Ice Monitoring With ExtremeEarth

George Mandilaras, Manolis Koubarakis

National and Kapodistrian University of Athens
{gmandi, koubarak}@di.uoa.gr

Abstract. Monitoring the trajectories of icebergs is a crucial task for the safety of ship traffic in the arctic circle. In this work, we present a system that addresses this problem by combining Earth observation techniques with large scale RDF analytics. To the best of our knowledge, this is the first Semantic Web application in this field.

1 Introduction

The safety of ship navigation in the arctic relies on the crucial and continuous task of monitoring icebergs. This task has recently become more critical, due to the climate change, which has significantly increased the number of icebergs on the sea. To address it, automatic techniques are now used to combine in-situ observational data with satellite images.

In more detail, the *Ice Watch project*[1] of the Norwegian Meteorological Institute collects data from ships performing visual sea ice observations while navigating the arctic. This in-situ observational data record the time, point locations, and other important properties of sea ice, which are then used in order to create sea ice charts. However, they require time consuming expert analysis and, thus, they are updated less frequently than desired and can be out-of-date by the time of issue.

Another valuable source of information is offered by various Earth observation programmes, such as the *US Landsat program*[2] and the *EU Copernicus Programme*[3]. Copernicus is actually the largest relevant programme at the moment, providing users with reliable and up-to-date information on a range of environmental and security issues under a free, full and open data policy. By 2030, it is expected to set 20 satellites, called *Sentinels*, in orbit.

In this work, we are interested in the set of satellite images that cover the arctic and chronologically coincide with the in-situ observations. Our goal is to interlink these two data sources, so as to identify in-situ observations that match closely in time and space to satellite images. The end result is useful for building training sets with satellite images associated with high quality ground observations and essential for the construction of a robust automatically derived mapping product that could be updated frequently. Most importantly, by

[1] https://icewatch.met.no/
[2] https://www.usgs.gov/land-resources/nli/landsat
[3] https://www.copernicus.eu/en

expressing the end result as RDF statements, we can leverage a wealth of Semantic Web tools for performing analytics, reasoning as well as visualization. A crucial aspect in this process is the time efficiency, as the large volume of data calls for scalable techniques.

More specifically, we have developed an open-source system that implements the end-to-end workflow depicted in Figure 1. First, it performs a spatiotemporal join between the two data sources using STARK [3], a framework built on top of Apache Spark[4] for massive parallelization. Then, it converts the interlinked items into RDF triples using GeoTriples [7] along with a domain-specific ontology. These triples are stored in a Strabon endpoint [6] and are visualized with Sextant [8] using GeoSPARQL queries.

We describe our approach in more detail below. We delve into the data sources we use in Section 2, while Section 3 discusses the components of our system. We present a preliminary experimental study in Section 4 and conclude the paper in Section 5 along with directions for future work.

2 Data Sources

The system takes as input three data sets: (i) a set containing observation points, (ii) a set that containing ice observation data, and (iii) a set containing information about the satellite images.

The observation point set contains the time and the location of each observation. The ice observation set contains detailed information about the ice (such as ice thickness, snow thickness, floe size, etc), which were recorded by the observer, along with an observation id that links back to the observation. The ice observation set contains more than 20,000 observations, however since it is only updated when a cruise has been out in the arctic and uploaded its data, there are large gaps with no observations. These sets are provided by the *Ice Watch project*.

The set of satellite images contains information about images captured by Sentinel-1[5] and is provided by the Copernicus project. This information includes the location of the image, the datetime (time the image was acquired by the satellite), the coverage (geographic extents of the images) as well as some other useful properties. Note that the data source of satellite images is updated on a daily basis and contains more than 200K images.

3 Approach

Our system workflow consists of four main steps:

1) We perform a spatiotemporal join in order to interlink our data sets, using STARK. **STARK** [3] is an open-source framework[6] that extends Apache Spark

[4] `https://spark.apache.org/`

[5] `https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-1`

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

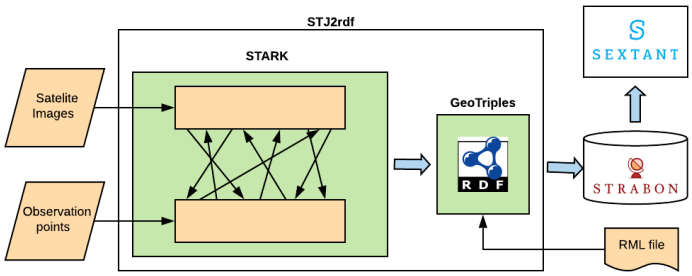

Fig. 1. The end-to-end workflow implemented by our system.

with spatiotemporal functionalities, by introducing new classes and spatiotemporal operators to standard RDDs. As a result, users are able to use the main functionalities of Spark and to perform spatiotemporal transformations to their data. Furthermore, the STARK framework supports spatial partitioning and indexing to achieve better data distribution and enables the user to perform join queries based on spatiotemporal relations like CONTAINS and INTERSECTS. STARK is probably the only framework that extends Spark, not only with spatial but also with temporal operations, and it is considered one of the fastest big spatial data processing frameworks [4].

In our case, we wanted to find which satellite images captured the ice observation points at the same day that the observations occurred. Consequently, the temporal dimension has a deterministic role as we are examining ice thickness which can alter from days to days. Therefore, we use STARK to perform a spatiotemporal join using the CONTAINS relation. This way, we link the images to the observation points that are spatially contained in the coverage of the image and also their timestamp is within a range of a day from the time that the image was shot.

2) We transform the results into RDF triples using GeoTriples. **GeoTriples** [7] is an open-source system that transforms geospatial data from a wide variety of original data formats into RDF.

As input, GeoTriples requires a mapping file, which contains the mapping rules that will be applied in the transformation, and also information about the data. Hence, it offers a mapping generation procedure, in which it parses the input data and then produces the mapping file, which then the user may edit in order to fit his needs. Given, though, that we already know the form of the data and the mapping rules that need to be applied, , we have pre-constructed this mapping file and forwarded it to GeoTriples.

Additionally, in this mapping file the user also defines the ontology that the produced triples follow. For the purpose of the project ExtremeEarth, we have developed an ontology based on the Sea Ice GeoReferenced Information and Data - SIGRID-3 [5], which describes a set of standards to code, exchange and archive digital ice charts. The central class of the ontology is the IceObservation class

which indicates the generic ontology for an ice observation. Using the property polar:hasForm we describe the form of an ice observation as Iceberg, Pancake-Ice, FastIce, etc, and the property polar:hasDevelopmentStage to describe its development stage, which is determined by the thickness of the ice and it is represented by certain classes. Therefore, the mapping file is adjusted so that the produced triples follow this specific ontology. Then, regarding the transformation, GeoTriples extracts the mapping rules from the mapping file and uses them to convert the input data into RDF. The generated triples are stored in the NTRIPLES format. We have automated the procedure of forwarding the results from STARK to GeoTriples, by combining these two tools into one, which we called STJ2rdf.

3) We store the produced triples into the spatiotemporal RDF store **Strabon** [6], a state-of-the-art open-source spatiotemporal triplestore that efficiently executes GeoSPARQL and stSPARQL queries. Strabon supports spatial datatypes, enabling the serialization of geometric objects in the OGC standards WKT and Geography Markup Language (GML). It has been implemented by extending the established RDF store Sesame (now called RDF4J[7]), using the spatially-enabled database PostGIS[8] as back-end so as to exploit its large variety of spatial functions and operators. It has been experimentally shown that Strabon is the most efficient spatiotemporal RDF store available today [1, 2].

4) The end result is visualized via **Sextant** [8], as shown in Figure 2. Sextant constitutes a web-based application for exploring, interacting and visualizing time-evolving linked geospatial data. Sextant is also capable of creating, sharing, searching and collaboratively editing maps and of producing statistical charts out of statistically enhanced data sets. Even though it relies heavily on Semantic Web technologies, it offers an intuitive interface that allows both domain experts and lay users to exploit all available features.

4 Experiments

The implementation of our system is publicly available as an open-source project[9]. We experimentally assessed its performance on a server with Intel Xeon E5-4603 v2 (2.2GHz, 16 physical cores), 128 GB RAM, running Ubuntu 14.04.5 LTS.

Figure 3 depicts the evolution of the wall-clock time that is required by the spatiotemporal join of our system, as the number of available cores increases. We observe that even with the minimum number of cores, less than 3 minutes are required for joining >20,000 ice observations with ∼200,000 satellite images. Most importantly, the running time decreases to a half, when doubling the minimum number cores. However, there is no significant improvement when using 8 or more cores. This should be attributed to the expensive data shuffles that are performed when splitting the input data into more partitions.

[7] `http://rdf4j.org`
[8] `https://postgis.net`
[9] `https://github.com/GiorgosMandi/SpatioTemporal-Join`

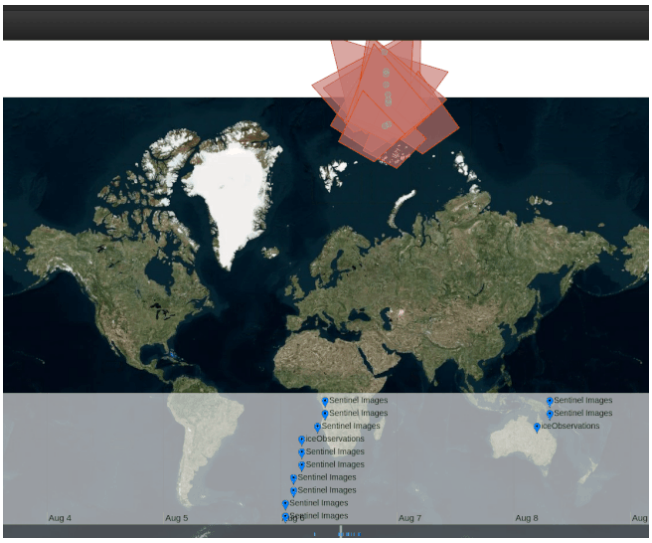

Fig. 2. The results of our system as they are presented to the user.

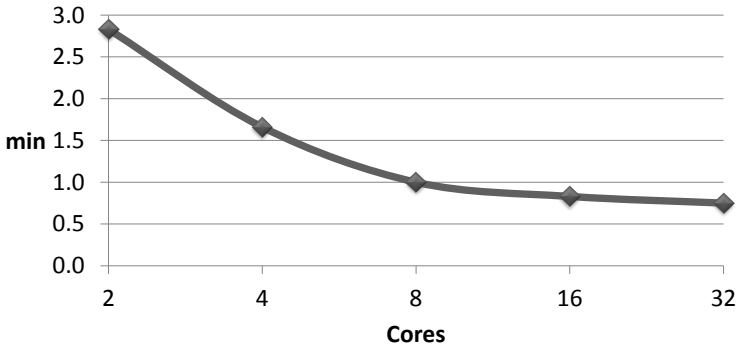

Fig. 3. The wall-clock time of the spatiotemporal join as the number of core increases.

5 Conclusions

We have presented a novel system for monitoring the icebergs in the arctic circle. For high time efficiency, it relies on a spatiotemporal join operation that is carried out on top of Apache Spark. To make the most of the detected data associations, it relies on Semantic Web technologies for querying and visualization.

This system was developed as part of the ExtremeEarth project[10], which focuses on Artificial Intelligence and Big Data technologies that scale to the petabytes of big Copernicus data. ExtremeEarth applies these technologies in two of the thematic exploitation platforms of the European Space Agency: one dedicated to Food Security and one dedicated to the Polar regions. Its goal is to

[10] `http://earthanalytics.eu`

develop techniques and software that will enable the extraction of information and knowledge from big Copernicus data using deep learning techniques and extreme geospatial analytics, making this information and knowledge available as linked data.

In the future, we plan to improve the scalability of the spatiotemporal join, reducing the data shuffles that are performed when Apache Spark uses more cores. We also plan to integrate more Semantic Web technologies into our system, such as a semantic reasoner, so as to provide users with more useful information.

Acknowledgements. This work has been partially supported by the EU project ExtremeEarth (contract no. 825258).

References

1. Bereta, K., Smeros, P., Koubarakis, M.: Representation and querying of valid time of triples in linked geospatial data. In: ESWC. pp. 259–274 (2013)
2. Garbis, G., Kyzirakos, K., Koubarakis, M.: Geographica: A benchmark for geospatial RDF stores (long version). In: ISWC. pp. 343–359 (2013)
3. Hagedorn, S., Götze, P., Sattler, K.: The STARK framework for spatio-temporal data analytics on spark. In: Datenbanksysteme für Business, Technologie und Web (BTW). vol. P-265, pp. 123–142. GI (2017)
4. Hagedorn, S., Götze, P., Sattler, K.U.: Big spatial data processing frameworks: Feature and performance evaluation pp. 490–493 (2017)
5. JCOMM Expert Team on Sea Ice: Sigrid-3: A vector archive format for sea ice georeferenced information and data (2014)
6. Kyzirakos, K., Karpathiotakis, M., Bereta, K., Garbis, G., Nikolaou, C., Smeros, P., Giannakopoulou, S., Dogani, K., Koubarakis, M.: The spatiotemporal RDF store strabon. In: Advances in Spatial and Temporal Databases - 13th International Symposium (SSTD). vol. 8098, pp. 496–500 (2013)
7. Kyzirakos, K., Savva, D., Vlachopoulos, I., Vasileiou, A., Karalis, N., Koubarakis, M., Manegold, S.: Geotriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings. J. Web Semant. **52-53**, 16–32 (2018)
8. Nikolaou, C., Dogani, K., Bereta, K., Garbis, G., Karpathiotakis, M., Kyzirakos, K., Koubarakis, M.: Sextant: Visualizing time-evolving linked geospatial data. J. Web Semant. **35**, 35–52 (2015)