# SOROCA – SCALABLE OPEN SOURCE EARTH OBSERVATION CATALOGUE

*Roman Benčík (Iguassu), Miroslav Houdek (Iguassu), Jordi Farres (ESA), Michele Iapaolo (ESA)*

Iguassu Software System a.s., Evropska 120, 16 000 Praha 6, Czech Republic
(ESA subcontractor)

## ABSTRACT

Continuously growing volume of remote sensed images brings growing demands not only to storage databases that store this images, but also to (spatial) databases that store metadata (with spatial content too) about this data and to Catalogue server (service) as access point to this metadata as well. This high demands to Catalogue server and its storage elements can be fulfilled by scaling. This document describes horizontal scalable EO Catalogue (SOROCA) and consist from two main part which are namely - (OpenSearch) server and (MongoDB) storage element.

***Index Terms***— SOROCA, EO catalogue, MongoDB, no relational spatial database, parallel query execution, opensearch

## 1. INTRODUCTION

SOROCA is highly scalable open source earth observation (EO) catalogue. Has been developed based on European Space Agency (ESA) request to speed up and horizontally scale up, fast growing earth observation image catalogues. This catalogues usually store a meta-data of remotely sensed images in PostgreSQL database extended with PostGIS and this solution (relational SQL database) does not offer enough possibility for horizontal scaling (except e.g. STADO extension which hasn't been enough sufficient for our purpose). This paper describes solution using MongoDB (no relational) database as highly scalable and reliable storage solution and Pycsw server as a skeleton for SOROCA catalogue server developed in conformance with OpenSearch standard 10-032r3. Mongo database didn't included enough sufficient spatial functionality and due to this reason has been extended. Data in SOROCA catalog are divided to same size block and distributed to several nodes. During the query execution, all node including relevant data will work on answer to query in parallel.

## 2. INITIAL INVESTIGATION

During investigation of input data we found, they don't have 'ideal' spatial character (there exists images with so hudge footprint, when approx. 33 pcs of this footprints cover whole earth/globe and most of this geometries will have almost same bounding boxes. More often used R-tree spatial indexing uses bounding box filtering (Fig.1) to avoid no sensual computation (CPU usage).

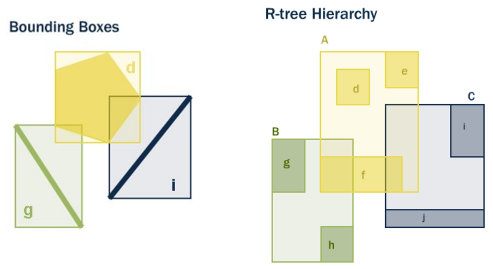

**Fig. 1.:** Bounding Box & R-TreeOne hierarchy

Since there exists a lot of footprints (of remote sensed images) which bounding boxes cover whole world/earth (Fig.2 and Fig. 3), sometimes of was better to not use this indexing at all. Of course it strongly related to count of returned results and spatial character of geometries. There is simple rule said, indexing is effective if less than half of all record is returned, in other cases sequential scan of whole database is faster.

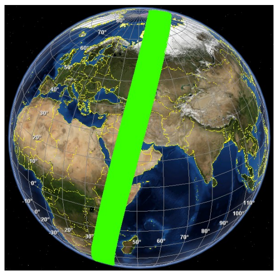

**Fig. 2.:** One image (green) footprint in 3D space

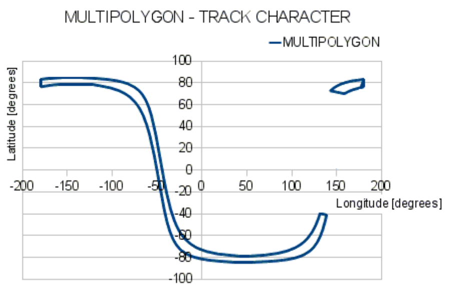

**Fig. 3.:** Image - Multipolygon footprint in 2D space

## 3. SCALING – DATA DISTRIBUTION

At beginning of this work we made some investigation of available storage solutions as a main part of EO catalogue service. We investigated possibility to scale existing PostgreSQL database with application STADO. This application offers parallel computing without replication. Data distribution to nodes was based on hashing of selected key – field value. MongoDB in contrast with, offered replication, data distribution balancing and two methods how to distribute data (first is hashing and second is range based distribution /what is basically opposite of first one/).

## 4. MONGO SHARDING

The basic concept behind MongoDB's sharding is to break up collections into smaller chunks. These chunks can be distributed across shards so that each shard is responsible for a subset of the total data set. We don't want our application to have to know what shard has what data, or even that our data is broken up across multiple shards, so we run a routing process called Mongos in front of the shards. This router knows where all of the data is located, so applications can connect to it and issue requests normally. As far as the application knows, it's connected to a normal Mongod. The router, knowing what data is on which shard, is able to forward the requests to the appropriate shard(s).

If there are responses to the request, the router collects them and sends them back to the application. In a sharded set-up, like shown Fig. 4, the client connects to a Mongos process, which abstracts the sharding away from the application. From the application's point of view, a sharded setup looks just like a nonsharded setup. There is no need to change application code when you need to scale.

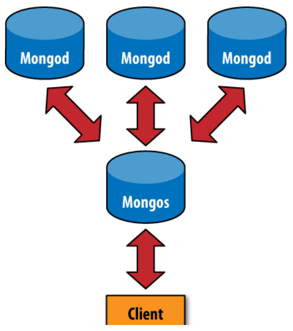

**Fig. 4.:** Mongo Sharding

## 5. SHARDED COLLECTION BALANCING

Balancing is the process MongoDB uses to distribute data of a sharded collection evenly across a sharded cluster. When a shard has too many of a sharded collection's chunks compared to other shards, MongoDB automatically balances the the chunks across the shards (Fig. 5). The balancing procedure for sharded clusters is entirely transparent to the user and application layer. [4]

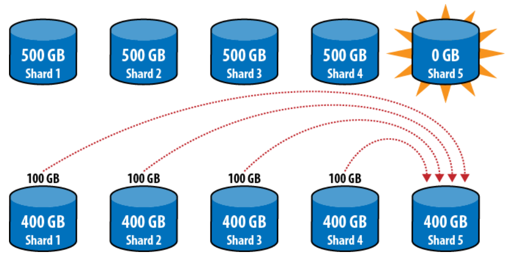

**Fig. 5.:** Balancing

## 6. CATALOG SERVER /SERVICE

Catalog Services has been developed in conformance of OpenSearch Specification 10-032r3 (part of future CSW 3.0). SOROCA catalogue has this characteristics:

Functionality (base):

-Full text search
-Bounding box search
-Arbitrary geometry search (using WKT)
-Spatial search using a point and a radius
-Get record by id
-Search by name
-Temporal search

Relational geo function:

-Contains
-Overlaps
-Disjoint

Geometries:

POINT
LINESTRING
POLYGON
MULTIPOINT
MULTILINESTRING
MULTIPOLYGON

.

## 9. CONCLUSION

The system has been developed, thorough testing and benchmarking is going to be performed in next months, as well as demonstration deployment in clouds and multiple nodes, preliminary results look promising. One of the greatest knowledge of this work is, that the most used R-Tree spatial indexing is not suited to all spatial data. Result of this work is Catalogue service able to select or disable filtering/indexing (bounding box filtering or grid index) with possibility to manage it separately for each different group of data (with different spatial character). Performance of this new features and type of filtering was not fully tested yet, but high scalability should offer sufficient performance without spatial indexing as well.

## 10. REFERENCES

[1] https://launchpad.net/stado

[2] http://www.mongodb.org/

[3] http://pycsw.org/

[4]http://docs.mongodb.org/v2.4/MongoDB-sharding-guide.pdf