ETH zürich

geoinformation engineering ikg eth zurich

# Real-Time Processing of Spatio-Temporal Data: Taxi Scheduling in New York City

**Geomatics Master – Master Thesis Spring 2018**

**Roswita Tschümperlin**
Geoinformation-Engineering, ETH Zurich
Supervisors: Dominik Bucher, Prof. Dr. Martin Raubal

## 1 Introduction

The emergence of autonomous mobility as well as the increasing amount of available mobility data ask for changes in the field of information technology. A possible deployment of autonomous taxis and the rising demand for car-sharing services require flexible and scalable systems that can automatically match available taxis to a large number of clients while considering the clients' demands.

In this thesis a Big Data streaming framework is used to develop an exploratory carpooling application that can match vehicles to client requests in real-time based on their current locations. Different approaches for a nearest neighbor matching will be compared and it will be assessed how well an external routing service can be integrated into the streaming architecture.

## 2 Application Architecture

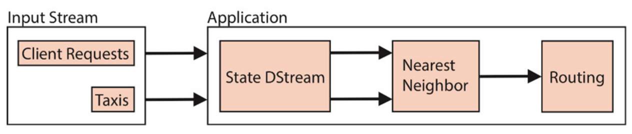

**Fig. 1.** Architecture of the carpooling application

The application receives client requests and taxi updates from two different input streams. The partial object updates are then combined to full taxi and client representations using a stateful stream.

### Nearest Neighbor

The top-*k* closest taxi candidates for each client are found using a nearest neighbor search. For this task four different approaches have been developed:

The Cartesian and Base-partition pipelines find the nearest taxis using the Euclidean distance between each taxi and client pair. The Cartesian performs a join between all taxi and clients and distributes the pairs across the cluster. The Base-partition pipeline first distributes the taxis, then collects the clients and sends them combined to each worker.

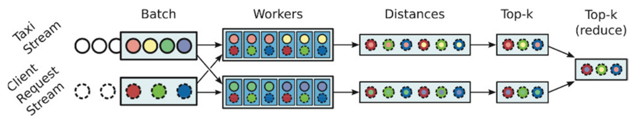

**Fig. 2.** Base-partition pipeline

The Separated and Multi-kNN pipelines use a spatial index for a collection of point objects (PointRDD). The Separated creates a PointRDD with all taxis and stores is locally. The nearest neighbor search is performed for each client separately using the taxi PointRDD. The Multi-kNN builds a PointRDD for both taxis and clients and performs a distributed nearest neighbor search.

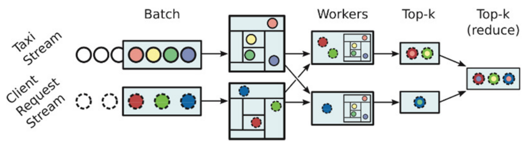

**Fig. 3.** Multi-kNN pipeline

### Routing

In a directed road network the nearest candidate is not necessarily the candidate that can reach the client fastest. The Open Source Routing Machine (OSRM) is used to compute the exact routes for all client-candidate pairs. The results from OSRM are further used to perform a route similarity check to find car-sharing options. The best candidate is assigned base on a minimal duration criterion.

## 3 Performance Tests

In order to evaluate the performance and scalability of the application and the different pipelines automated tests are run over a duration of 5 minutes. As application inputs two configurable streams deliver data retrieved from a real, historic taxi trip dataset from New York City.

The application is being deployed in a cluster of PCs. The input streams, application input point and cluster manager are running on one machine while the other computers represent the worker machines. Each worker has an instance of OSRM running locally.

## 4 Results and Discussion

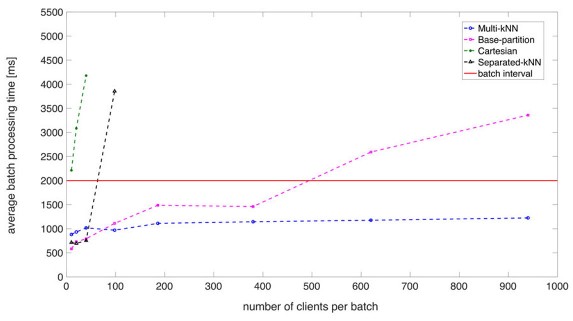

**Fig. 4.** Comparison of the pipelines with different client throughputs and 20'000 taxi updates per batch

The incoming data streams are split into small batches of a given duration. In order to perform real-time matching the processing time for the data within a batch must be smaller than the batch duration (resp. interval). The ideal batch interval depends on the data throughput and must be found in a trial-and-error procedure.

The Cartesian pipeline exceeded the batch interval of 2 sec already for small client throughputs due to its expensive join computation. The Separated pipeline performed well for small throughputs but was overburdened with more than 100 clients per batch. The Base-partition and Multi-kNN pipelines performed best, even for large client numbers. The superior performance of Multi-kNN over Base-partition might be due to its use of a spatial index.

The duration for each OSRM route query was measured and resulted in an average of 20 ms. Considering that for each client at least *k* queries have to be performed it can be said that the routing is too expensive for real-time matching.

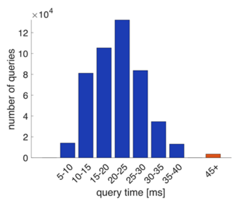

**Fig. 5.** Duration of OSRM route query

## 5 Conclusion and Future Work

A streaming application for carpooling has been developed and four different pipelines for nearest neighbor matching are introduced. The Base-partition and Multi-kNN pipeline present promising results and show that real-time nearest neighbor matching is possible. With an average route query duration of 20 ms real-time assignment of a taxi to a client request is not possible.

For a better evaluation longer performance tests must be run. The utilization of the cluster capacity should be further analyzed to gain better scalability. As a critical issue the routing procedure must be optimized in order to reach real-time processing times.