Scalable Spatio-temporal Indexing and Querying over a Document-oriented NoSQL Store

Nikolaos Koutroumanis, Christos Doulkeridis

Department of Digital Systems
University of Piraeus (UPRC), Greece

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780754 (Track&Know project), and the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT) under grant agreement No HFRI-FM17-81 (Chorologos project).

Contents

I. Introduction/Motivation

II. Baseline

III. Our approach

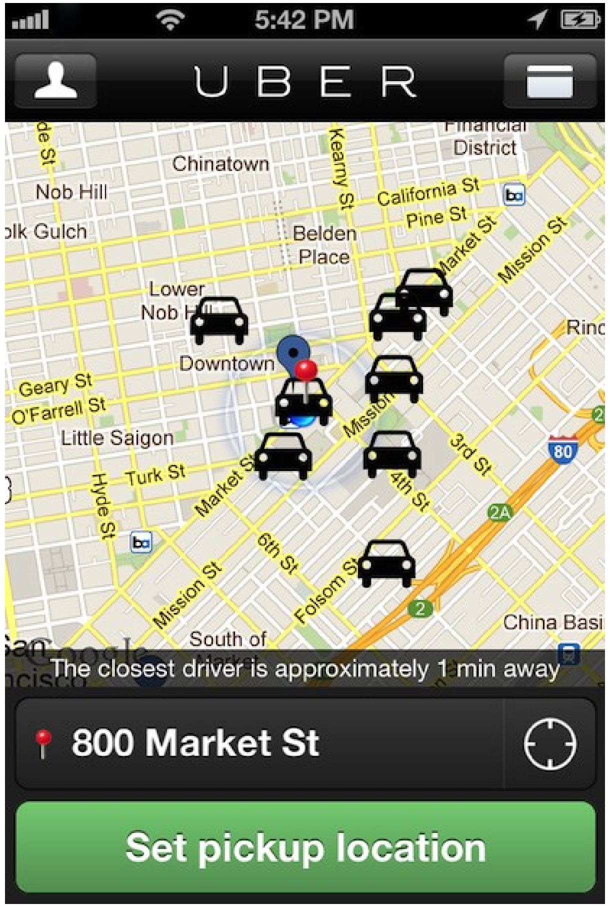
IV. Usage of zones

V. Experiments

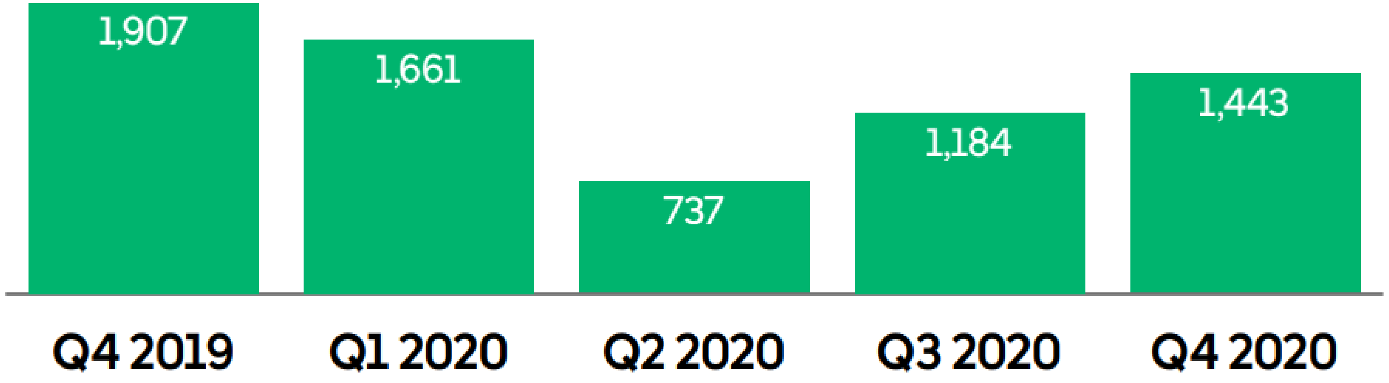
VI. Conclusions

Introduction/Motivation (1/2)

- In recent years the increasing size of the spatio-temporal data requires new approaches for their storage and retrieval
- Scalable querying of spatio-temporal data management is a challenging topic
- Uber reports Millions of trips per quarter year;



Uber Technologies, Inc. Q4 2020 Earnings, Supplemental Data



Introduction/Motivation (2/2)

- NoSQL stores are exploited by modern applications for data storage and querying, providing scalability and availability

- Despite the popularity of NoSQL systems, they are not optimized for spatial data

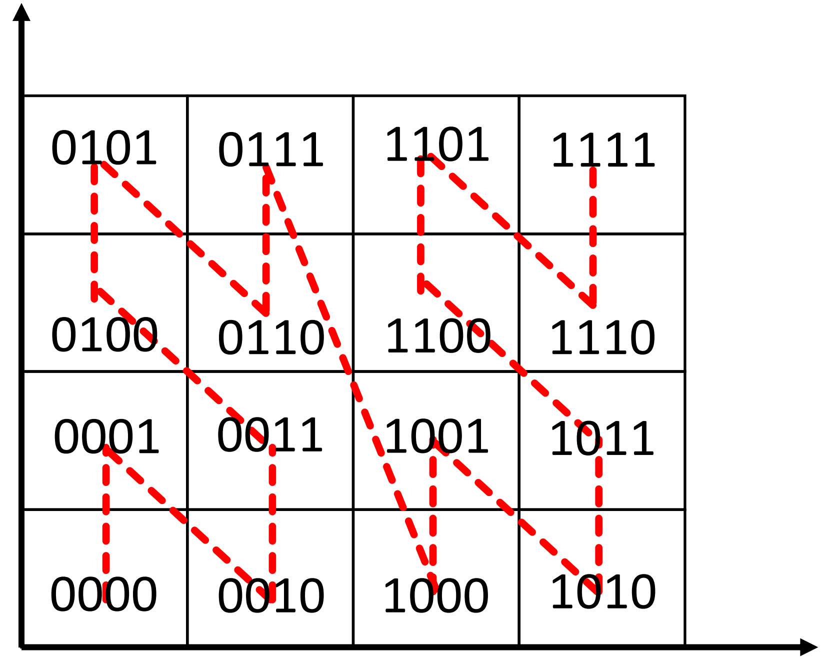
- In this work we opt for MongoDB store to support efficient spatio-temporal querying, as it embeds geospatial features

Baseline (1/3)

- MongoDB offers spatial indexes (2d/2dsphere)
- The geospatial **2dsphere** index is based on geohashing

{location: “2dsphere”}

- The geohash values are indexed by *B-trees*

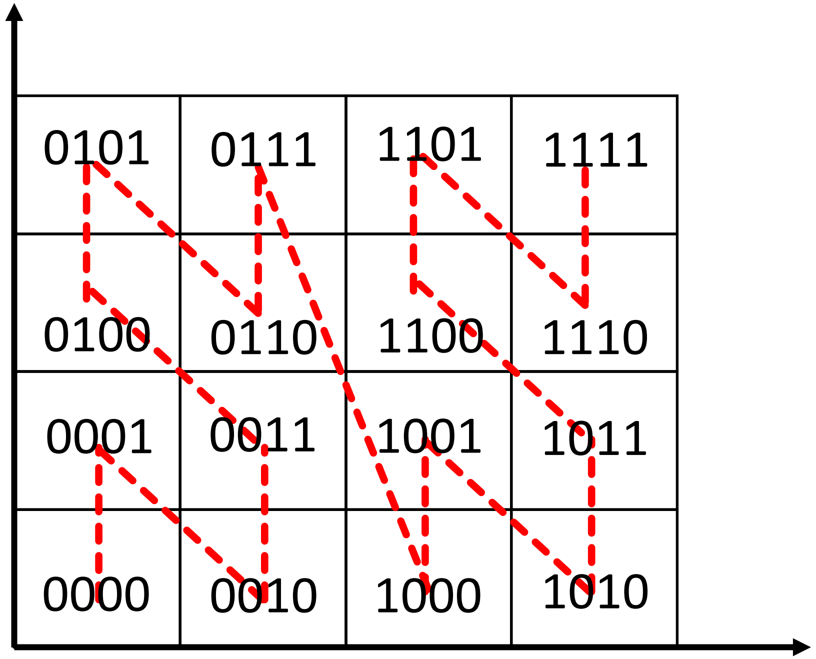


Z-order curve

```
{
 _id: 1,
 location: {"type": Point, coordinates: [37.983810, 23.727539]},
        ...
}
```

Baseline (2/3)

- MongoDB also offers compound indexes for indexing two or more fields in a single structure
- A spatio-temporal index in MongoDB is created in the two following ways:
  - `{ location: “2dsphere”, date: 1 }`
  - `{ date: 1, location: “2dsphere” }`



Z-order curve

```
{
 _id: 1,
 location: {"type": Point, coordinates: [37.983810, 23.727539]},
 date: ISODate("2018-09-12T12:15:17.777Z"),
       ...
}
```

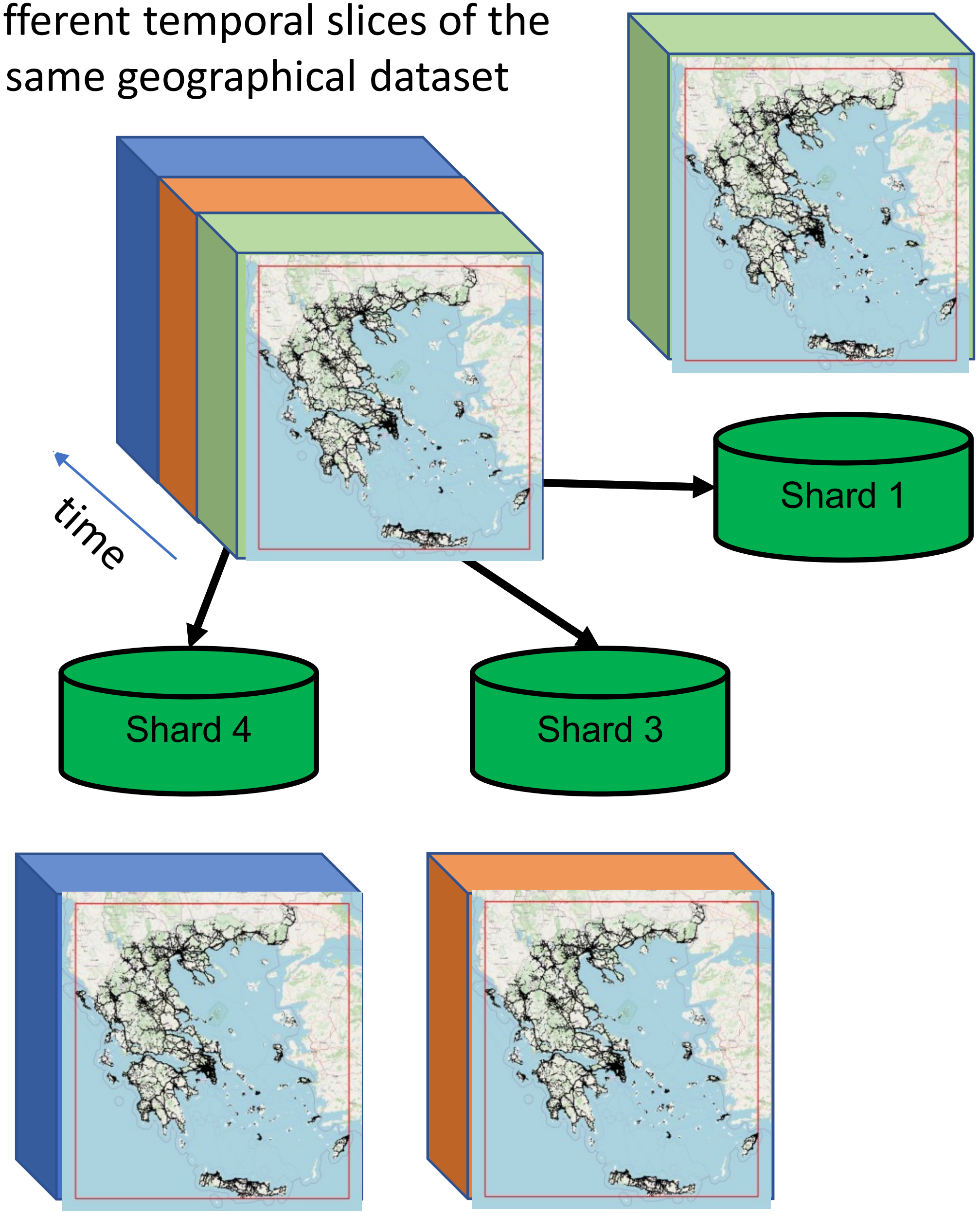
Baseline (3/3)

- However, MongoDB cannot distribute documents on a shards based on their spatial information
- Thus, we are restricted to the integration of the temporal field only as a shard key for spatio-temporal querying:

  `{date: 1}` **(bsl)**

- On each shard we create one of the following compound indexes (co-existing with a local single index based on the date field)
  - `{ location: “2dsphere”, date: 1 }` **(BslST)**
  - `{ date: 1, location: “2dsphere” }` **(BslTS)**

Different temporal slices of the same geographical dataset



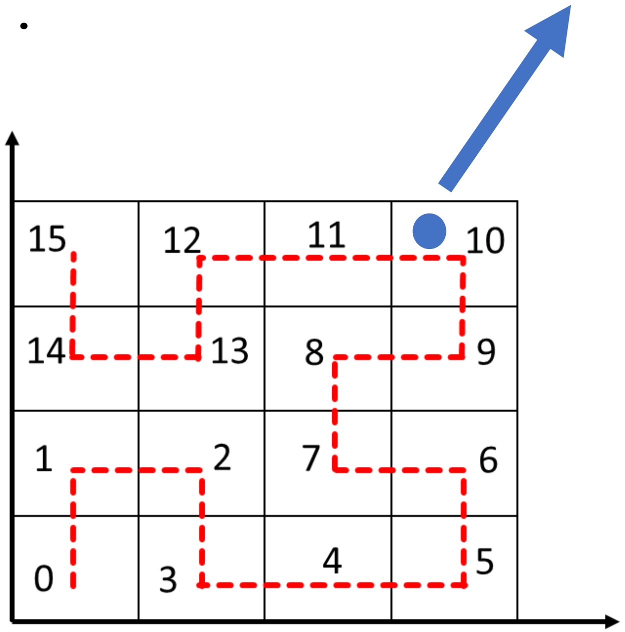
Our approach (1/2)

- We exploit the Hilbert space-filling curve and integrate the 1D numeric value in a new field in each document

- We form the compound indexes based on the fields:

  {hilbertIndex: 1, date: 1}

```
{
 _id: 1,
 location: {"type": Point,
        coordinates: [37.983810, 23.727539]},
 date: ISODate("2018-09-12T12:15:17.777Z"),
 hilbertIndex: NumberLong(10),
 ...
}
```



Hilbert curve

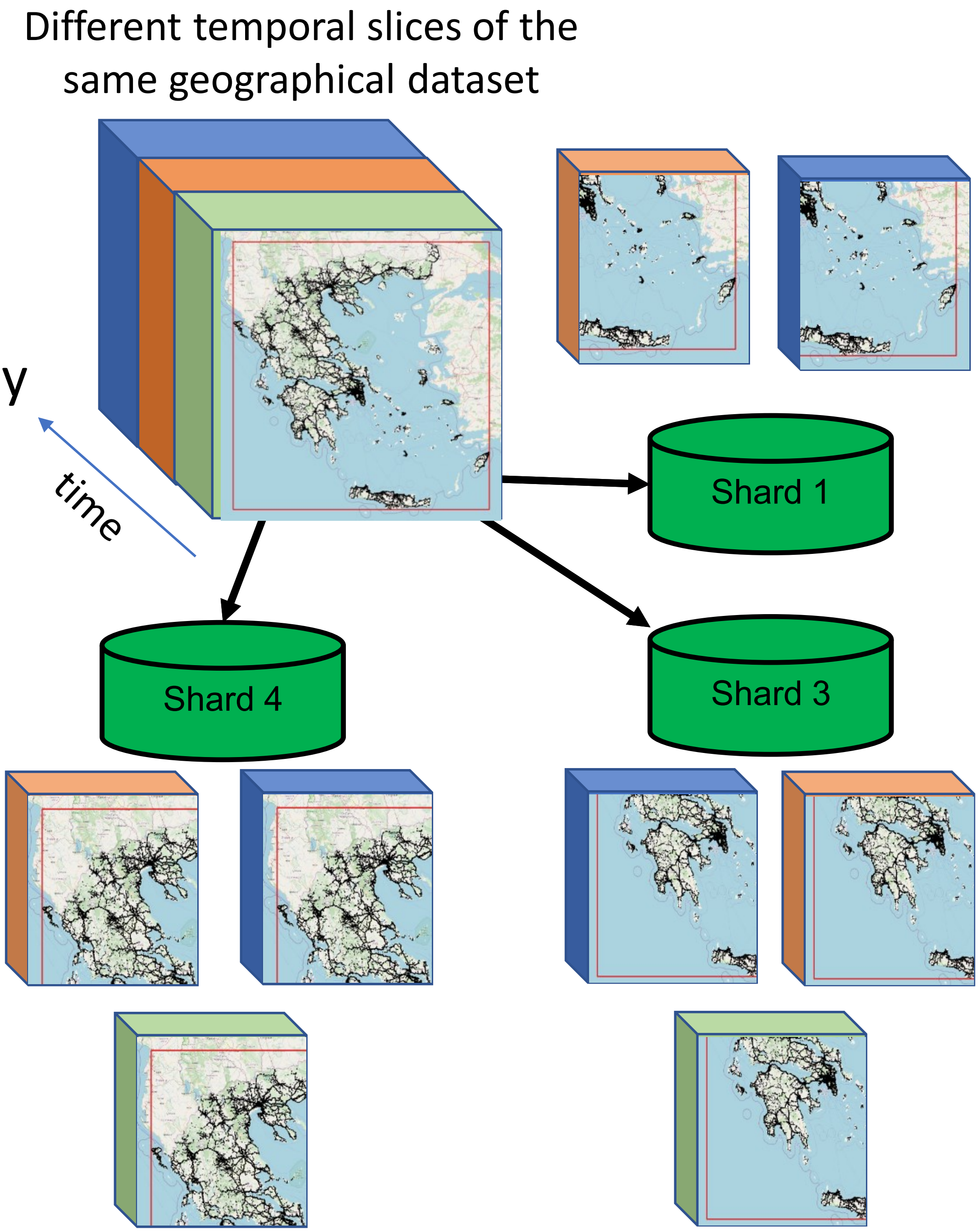
Our approach (2/2)

- Since the spatial information is embedded in every document as a numeric field, we use the shard key:

  `{hilbertIndex: 1, date: 1}` <u>(hil)</u>

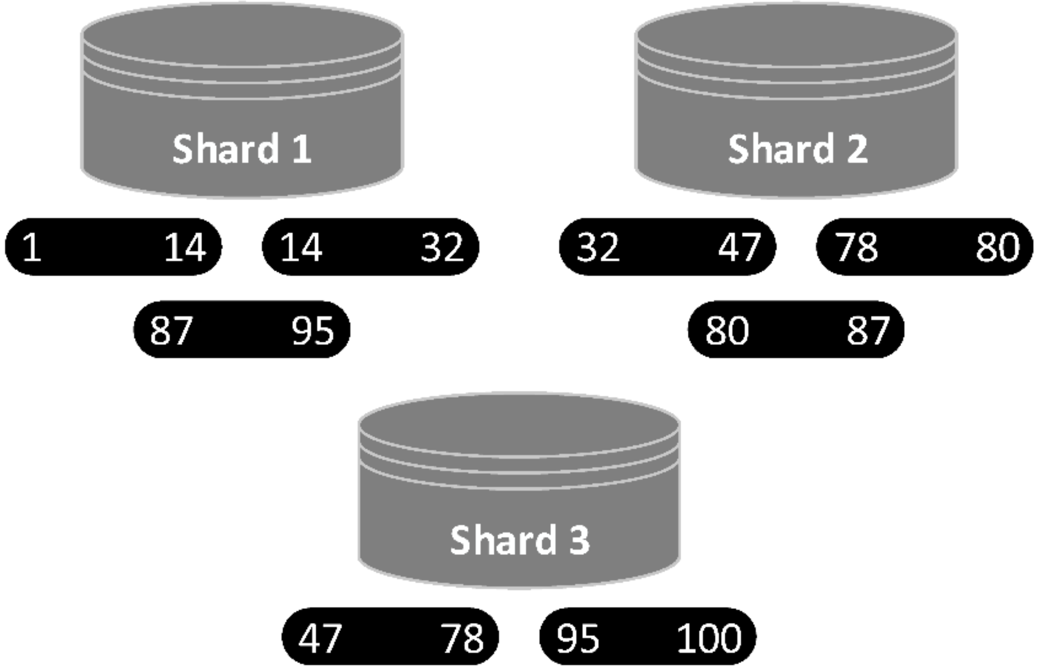
- By default, each shard has compound indexes based on the declared fields in the shard key

- Shard overloading is unlikely to occur in case of spatial/spatio-temporal skewness



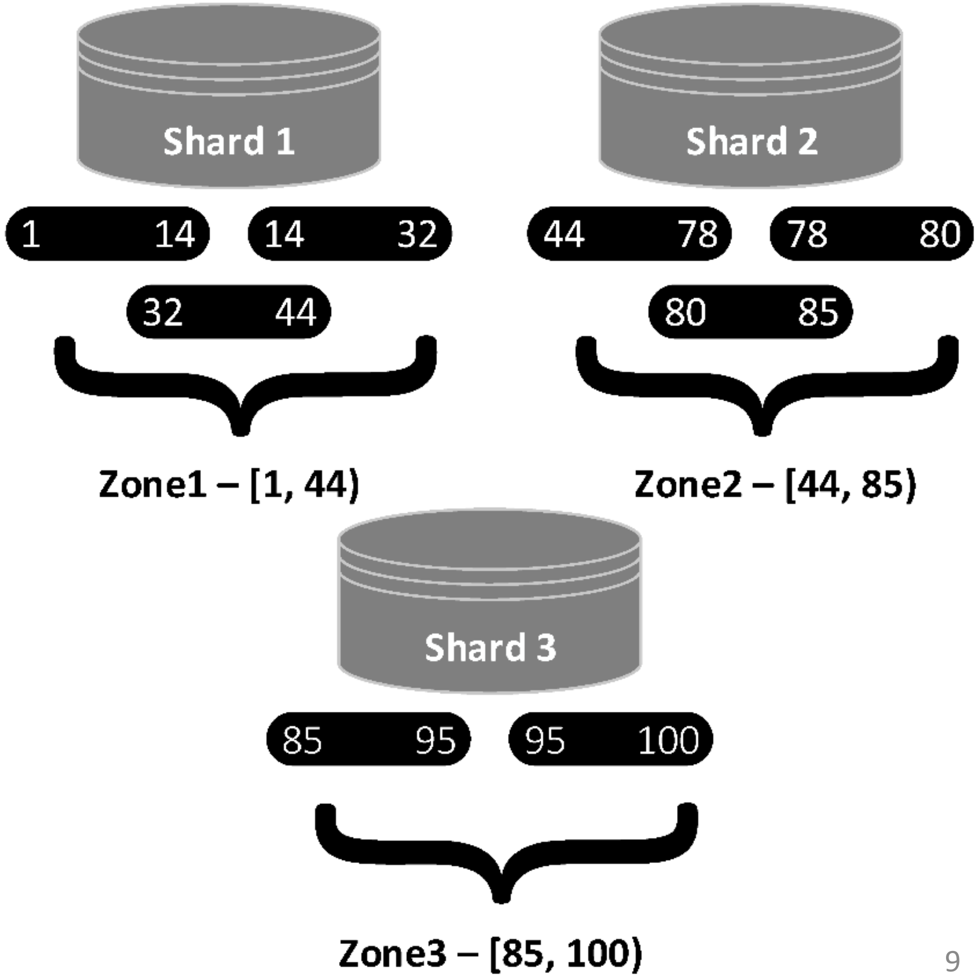
Usage of zones

Default distribution



Continuous ranges of shard keys (*chunks*) are distributed evenly in the cluster

Zone usage

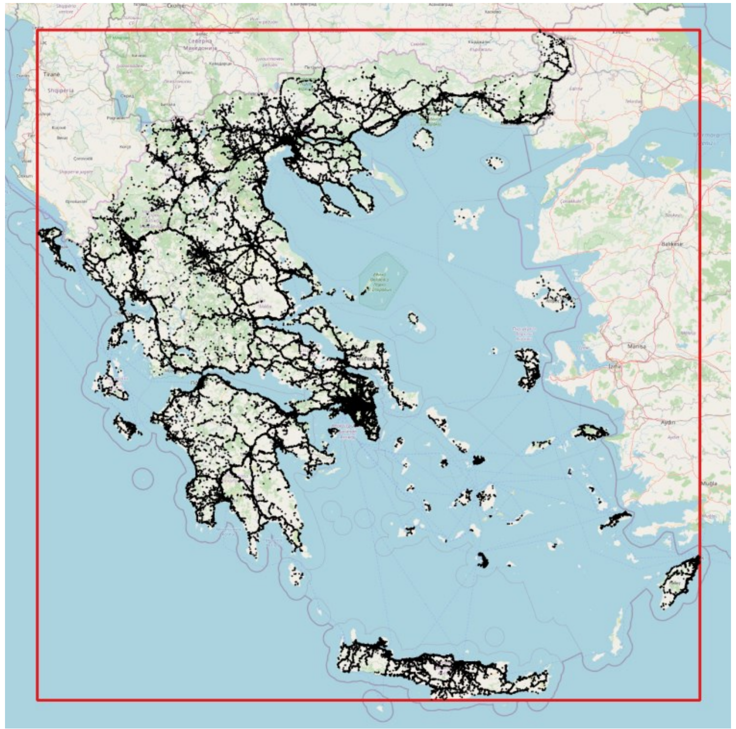


Experiments (1/5) – Setup

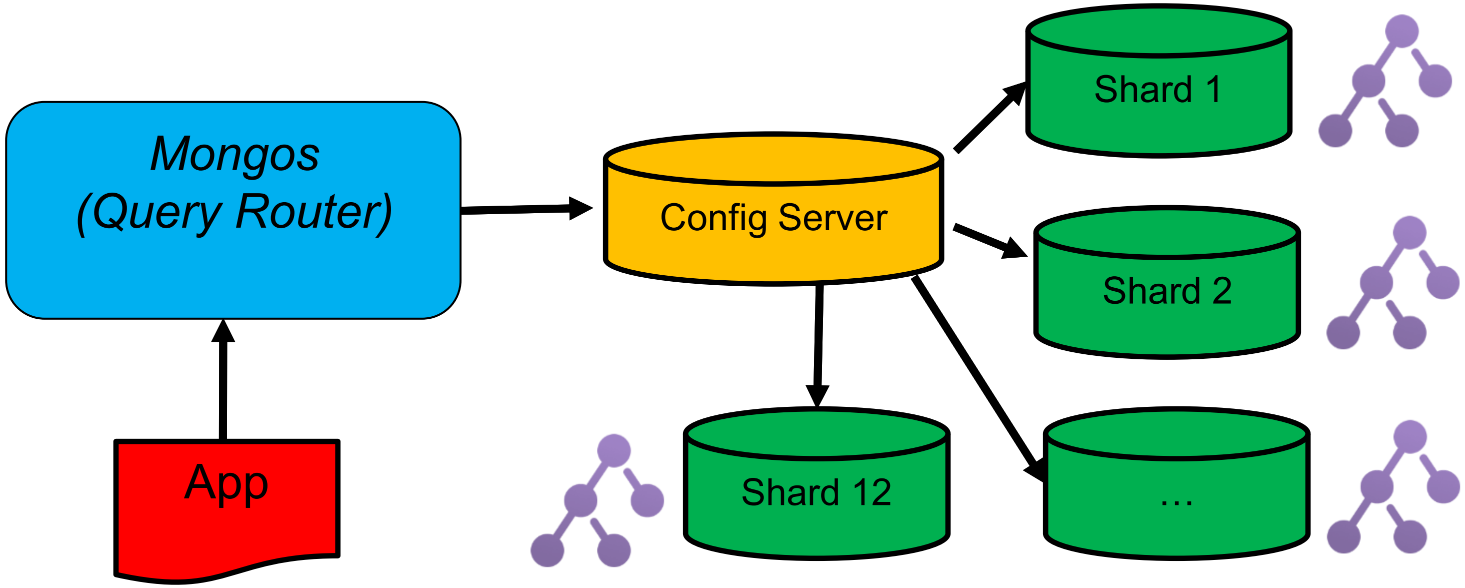
Real data set	
Size in MongoDB	*#points*
~40GB	**15.2M**

- Assess the efficiency of the proposed Hilbert-based approach against the baseline on a MongoDB Cluster
- We consider two types of spatio-temporal queries; the “spatially small” ($\boldsymbol{Q_x^s}$) and the “spatially big” ($\boldsymbol{Q_x^b}$)

Time periods
$\boldsymbol{Q_1^x}$ **–** One hour
$\boldsymbol{Q_2^x}$ **–** One day
$\boldsymbol{Q_3^x}$ **–** One week
$\boldsymbol{Q_4^x}$ **–** One month



Experiments (2/5) - Setup

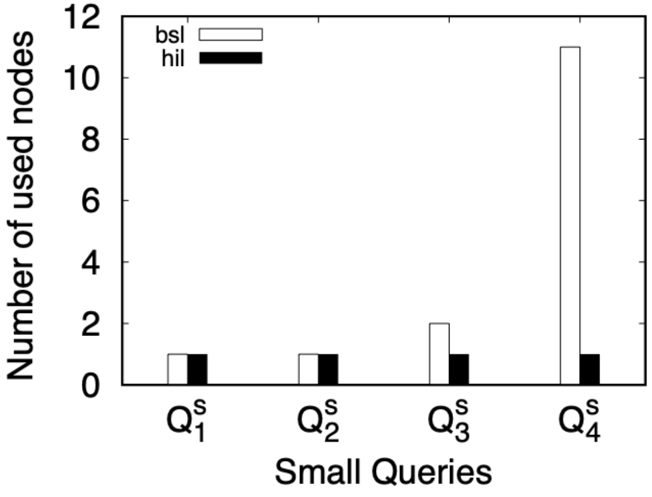


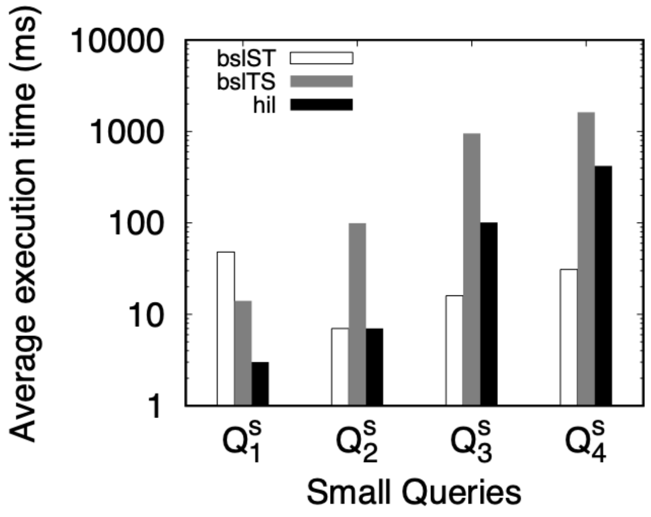
Execution of queries

17 nodes, 8GB RAM, x4 CPU cores,
12 nodes are used as shards with 100GB disk

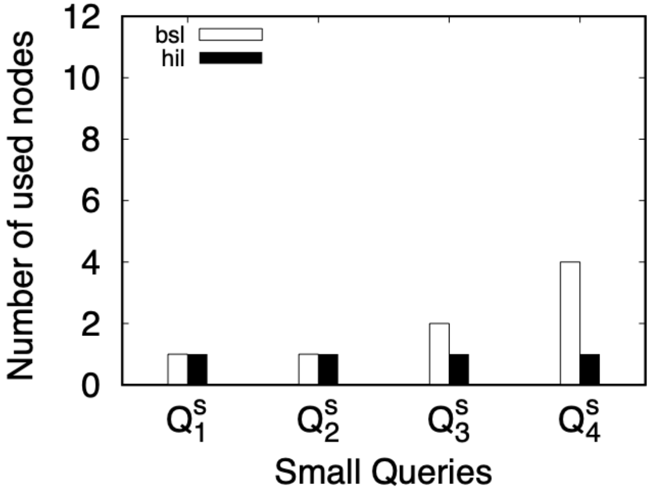
Experiments (3/5) - Performance of “spatially small” ST queries

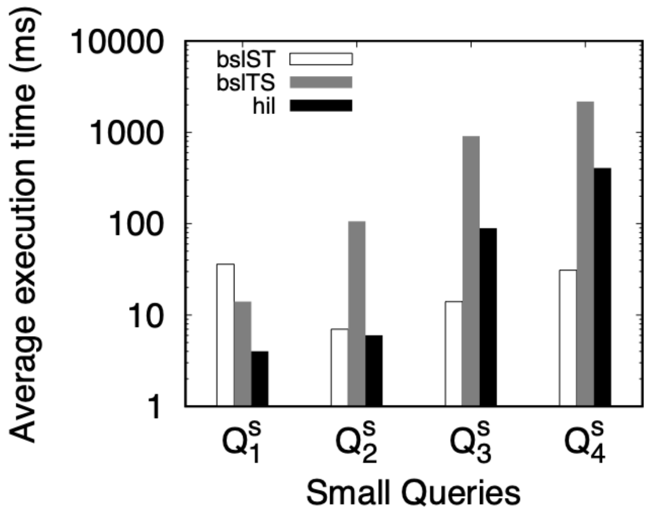
Default distribution





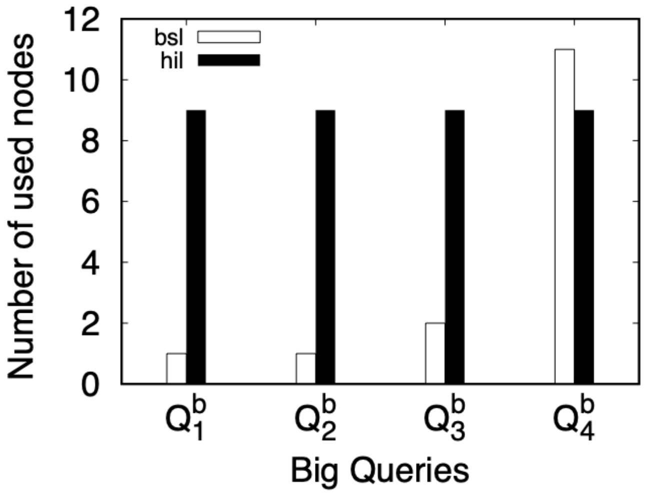
Zone usage

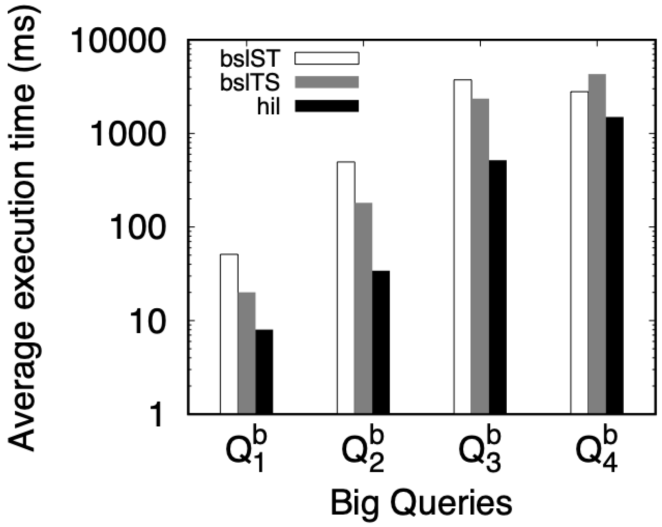




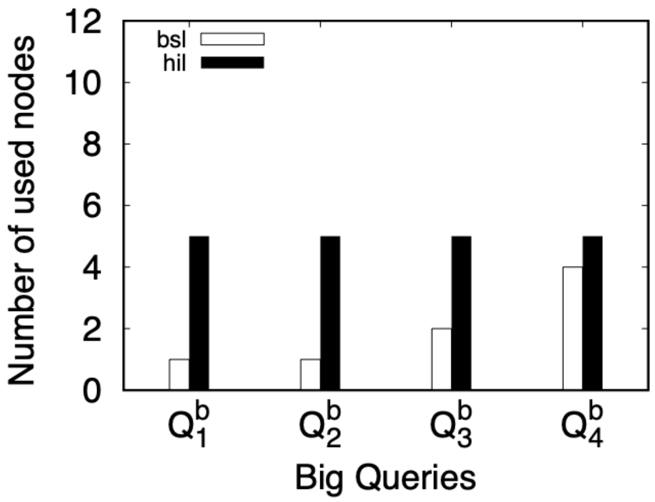
Experiments (4/5) - Performance of “spatially big” ST queries

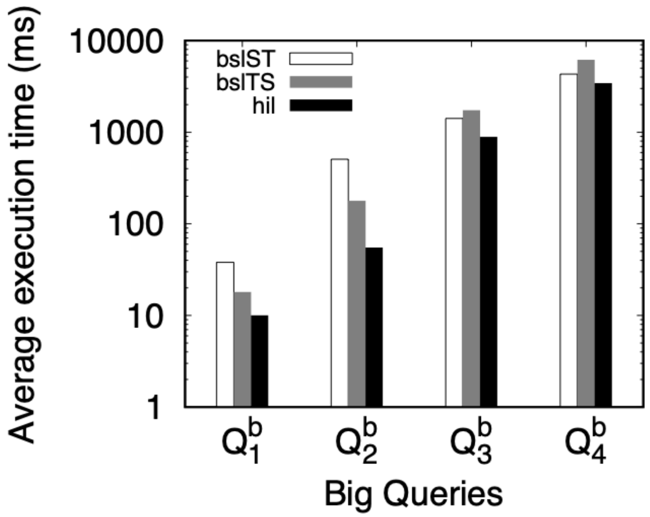
Default distribution





Zone usage

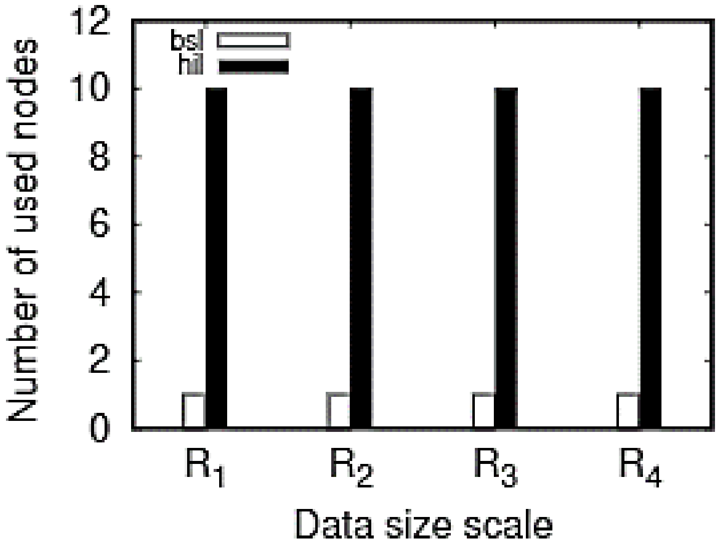


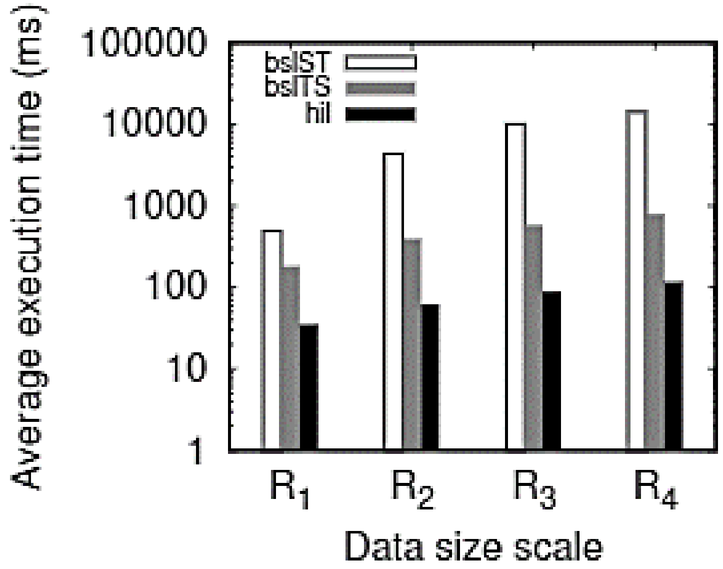


Experiments (5/5) - Scalability study

- We record the performance of $\boldsymbol{Q_2^b}$

	Real data set	
Size Factor	*Size (GB) in MongoDB*	*#points (M)*
x1 **(R_1)**	**~40**	**15.2**
x2**(R_2)**	**~83.87**	**31.4**
x3 **(R_3)**	**~127.21**	**47.7**
x4 **(R_4)**	**~171.39**	**63.9**





Conclusions

- We proposed a spatio-temporal approach for indexing and sharding on MongoDB
- The referred approaches were collated and evaluated on a MongoDB cluster
- Demonstrated the advantages of the proposed approach on specific queries
- Can be easily adopted as a solution on top of MongoDB

DATA STORIES

Thank you for your attention

More info:

our group: http://www.datastories.org/

project Track & Know: [*https://trackandknowproject.eu/*](https://trackandknowproject.eu/)

project Chorologos: [*https://www.ds.unipi.gr/chorologos/*](https://www.ds.unipi.gr/chorologos/)

e-mail: koutroumanis@unipi.gr

Track & Know

Chorologos

European Commission

The EU Framework Programme for Research and Innovation
HORIZON 2020

H.F.R.I.
Hellenic Foundation for Research & Innovation

GSRT
GENERAL SECRETARIAT FOR RESEARCH AND TECHNOLOGY