Broadcast Spatial Join **faster** than **Distributed** Spatial Join for trajectory enrichment

Georg Heiler, Allan Hanburry

Introduction

- More and more geospatial data is being proessed (IoT, 5G)
- Established tooling cannot scale
- Hadoop/Spark based implementaions promise to handle the load. But a naive implementation would be infeasible due to high load
- Multiple libraries implementing a distributed spatial join on spark exist
- Use Case: enrich trajectories with point of interest (POI) information

Experimental Setup

- Data is simulated with exponentially increasing load
- Multiple use cases (number of events per user, number of periods) are generated
- 3 methodologies are implemented
  1. GeoSpark non data locality preserving (default) only inner join. Time to un-nest the input is not counted
  2. Geospark locality preserving. Unnesting and aggregation afterwards are included as well as a left join
  3. broadcast spatial join.

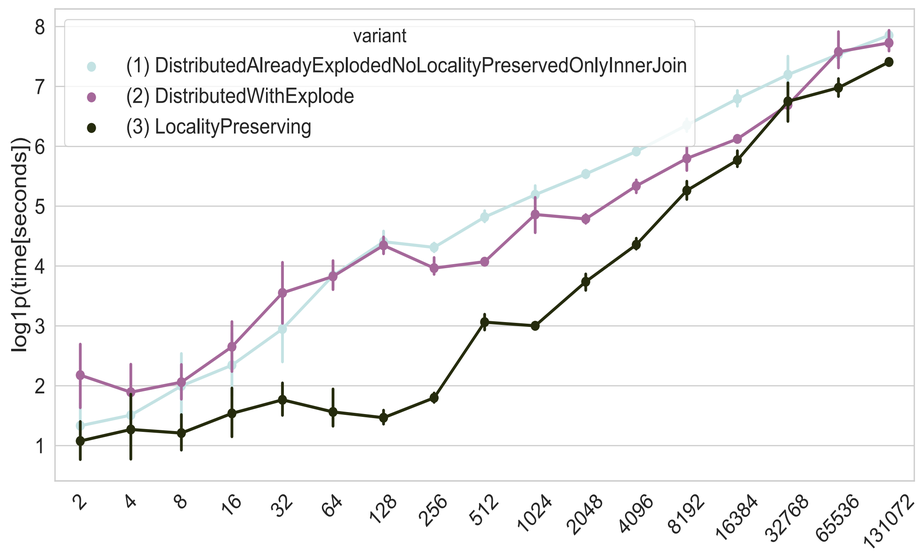
Results

- When considering 200 events per user for 3 periods and 10k POI, the broadcast spatial join is the fastest in most cases, but the advantage is diminishing for very large data sets
- When increasing the load per user the advantage of (3) increases

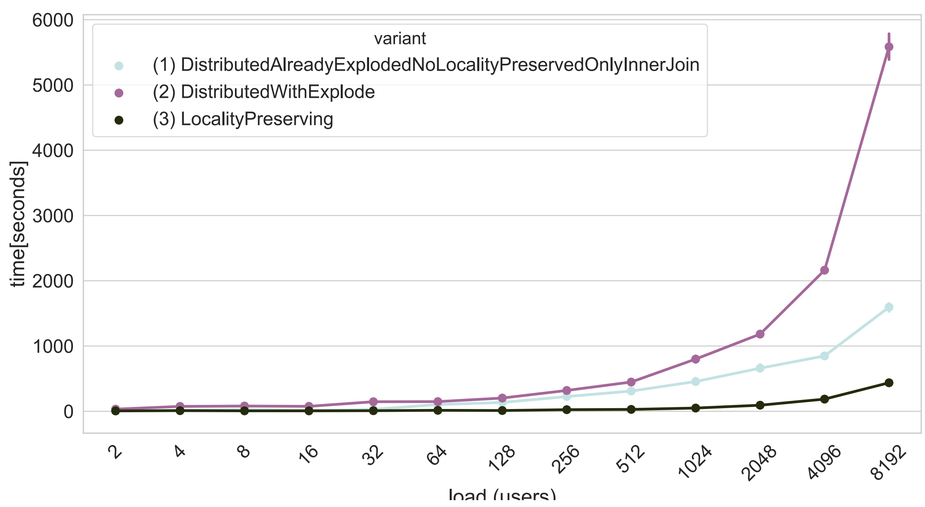
Summary

- A generic framework like geospark is useful in many cases
- For specific problems like trajectory enrichment with POI data a broadcast spatial join is more efficient

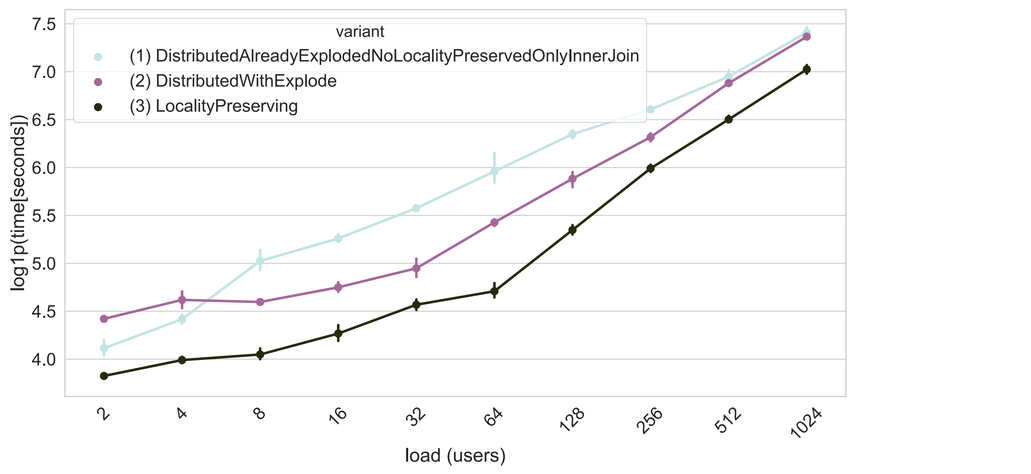
Scan me



200 events per user per period for 3 periods. Load of users (x axis), processing time shown in logarithmic scale (y axis) for the 3 different implementations of a spatial join. Each one was run 5 times. The graph shows the mean and 95% confidence intervals as error bars.



Increased number of events per user to 2000 events per period for 3 periods. Load of users (x axis), processing time shown in seconds (y axis), 5 runs each.



200 events per user per period. Increased number of periods to 300. Load of users (x axis), processing time as shown in logarithmic scale (y axis). For each methodology 5 runs were computed.

ttkkk

@geoheil heiler@csh.ac.at github.com/complexity-science-hub/distributed-POI-enrichmentt

This work was supported by the Austrian Research Promotion Agency FFG under grant # 857136.