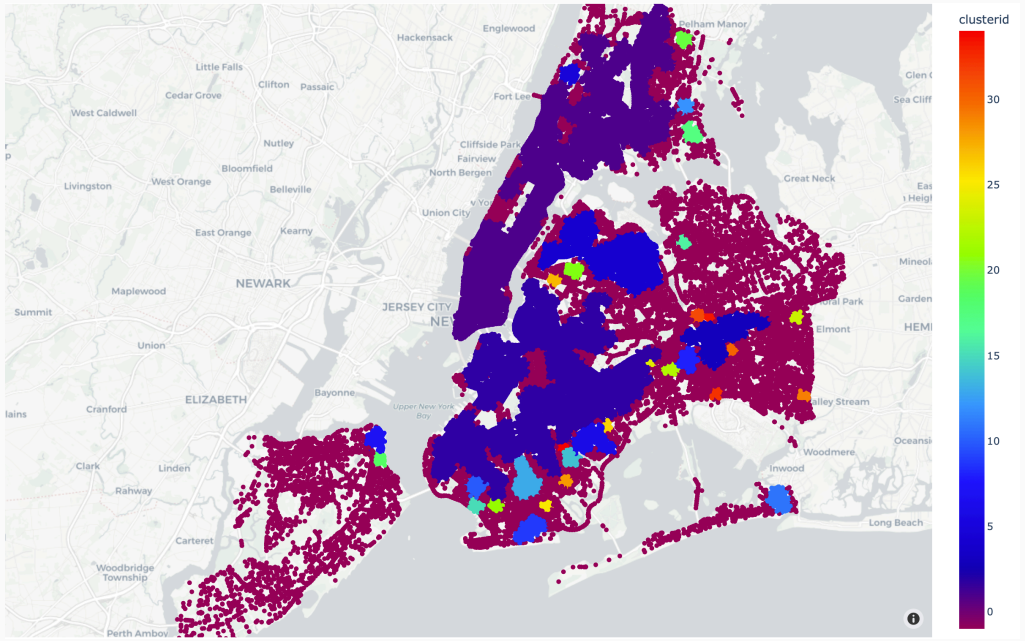

Figure 4: NYC accidents 2022: eps=0.4 (Haversine), minPoints=50, 34 clusters.

Performances

The performances and scalability of the program have been tested on **2**, **4**, **8** and **16 partitions** over **three datasets** with different sizes.

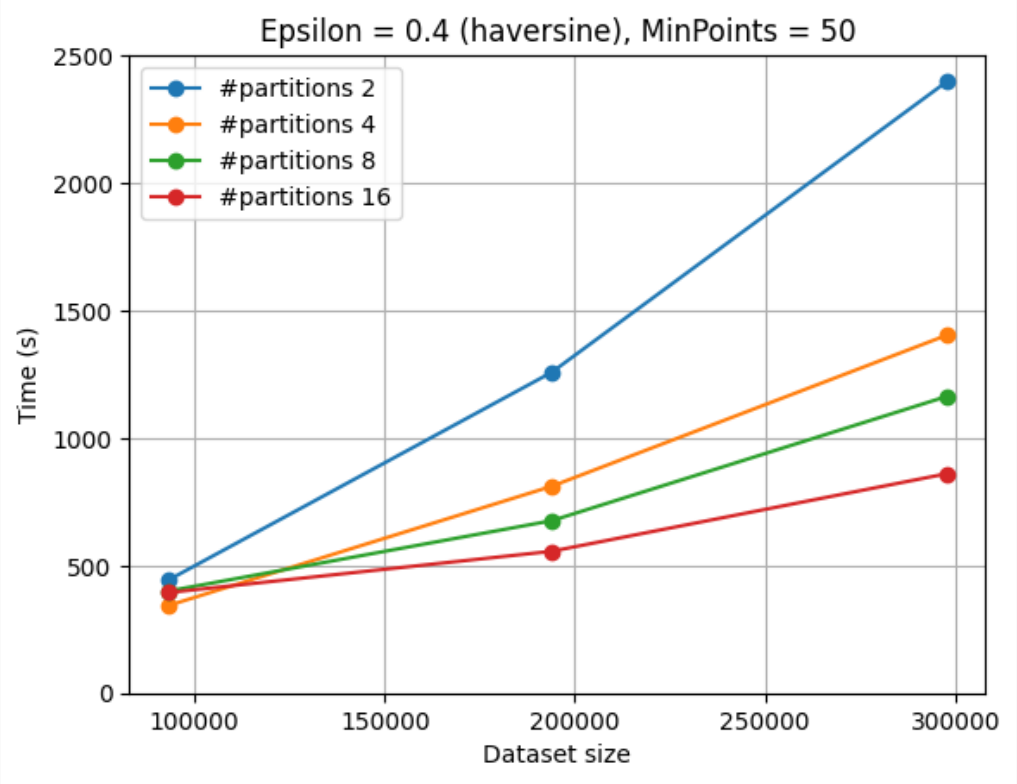

Figure 5: DBSCAN performance with different setups.

Conclusions

Conclusions

Results

- The algorithm benefits from distributed systems the more dataset size increases.

Limitations

- Parameters eps and minPoint play a crucial role. If the algorithm detects many small clusters, the scalability is reduced.

Future works

- Introduction of a tree data structure (R-Tree, KD-Tree) for $O(logn)$ region query.
- Helper for the choose of ideal parameters.

Questions?