ToSS-it: A Cloud-based Throwaway Spatial Index Structure for Dynamic Location Data

Afsin Akdogan, Cyrus Shahabi, Ugur Demiryurek
Integrated Media Systems Center
University of Southern California

Introduction

- The popularity of GPS enabled devices have led to a number of emerging applications.
- These applications include Facebook’s nearby friends, Foursquare’s adds, Uber taxi’s around me, etc.
- The increasing amount of data and large query volume brings the scalability problem.

Motivation

Goals

+ Handle dynamic datasets (moving objects)
+ Support spatial queries such as range and k nearest neighbor
+ Provide high query throughput (processed queries per second) to support internet scale applications
+ Scale linearly as the data grows

ToSS-it

- A distributed throwaway spatial index structure
- Adapts two-level parallelism (inter-node and intra-node)
- Distributed Voronoi diagram

Why throwaway

- Distributed updates are too costly due to high network overhead
- Rebuilding is faster than update because of extreme parallelism and lower network traffic

Challenge

- Partition the data and minimize network I/O
- Fixing inaccurate Voronoi cells at the border of the partitions
- Breaking the local Voronoi diagram generation problem into smaller pieces to enable intra-node parallelism

ToSS-it Construction (3-step Approach)

Partitioning

- Learn from the data and redistribute the objects to cluster spatially close objects together in the same server
+ efficient and no prior knowledge about the data is needed

Geospatial Replication

- Replicate only the borders
+ less network I/O

Local Voronoi Diagram Generation

- Break the problem into smaller independent pieces and use multi-core CPUs
+ intra-node parallelism
- Influence region: The region that can effect a Voronoi cell.
+ removes the merging bottleneck, thus provides high scalability

Load Balancing

- Query can be forwarded to any node

Performance Evaluation

- Dataset:
  a) Uniform (UD)
  b) Beijing Taxi dataset for 3 months (TD)
- Size: 60+ million objects

Conclusion

- Index construction time remains almost constant as the data size increases
- Provides high query throughput which increases linearly as more number of nodes are used