

Big Data and Spatial Databases

A Comparative Analysis

Joana Simões ¹, Rafael Giménez ¹, Marc Planagumà ¹

¹Bdigital

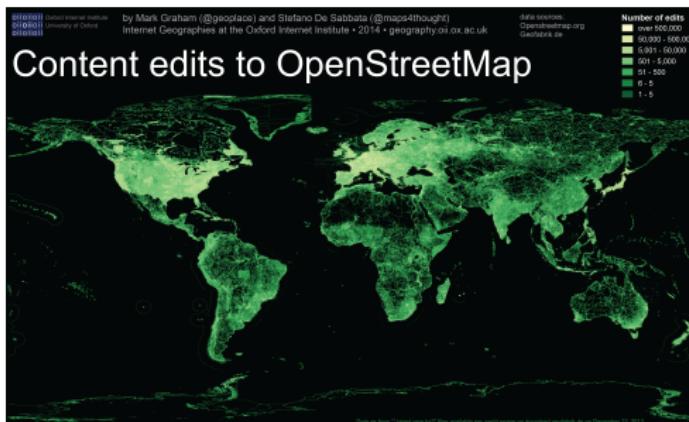
March 24, 2015

Table of Contents

- 1 Introduction
- 2 Benchmarking Setup
- 3 The Queries
- 4 Results
- 5 Conclusions

Motivation

- The volume of geospatial data has increased largely in the past few years.



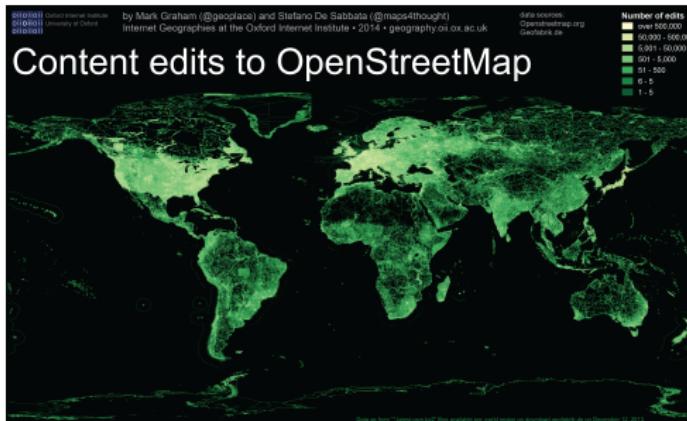
Motivation

- The volume of geospatial data has increased largely in the past few years.
 - Cheaper and widely adopted positioning technologies.



Motivation

- The volume of geospatial data has increased largely in the past few years.
 - Cheaper and widely adopted positioning technologies.
 - Spatial crowd-sourcing movements/ Volunteer Geographic Information (VGI).



What is so *Special* about *Spatial*

- Spatial queries rely on geometrical operations, not only for **computing measurements** and **generating new objects**, but also as **logical operations for topology relationships**.

What is so *Special* about *Spatial*

- Spatial queries rely on geometrical operations, not only for **computing measurements** and **generating new objects**, but also as **logical operations for topology relationships**.
- The increasing volume of spatial information, coupled with the computationally intensive nature of spatial queries demand scalable and efficient solutions.

What is so *Special* about *Spatial*

- Spatial queries rely on geometrical operations, not only for **computing measurements** and **generating new objects**, but also as **logical operations for topology relationships**.
- The increasing volume of spatial information, coupled with the computationally intensive nature of spatial queries demand scalable and efficient solutions.
 - fast query response, which requires a scalable architecture.
 - Support to clusters on a cost-effective architecture, such as commodity clusters or cloud environments.

Our Approach

- To track the performance of different spatial warehouse environments on the cloud, regarding a closed set of spatial queries.

Our Approach

- To track the performance of different spatial warehouse environments on the cloud, regarding a closed set of spatial queries.
- To compare a relational database server running on a single-machine with different clusters.

Our Approach

- To track the performance of different spatial warehouse environments on the cloud, regarding a closed set of spatial queries.
- To compare a relational database server running on a single-machine with different clusters.
- **Disclaimer:** we focused within the scope of our work environment.

The cloud



Introduction

The Queries

Results

Conclusions

RDS vs EMR

**Relational Database Service
(RDS)**

Elastic Map Reduce (EMR)

RDS vs EMR

Relational Database Service (RDS)

- RDBMS on a dedicated server, with an optimized configuration.

Elastic Map Reduce (EMR)

- A Linux-based distributed system that uses Hadoop as framework.

RDS vs EMR

Relational Database Service (RDS)

- RDBMS on a dedicated server, with an optimized configuration.
- PostgreSQL v.9.3.3
- PostGIS 2.1

Elastic Map Reduce (EMR)

- A Linux-based distributed system that uses Hadoop as framework.
- Hadoop 2.4.0
- Hive 0.13.1
- **GIS tools for Hadoop 2.0**

Introducing GIS tools for Hadoop



Introducing GIS tools for Hadoop

- FOS toolkit for "Big Spatial Data Analytics".
- Hosted on github.
- It consists in three libraries:
 - Esri Geometry API for Java.
 - Geoprocessing Tools for Hadoop.
 - **Spatial Framework for Hadoop (SFH)**: extends Hive to enable spatial queries and geometry types.



Hardware Description

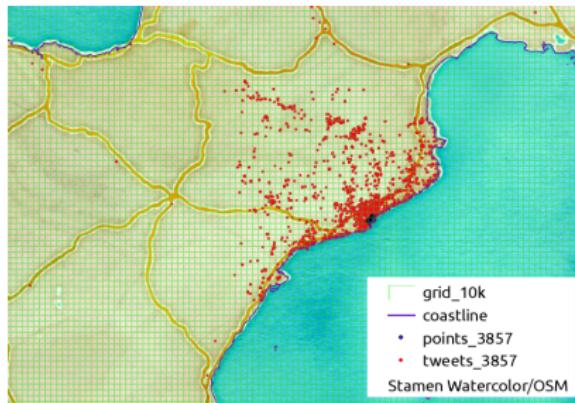
Designation	Description
RDS	db.m3.medium: 100 GB, 1 CPU, 3.75 GB RAM
EMRx3	master: m3.xlarge; cores: m1.medium (3)
EMRx6	master: m3.xlarge; cores: m1.medium (6)
EMRx9	master: m3.xlarge; cores: m1.medium (9)
EMRx3Large	master: m3.xlarge; cores: m3.xlarge (3)

m1.medium 410 GB, 3.75 GB RAM, 1 CPU:

m3.xlarge SSD 2x40 GB, 15 GB RAM, 4xCPU;

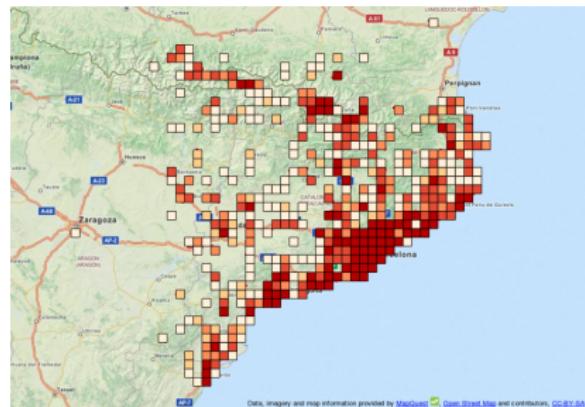
Introducing the Queries

- Four queries.
- Four Layers with different geometries (e.g: point, polygon, polyline).
- Same CRS: Google Mercator (EPSG:3857).
- Syntax may differ slightly on PostGIS and SFH.



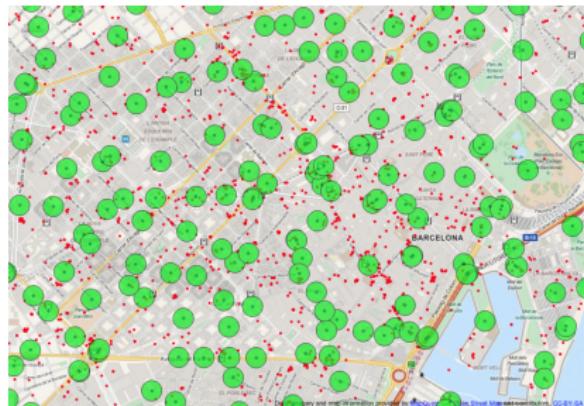
Create Density Grid

- Objective: transforming a point cloud in a density surface.
- Scenario: create a grid for displaying the density of tweets.
- Operations involved: create table, aggregation (GROUP BY), ST_CONTAINS.



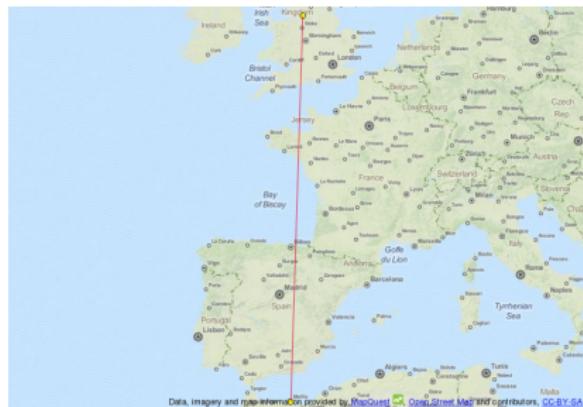
Buffers

- Objective: count how many points lie within a buffer.
- Scenario: how many tweets are sent within 100 m of a biking station.
- Operations involved:
ST_BUFFER,
ST_CONTAINS.



Select Maximum Distance

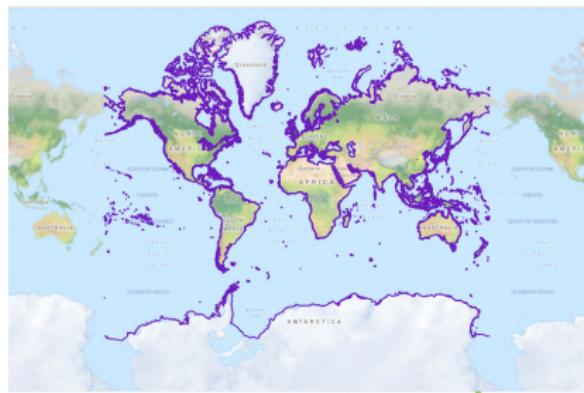
- Objective: measure distance between two points.
- Scenario: calculate the maximum distance between tweets mentioning New Years, on New Years Day (01/01/2015).
- Operations involved: MAX, ST_DISTANCE, filters (WHERE).



Data, Imagery and map information provided by MapQuest OpenStreetMap contributors, CC-BY-SA

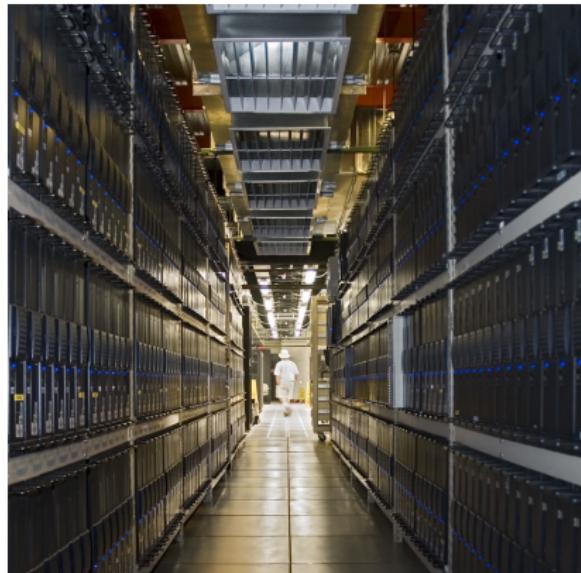
Import Layer

- Objective: importing a layer into the system.
- Scenario: instantiate geometry from a CSV file, with WKT definitions.
- Operations involved: create table, copy values, `ST_GeomFromText`.



Running the Queries

- Query times on clusters showed a large variability, so we used average values.



General Overview

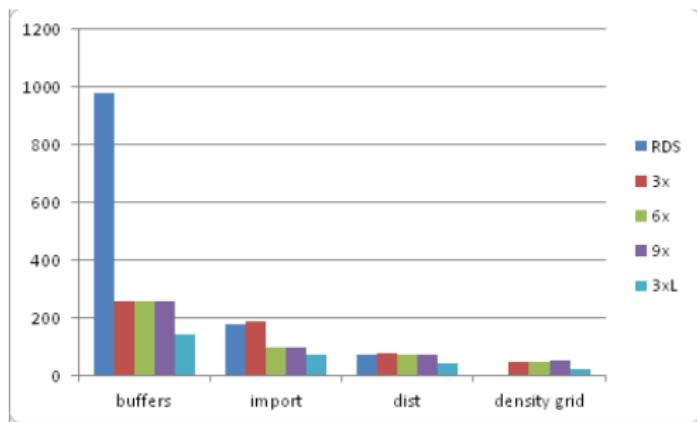


Figure : Results of the benchmarking of the different queries, on different setups on the cloud; query duration is shown in seconds.

General Overview

- The *buffers* query, is by far the most expensive one.

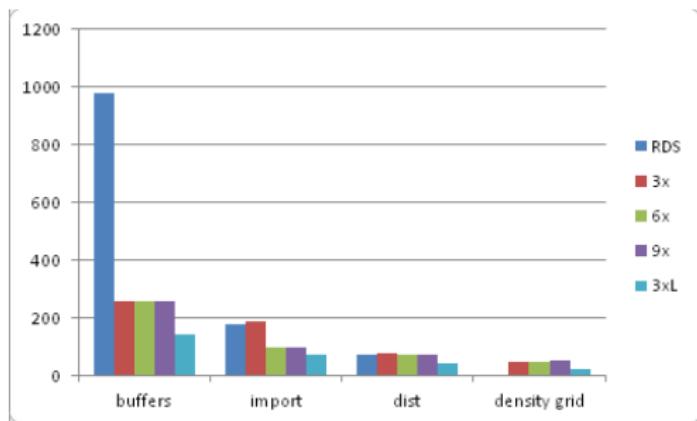


Figure : Results of the benchmarking of the different queries, on different setups on the cloud; query duration is shown in seconds.

Buffers

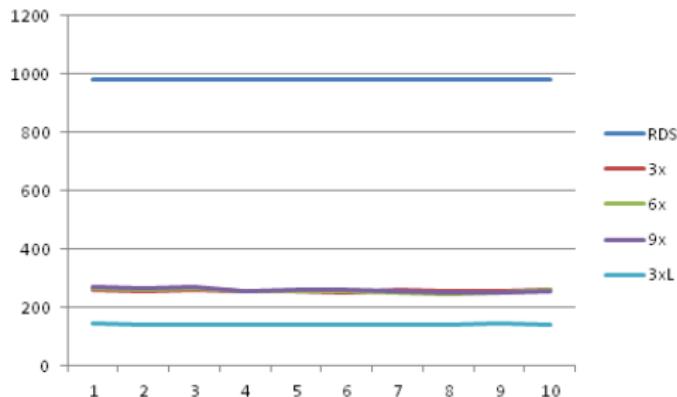


Figure : duration of the *buffers* query.

Buffers

- This is where RDS has its worst performance.

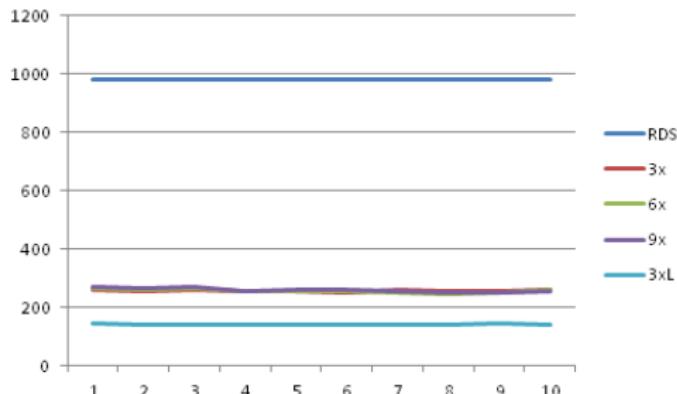


Figure : duration of the *buffers* query.

Import Layer

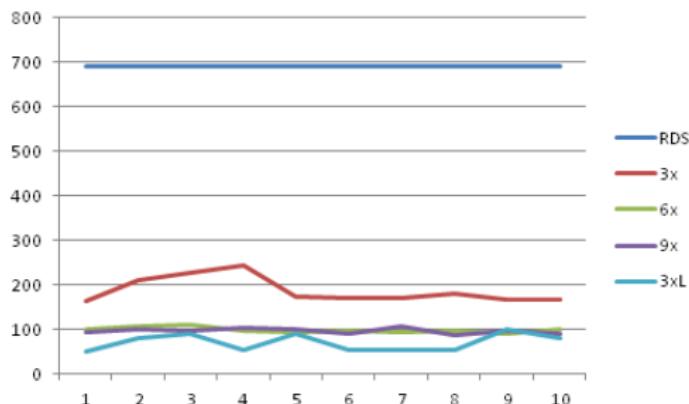


Figure : duration of the *import* query.

Import Layer

- On this query RDS also performs much worst than any of the clusters.

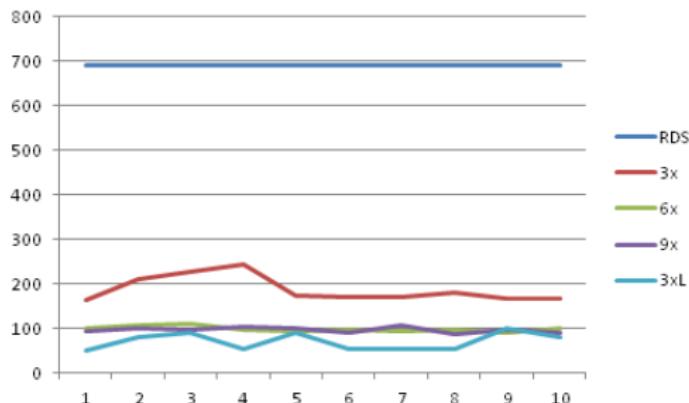


Figure : duration of the *import* query.

Import Layer

- On this query RDS also performs much worst than any of the clusters.
- Using an alternative method for importing (shp2pgsql) improves dramatically the import time for RDS.

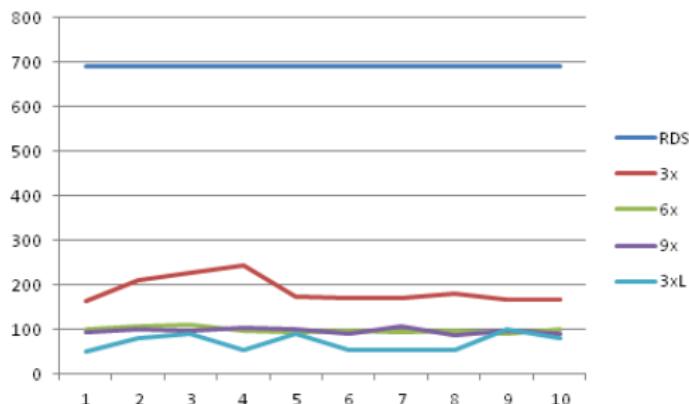


Figure : duration of the *import* query.

Create Density Grid

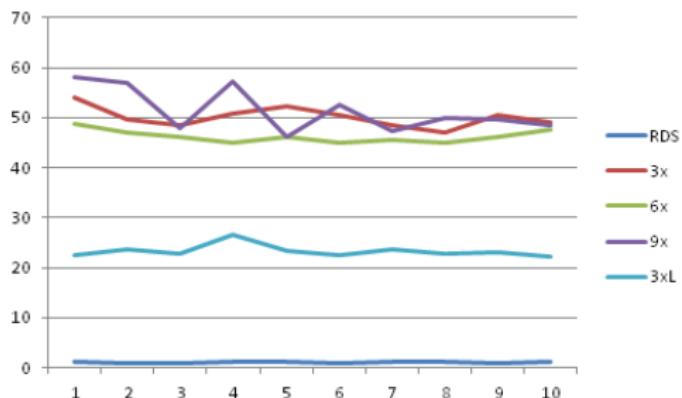


Figure : duration of the *grid* query.

Create Density Grid

- RDS is 20 to 50 times faster than the clusters.

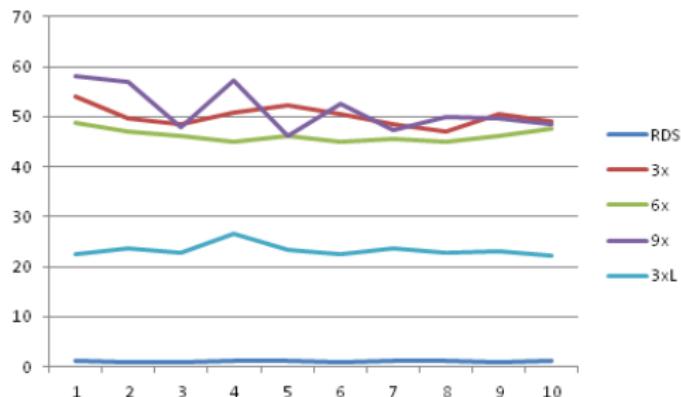


Figure : duration of the *grid* query.

Create Density Grid

- RDS is 20 to 50 times faster than the clusters.
- Using more workers (9), actually results in longer response times.

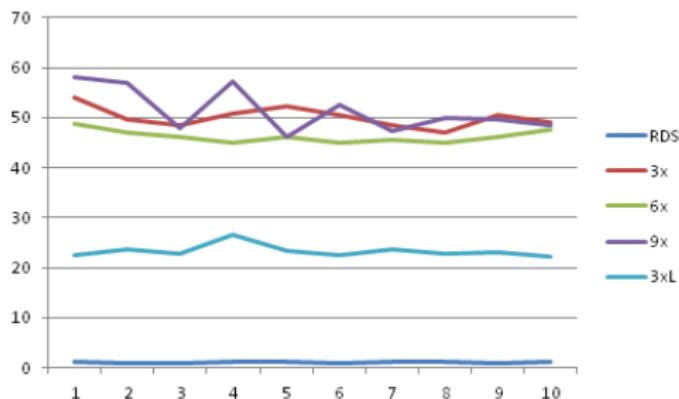


Figure : duration of the *grid* query.

Select Maximum Distance

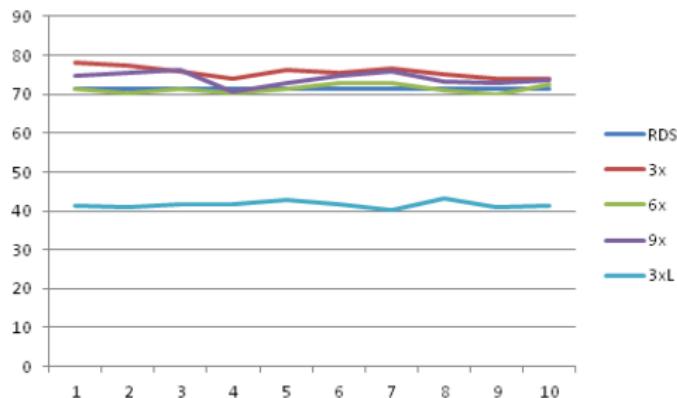


Figure : duration of the *distance* query.

Select Maximum Distance

- Second less expensive query.

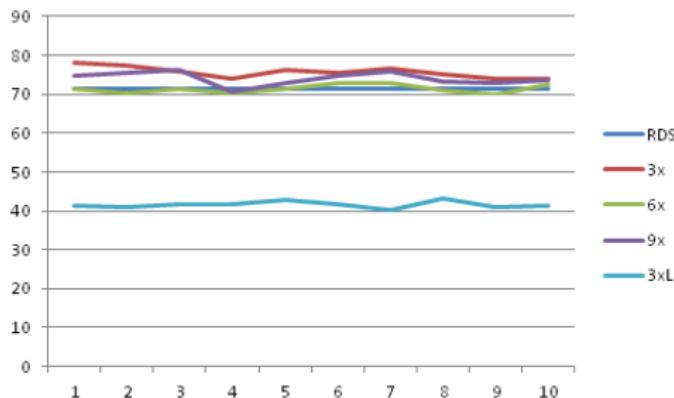


Figure : duration of the *distance* query.

Select Maximum Distance

- Second less expensive query.
- RDS performs better than the smaller cluster (EMRx3).

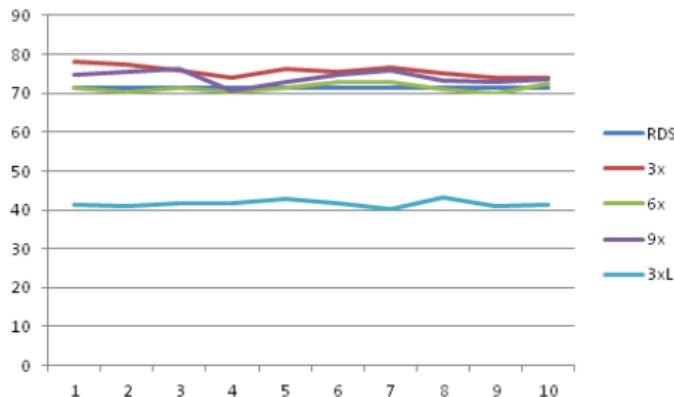


Figure : duration of the *distance* query.

\$\$\$



Costs

- The costs below were calculated using the AWS calculator.
- Since all the queries were completed in less than 1 hour, we used the cost of 1h/mnth.

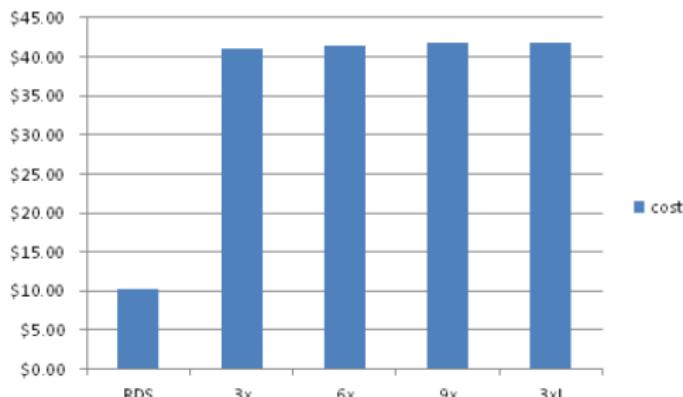


Figure : Cost in dollars, for deploying each benchmarking environment in AWS.

Time for Some Conclusions?



Disruptive Conclusions

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.
- Adding nodes to a cluster does **not** necessarily result in a better performance.

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.
- Adding nodes to a cluster does **not** necessarily result in a better performance.
- It was nearly always the case, that the best performance was achieved using a small cluster with *large* machines.

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.
- Adding nodes to a cluster does **not** necessarily result in a better performance.
- It was nearly always the case, that the best performance was achieved using a small cluster with *large* machines.
- Price-wise, RDS and EMRx3Large are good deals.

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.
- Adding nodes to a cluster does **not** necessarily result in a better performance.
- It was nearly always the case, that the best performance was achieved using a small cluster with *large* machines.
- Price-wise, RDS and EMRx3Large are good deals.
- On the context of this benchmarking, the cluster with more nodes (EMRx9) seems like a waste of money.

Disruptive Conclusions

- Don't throw PostgreSQL away: a cluster is not **always** preferable to RDS.
- Adding nodes to a cluster does **not** necessarily result in a better performance.
- It was nearly always the case, that the best performance was achieved using a small cluster with *large* machines.
- Price-wise, RDS and EMRx3Large are good deals.
- On the context of this benchmarking, the cluster with more nodes (EMRx9) seems like a waste of money.
- These disruptive results enforce the urge to do more benchmarkings, with larger sample sizes, and to review the code from SFH.

References

- Dunning, T. and Friedman, E. (2014) Time Series Databases: New Ways to Store and Access Data. O'Reilly Media; 1 edition.
- OpenStreetMap. OpenStreetMap. Retrieved March 9, 2015, from: <https://www.openstreetmap.org/>
- Ushahidi. Ushahidi. Retrieved March 9, 2015, from: <http://www.ushahidi.com/>
- Aji, A.; Wang, F.; Vo, H.; Lee, R.; Liu, Q.; Zhang, X. & Saltz, J. (2013). Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. Proceedings of the VLDB Endowment International Conference on Very Large Data Bases, 6(11), p1009.
- Amazon. Amazon Web Services. Retrieved March 15, 2015, from: <https://aws.amazon.com/>
- PostgreSQL. PostgreSQL. Retrieved March 15, 2015, from: <http://www.postgresql.org/>
- License. PostgreSQL. Retrieved March 9, 2015, from: <http://www.postgresql.org/about/licence/>
- PostGIS. Spatial and Geographic objects for PostgreSQL. Retrieved March 9, 2015, from: <http://postgis.net/>
- Apache. Hadoop. Retrieved March 9, 2015, from: <https://hadoop.apache.org/>
- The Apache Software Foundation. Apache License, Version 2.0. Retrieved March 15, 2015, from: <http://www.apache.org/licenses/LICENSE-2.0>
- ESRI. Spatial Framework for Hadoop. Retrieved March 15, 2015, from: <https://github.com/Esri/spatial-framework-for-hadoop>
- ESRI. Spatial Framework for Hadoop license. Retrieved March 15, 2015, from: <https://github.com/Esri/spatial-framework-for-hadoop/blob/master/license.txt>
- Stolze, K. SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems. Retrieved March 15, 2015, from: <http://doesen0.informatik.uni-leipzig.de/proceedings/paper/68.pdf>
- Wikipedia. Bicing. Retrieved March 15, 2015, from: <https://en.wikipedia.org/wiki/Bicing>
- Stack Overflow. What is the best hack for importing large datasets into PostGIS? Retrieved March 9, 2015, from: <http://gis.stackexchange.com/questions/109564/what-is-the-best-hack-for-importing-large-datasets-into-postgis>
- Amazon Web Services. Simple Monthly Calculator. Retrieved March 9, 2015, from <https://calculator.s3.amazonaws.com/index.html>

Thank You!



This presentation is available at:
<http://tinyurl.com/q9rh3co>