Processing Large Raster and Vector Data in Apache Spark

Stefan Hagedorn[1], Timo Räth[1], Oliver Birli[2], Kai-Uwe Sattler[1]

Abstract: Spatial data processing frameworks in many cases are limited to vector data only. However, an important type of spatial data is raster data which is produced by sensors on satellites but also by high resolution cameras taking pictures of nano structures, such as chips on wafers. Often the raster data sets become large and need to be processed in parallel on a cluster environment. In this paper we demonstrate our STARK framework with its support for raster data and functionality to combine raster and vector data in filter and join operations. To save engineers from the burden of learning a programming language, queries can be formulated in SQL in a web interface. In the demonstration, users can use this web interface to inspect examples of raster data using our extended SQL queries on a Apache Spark cluster.

1 Introduction

An enormous number of spatial and spatio-temporal data objects is produced every second by thousands or even millions of sources. This data is represented as vector objects (i. e. a single point, a sequence of points that form a line string or polygon) or raster data. Raster data sets are created, e. g. by sensors and cameras where each pixel of the camera sensor is stored as a value in the data set.

Large raster data sets are produced by satellites observing the Earth and by high resolution cameras taking pictures of nano-sized structures – as performed in the Nano Positioning and Measurement Machines (NPMM)[3][Ba14; Ha02]. Depending on camera resolution and magnification, the NPMM200 at TU Ilmenau produces data sets up to 17 TB per object, which cannot be handled by single node DBMSs anymore.

For managing and processing raster data special purpose DBMSs have been proposed in the past, e. g. SciDB [Br10] or RasDaMan [Ba98]. Popular platforms for very large data sets are Hadoop MapReduce and Apache Spark. However, since their generic data model is not able to utilize characteristics of spatial objects (neighborhood, distances, etc.), extensions that add spatial *vector*-only data support have been proposed [EM15; YWS16]. Rasterframes[4] is the only system that supports raster data. The combination of raster and vector data, however, is limited to range queries only.

[1] Technische Universität Ilmenau, Databases & Information Systems, Ilmenau, first.last@tu-ilmenau.de

[2] Technische Universität Ilmenau, PMS, Ilmenau, first.last@tu-ilmenau.de

[3] `https://www.tu-ilmenau.de/cc-npmm/`

[4] `http://rasterframes.io/`

doi:10.18420/btw2019-43

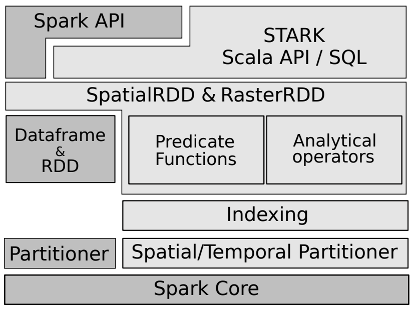


Fig. 1: STARK's architecture. All components use standard Spark APIs to integrate into the platform.

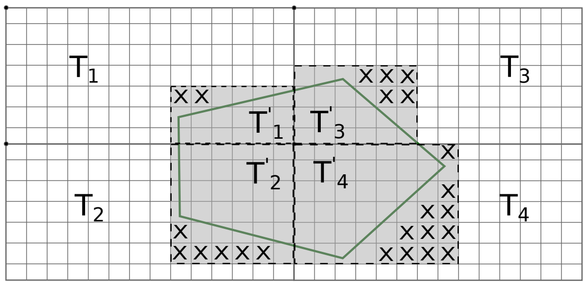


Fig. 2: Result tiles of a combination with a vector object – each with different dimensions. Pixels marked with `X` mean `NULL` or `NODATA`.

Typically, engineers need to transform, filter, and join their raster data sets with additional data. In the scenario of precision measurement at nano scale, a use case is to inspect the values (e. g. depth images of chips on a wafer, micromechanical structures, optical assemblies) to find defects in the components. This requires to apply transformations to the original raster tiles to correct skews. Furthermore, engineers need to manually navigate through the images by selecting areas (rectangles and polygons) of special interest to overlap them with other (reference) images and use it for further processing. Analysis on the images include measurements of geometries and inspection of material properties (white light interferometry or spectroscopic measurements). For this, appropriate functions need to be implemented.

In an example Earth observation use case, a raster data set with temperature measurements has to be joined with the vector data set of country borders, to calculate the average temperature value per country.

In this paper we demonstrate our STARK[5] framework [HGS17] along with a web interface to interactively explore raster and vector data using SQL and to visualize query results.

2 The STARK Framework

STARK is tightly integrated with Apache Spark by leveraging Scala language features so that users who are familiar with the RDD API can intuitively use STARK's functions. Besides the Scala API based on the core RDDs, STARK is integrated into SparkSQL and implements SQL functions to filter, join, and aggregate vector and raster data. The overall architecture is shown in Fig. 1. In its core STARK implements two specialized classes that extend Spark's RDD: `SpatialRDD` and `RasterRDD`. A `SpatialRDD` is used to store spatio-temporal vector data sets, while a `RasterRDD` represents a raster data set consisting of a set of tiles. A `Tile` type is a rectangle that stores the actual payload data, the pixels of the raster, in a generic array. In addition to the payload data, a `Tile` stores some meta data: the coordiantes of the upper left corner point as well as the width and height (number of pixels)

[5] `https://github.com/dbis-ilm/stark/`

of the represented rectangle and the resolution. Vector data objects are represented using a `STObject` class that stores the geometric object (point, polygon, etc.) as well as a temporal component.

A filter operator on a raster data set expects a `STObject` (or a plain rectangle) as query range. As a result, the operator finds all tiles that match the query range according to the provided predicate (intersects, contains, etc.) and returns them as a new `RasterRDD`. The explicit storage of the meta information (position, width, height) in each tile allows for a independent data parallel processing of all tiles. As shown in Fig. 2, a tile may only partially intersect with a polygonal query range and hence, the resulting tiles may be of different extents. This new information is saved into the resulting tiles T_1', T_2' etc. The rectangular result tiles of the filter operation represent the minimum bounding rectangle of the intersection with the query range and the input tile. Pixel values in the result tile that do not match the predicate, are set to `NULL` or a `NODATA` constant.

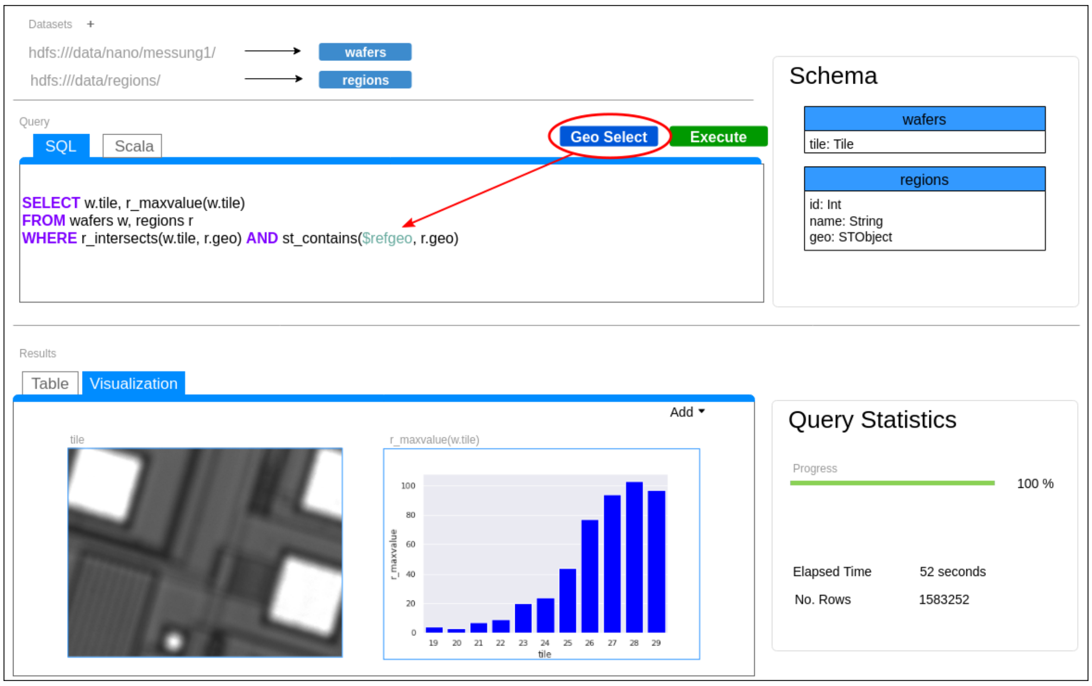
Besides the filter, `RasterRDDs` and `SpatialRDDs` can be joined with another `SpatialRDD`. Similar to the filter, the join operation finds all tiles (or `STObjects`) from the left input that have a join partner in the right input. In STARK, tiles are vectorized into rectangles using their stored meta information when their relation with instances of `STObject` has to be computed in filters and joins.

Operators are supported by spatial and temporal partitioners to assign data objects to worker nodes respecting their spatial and temporal neighborhood. Operators additionally benefit from a partition-local spatial or temporal indexing.

The web frontend is specifially designed to let users explore the raster and vector data sets with SQL. Users can define their relations by providing the path in the file system and the schema. The formulation of the spatial filter conditions is aided by a graphical selection tool that displays the spatial component (raster or vector) and lets users draw their region of interest . The SQL query is executed on a Spark cluster using our STARK framework. Results of the query can either be displayed in tabular form or as pictures (from raster tiles), on maps (vector objects), or charts. Users can select what and how attributes from the result should be visualized. Image rendering is performed by STARK (rather than by the browser) to account for large data sets.

3 Demonstration

We present our prototype designed to support scientists and engineers working with large spatial raster and vector data sets. A screenshot of the web frontend is shown in Fig. 3. During the demonstration users can interact with the web application: We will prepare real high resolution raster data sets caputered by the NPMM as well as satellites images. Users can run queries (our predefined or their own ones) that show the various features of the STARK framework itself as well as the capabilities of the web interface. The queries



hdfs:///data/nano/messung1/ → wafers
hdfs:///data/regions/ → regions

```
SELECT w.tile, r_maxvalue(w.tile)
FROM wafers w, regions r
WHERE r_intersects(w.tile, r.geo) AND st_contains($refgeo, r.geo)
```

Schema

wafers
tile: Tile

regions
id: Int
name: String
geo: STObject

Query Statistics

Progress 100 %

Elapsed Time 52 seconds
No. Rows 1583252

Fig. 3: The user interface for querying raster data with SQL. With `Geo Select` users can use a graphical tool to select a region of interest, referred to in the `$refgeo` variable.

demonstrate STARK's unique feature of combined processing of raster and vector data on the Spark platform whereas the web interface lowers the entrace barrier of creating and submitting jobs to a cluster. Furthermore, users can choose from a list of available charts how results should be visualized and dynamically add them to the web page.

References

[Ba14] Balzer, G. F.: Entwicklung und Untersuchungen zur 3-D-Nanopositioniertechnik in großen Bewegungsbereichen, PhD thesis, TU Ilmenau, 2014.

[Ba98] Baumann, P.; Dehmel, A.; Furtado, P.; Ritsch, R.; Widmann, N.: The multidimensional database system RasDaMan. In: SIGMOD Rec. 1998.

[Br10] Brown, P. G.: Overview of sciDB: Large Scale Array Storage, Processing and Analysis. In: SIGMOD. Pp. 963–968, 2010.

[EM15] Eldawy, A.; Mokbel, M. F.: SpatialHadoop: A MapReduce Framework for Spatial Data. In: ICDE. Seoul, 2015.

[Ha02] Hausotte, T.: Nanopositionier- und Nanomessmaschine, PhD thesis, TU Ilmenau, 2002.

[HGS17] Hagedorn, S.; Goetze, P.; Sattler, K.-U.: The STARK Framework for Spatio-Temporal Data Analytics on Spark. In: BTW. Pp. 123–142, 2017.

[YWS16] Yu, J.; Wu, J.; Sarwat, M.: A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In: ICDE. Pp. 1410–1413, 2016.