# The SPADES Framework for Scalable Management of Spatio-textual Data

AKRIVI VLACHOU, Dept.of Inf. & Com.Syst.Engineering
University of Aegean, Greece
CHRISTOS DOULKERIDIS, Dept. of Digital Systems
University of Piraeus, Greece
NIKOLAOS KOUTROUMANIS, Dept. of Digital Systems
University of Piraeus, Greece
DIMITRIS POULOPOULOS, Dept. of Digital Systems
University of Piraeus, Greece
KJETIL NØRVÅG, Dept. of Computer Science
Norwegian University of Science and Technology, Norway

This paper presents the research activities in the context of the SPADES project for scalable indexing and processing of big spatial and spatio-textual data. Management of spatio-textual data raises challenges due to the high dimensional nature of text, in combination with the problem of preserving data locality for spatial data. In this paper, we provide an overview of our contributions in this field, both for centralized and parallel processing. We present an indexing layer that supports spatio-textual data, as well as other data types: spatio-temporal, multidimensional as well as semantic data represented in RDF. This is coupled with a processing layer that encompasses algorithms both for centralized and parallel processing.

CCS Concepts: • **Information systems → Data management systems**;

Additional Key Words and Phrases: Spatio-textual data, big data, indexing, processing

**ACM Reference Format:**
Akrivi Vlachou, Christos Doulkeridis, Nikolaos Koutroumanis, Dimitris Poulopoulos, and Kjetil Nørvåg. 2020. The SPADES Framework for Scalable Management of Spatio-textual Data. In *PCI '20: Panhellenic Conference on Informatics, November 20–22, 2020, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3437120.3437293

## 1 INTRODUCTION

An ever-increasing amount of data that combine spatial/spatio-temporal with textual information is generated daily at unprecedented rates, as a result of social networking, GPS-enabled devices, and location-based search. This wealth of information has motivated the development of applications and services that deliver personalized results to end-users based on complex search criteria that combine textual relevance with spatial proximity. In consequence, there is a need for efficient access and scalable processing of data that is characterized by multiple dimensions: space, time and text.

SPADES is a research project that identifies a set of research and technological challenges that need to be effectively addressed, in order to support spatio-textual and spatio-temporal queries over



*PCI '20, November 20–22, 2020, Athens, Greece*

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/3437120.3437293

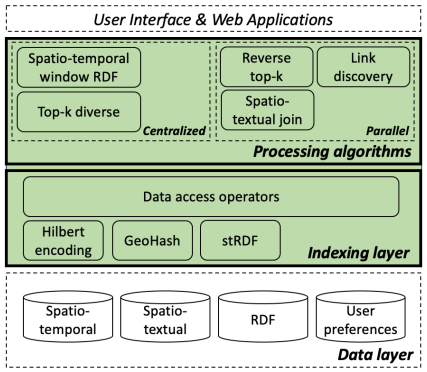



Fig. 1. The SPADES architecture at high-level.

massive data. Essentially, SPADES targets advanced spatio-textual query types, new distributed indexing methods that are applicable for scalable storage of spatio-textual data, abstractions for unified data access, as well as more efficient query processing.

Figure 1 shows the architecture of SPADES, which consists of an indexing layer and a processing layer. At the bottom, different data sources are depicted, containing spatio-temporal or spatio-textual data, which are often available in semantically enriched representations (e.g., using RDF), as well as multidimensional data, such as user preferences.

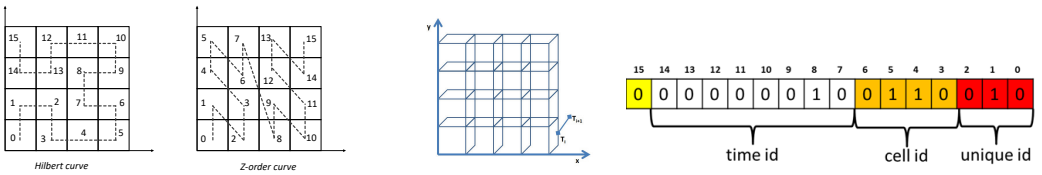
The indexing layer (Section 3) provides techniques for encoding the spatial, spatio-temporal and spatio-textual information in one-dimensional (1D) values, which can be indexed by standard access methods (e.g., B-trees) that are provided by any database system (relational or NoSQL). In the processing layer (Section 4), some of the centralized algorithms (spatio-temporal window RDF) adopt and exploit the offerings of the indexing layer, others (top-$k$ diverse) have been implemented on top of existing – or extended – data structures, such as the R-tree. The parallel algorithms focus on scalable processing over large data sets, such as associating objects from different data sets based on spatio-temporal or spatio-textual criteria (link discovery, using variants of spatial joins), as well as ranked retrieval of multidimensional objects based on user preferences (reverse top-$k$).

## 2 RELATED WORK

Our work mainly relates to scalable indexing and processing of spatio-textual and spatio-temporal data. Existing approaches for spatio-textual indexing can be classified in the following categories: spatial-first, text-first, and interleaved. We refer to [10] for an in-depth overview of the related research. Lately, several research projects have extended popular parallel data processing platforms, such as Hadoop or Spark, in order to provide customized solutions for big spatial or spatio-temporal data. The most prominent prototypes and systems in this field include Hadoop-GIS [1], Parallel SECONDO [8], SpatialHadoop [5], ST-Hadoop [2], SpatialSpark [20], GeoSpark [21], LocationSpark [16], Simba [19]. We also refer to [6] for a comparative evaluation of big spatial data processing systems. Also, we refer to [3, 4] for an overview of query processing algorithms for spatio-textual data.

## 3 INDEXING LAYER

The indexing layer of SPADES relies on a variety on mapping techniques that aim to transform spatial, spatio-temporal and spatio-textual data to one-dimensional (1D) values, which can be efficiently indexed using standard techniques, such as B-trees. In addition, one-dimensional mappings fit nicely with scalable NoSQL stores, which typically rely on key-based access to data.




(a) Hilbert and Z-order curve. (b) 3D partitioning. (c) Bit representation of the identifier (ID).

Fig. 2. Examples of one-dimensional mappings.

### 3.1 One-dimensional Mappings

*3.1.1 Spatial Data.* In the case of spatial data, we use different mappings based on space-filling curves, for example based on the Hilbert curve, the z-order curve and GeoHashes[1]. Figure 2a shows an example of ordering the cells of the 2D spatial domain according to the Hilbert curve (left) and the z-order (right). The aim is to preserve spatial locality in the 1D values, to the extent possible. In SPADES, we exploit and extend such one-dimensional mappings for different purposes. In brief, the Hilbert curve has been shown to have nice clustering properties, whereas the z-order is much simpler to implement and generalize to higher dimensions.

*3.1.2 Spatio-temporal Data.* In the case of spatio-temporal data, we propose an extension that supports one-dimensional mapping of dynamic data, i.e., data with new timestamps that arrive as time progresses. Moreover, we generate a *unique* identifier (IDs) for each object, which is often necessary for storage in key-based systems (e.g., HBase, Redis, etc.).

Consider a 2D regular spatial grid that consists of $2^m$ equi-sized cells. To handle the temporal dimension, we use a sequence of such grids, each associated with a temporal partition $T_i$ (see Figure 2b). Each spatio-temporal object is mapped to an ID consisting of $b$ bits (typically $b = 64$). We set the most-significant bit to 0 for all IDs of spatio-temporal objects, to separate them from other objects (not spatio-temporal). We also keep $m$ bits to represent the different $2^m$ spatial grid cells. Then, in each 3D cell, we reserve $k$ bits for an auto-incremented ID, encoded in the rightmost bits. The remaining $b - (m + k + 1)$ bits are used for encoding the time. In the example of Figure 2c, we have $b$=16, $m$=4, and $k$=3, and the depicted identifier is $2^8 + 2^5 + 2^4 + 2 = 306$. The spatial cell in which it belongs is 6 (=0110), and the spatial grid contains $2^4 = 16$ cells in total. This encoding can accommodate $2^{b-(m+k+1)} = 2^8 = 256$ temporal partitions. We have used this idea for indexing spatio-temporal RDF data (stRDF) in centralized [17] and distributed environments [14], however the mapping mechanism is generic and can be readily applied for non-RDF data.

*3.1.3 Spatio-textual Data.* A spatio-textual object $o$ consists of a spatial location $o.l$ and a set of keywords $o.k = \{k_1, k_2, \ldots, k_n\}$. In this work, we consider point data, so the location of a spatio-textual object is a single point in the 2D geographical space. Our approach maps a spatio-textual data object to one or more one-dimensional (1D) keys, which are used for indexing. Locality-preserving 1D mappings have been extensively used in spatial databases (e.g., using space-filling curves), but also in NoSQL context. In order to include the textual information, our approach is to concatenate the 1D value produced by the spatial information with each keyword of the spatio-textual object. For instance, if a spatio-textual object $o$ contains three keywords $\{k_1, k_2, k_3\}$, we generate three keys for indexing this object; each key consists of the 1D value produced by $o.l$ and a keyword $k_i$ where $i \in [1, 3]$. Notice that a similar approach has been followed by ST-HBase [9].

[1]A technique proposed by G. Niemeyer in 2008 that bears similarities with z-order, and has been adopted by many NoSQL stores, e.g., MongoDB, Elastic, Redis, etc.

### 3.2 Data Access Operators

In order to support scalable access to spatial and spatio-textual data, we choose a NoSQL solution at the storage layer, which exploits the one-dimensional mappings. However, no single NoSQL store fits bets for all data models and use-cases, and each store uses a different, non-standardized API. Consequently, this hinders the adoption of different stores and makes the development of big data applications cumbersome.

Our solution to this problem is the proposal of a unified API, called NoDA [7], that consists of basic data access operators (e.g., filter, project, aggregate, etc.), which can be implemented for different NoSQL stores, thus offering a single view to big data developers. NoDA abstracts the individual details and peculiarities of the APIs of NoSQL stores, and offers a simple, unified interface to developers, which consists of familiar data operators. In turn, this offers many advantages: (a) the programming API makes application development much easier, since developers need to learn a single API (instead of different APIs), (b) it permits changing the choice of storage during application development, without affecting the existing code base of the application, (c) it can be coupled with an SQL-like interface that allows declarative querying of the underlying NoSQL stores.

## 4 QUERY PROCESSING

In this section, we present the query processing algorithms developed in the context of SPADES, both for centralized (Section 4.1) as well as for parallel (Section 4.2) environments.

### 4.1 Centralized Algorithms

*4.1.1 Spatio-temporal Window (StW) RDF Queries.* In [17], we have introduced query processing algorithms for efficient processing of spatio-temporal RDF data, by exploiting the 1D mapping presented in Section 3.1.2. We focus on a specific class of queries, called spatio-temporal window (StW) RDF queries, which covers many requirements and use-cases. Informally, such a query retrieves spatio-temporal entities (expressed in RDF) based on a combination of spatio-temporal filter and a graph pattern. Processing a StW query is challenging because it is difficult to combine the two constraints during data access. Our solution capitalizes on the 1D mapping presented in Section 3.1.2. RDF engines typically store triples (subject, predicate, object) encoded, by substituting strings with integer values. This offers advantages for compression and more efficient indexing. Our proposed mapping targets specifically spatio-temporal entities, thus allowing to transform the two query constraints to a graph pattern query that can be handled by the query engine in a uniform manner.

*4.1.2 Ranked Diversity-based Retrieval of POIs.* Retrieval of Points-of-Interest (POIs) based on distance and popularity is an interesting research topic, due to the advent of location-based social networks (LBSNs) that allow users to share their current geographic location with a "check-in". Existing approaches rank POIs based on a weighted sum of their popularity and proximity. However, this approach often produces homogeneous results, i.e., POIs that are frequented by the same people. Instead, we wish to obtain POIs with diversity, i.e., POIs attracted by many different people. In technical terms, our interest is not in maximizing the sum of check-ins, but the number of unique check-ins. Motivated by this, we introduce a new way to define diversity based on check-ins, namely in terms of set union, rather than count. Formally, we aim to retrieve the $k$ POIs $P' = \{p_1, \ldots, p_k\}$ that maximize a weighted sum of spatial proximity and diverse appeal [11]:

$$f(q) = \alpha \cdot \pi(q, P')/k + (1 - \alpha) \cdot S(P')$$

where $\alpha$ is a weight, $S(P')$ denotes diverse appeal as the percentage of distinct users (out of all users) that have visited at least one $p \in P'$, and $\pi(q, P') = \sum_{p \in P'} 1 - \delta(q, p_i)/D$, with $\delta()$ the

distance function and $D$ the maximum distance of any pair of POIs. Interestingly, when $\alpha = 1$ the problem is reduced to the classical nearest-neighbor problem, and when $\alpha = 0$ it is reduced to the NP-Hard maximum coverage problem. In [11], we have proposed several algorithms for solving the problem, including greedy 1-pass and $k$-pass algorithms, linear programming, and algorithms based on R-tree extensions.

### 4.2 Parallel Algorithms

*4.2.1 Parallel Joins.* The join operator is fundamental in database systems in order to retrieve records from different tables based on some matching condition. In the case of spatial data, the matching condition is of spatial nature, typically reflecting a topological or a proximity-based relationship. We have applied parallel join techniques for addressing the problem of link discovery in spatio-temporal data expressed in RDF. The proposed approach stLD [15] aims at identifying spatio-temporal relationships between spatio-temporal RDF entities, in order to interlink the entities. Moreover, the data sets involved can be static or stream-based. Examples of spatio-temporal relationships in real-life applications include finding the fishing vessels that have entered a protected area, finding vessels that approach ports, or identifying the sectors crossed by a given aircraft.

At the heart of this problem lies a spatial join algorithm. To achieve scalable processing for input streams of high rate or massive static data sets, we have built a data-parallel implementation of our framework in Apache Flink, which provides a high-throughput, low-latency streaming engine that fits our requirements. We have implemented two join variants; when one of the data sets is static and relatively small in size, we use a *broadcast join*, whereas when both data sets are stream-based or too large to be broadcast, we use a *repartition join*.

*4.2.2 Parallel Spatio-textual Joins.* In the case of spatio-textual data, each object is described by a spatial component (e.g., point, polygon, polyline) and a textual description, typically modelled as a set of keywords. The join operator retrieves pairs of spatio-textual objects whose distance is below a user-defined threshold and their textual descriptions are similar based on some set similarity function. Our work targets parallel processing and particularly non-uniform data distributions, where the challenge is to partition the data to nodes in a way that enforces load balancing. Even though this part of our work is ongoing, our preliminary research indicates that there exists a trade-off between fine-grained partitioning (that enables load balancing) and object duplication to neighboring partitions (that is required in order to process each partition independently).

*4.2.3 Reverse top-k Queries.* Given a database of objects $S$, a set of user preferences $W$, and a query object $q$, the reverse top-$k$ query [18] returns the subset of user preferences for which $q$ belongs to the top-$k$ results. From a modeling perspective, an object $p \in S$ is represented as an $n$-dimensional point, whereas the preferences of a user are represented by a vector $w \in W$ which assigns weights to the $n$ dimensions. Then, the score of $p$ for user $w$ is the dot product: $f_w(p) = p \cdot w = \sum_{i=1}^{n} p[i]w[i]$, and the top-$k$ objects for this user are the ones with best scores. Then, the reverse top-$k$ query takes as input an (existing or new) object $q$ and returns the subset of users that have $q$ in their top-$k$ result. However, processing a reverse top-$k$ query is a costly operation for centralized algorithms when applied on very large data sets.

To this end, we propose parallel algorithms for processing reverse top-$k$ queries [12, 13]. Our first finding is that the problem can be trivially parallelized, if data set $S$ is replicated to all worker nodes. However, for really large data sets, this may be prohibitively expensive. Thus, we introduce different techniques to partition both the data set $S$ and the user preferences $W$, while guaranteeing correctness and pruning unnecessary objects during query processing. Our algorithms are implemented in Apache Hadoop and we demonstrate results over TB-sized data sets in [13].

## 5 CONCLUSIONS AND OUTLOOK

In this paper, we have presented an overview of results of SPADES, a research project targeting scalable management and exploration of spatial and spatio-textual data. Our individual research activities in the context of SPADES have (so far) produced published results [7, 11–13, 15, 17]. As for future work, several research directions can be identified both at indexing and processing layers. Indicative directions include joint management of the temporal dimension together with space and text, as well as extensions towards semantic text retrieval instead of exact keyword matching.

## ACKNOWLEDGMENTS

This project has received funding from the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under grant agreement No 1667.

## REFERENCES

[1] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel H. Saltz. 2013. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *PVLDB* 6, 11 (2013), 1009–1020.

[2] Louai Alarabi, Mohamed F. Mokbel, and Mashaal Musleh. 2017. ST-Hadoop: A MapReduce Framework for Spatio-Temporal Data. In *Proc. of SSTD*. 84–104.

[3] Lisi Chen, Gao Cong, Christian S. Jensen, and Dingming Wu. 2013. Spatial Keyword Query Processing: An Experimental Evaluation. *PVLDB* 6, 3 (2013), 217–228.

[4] Lisi Chen, Shuo Shang, Chengcheng Yang, and Jing Li. 2019. Spatial keyword search: A survey. *Geoinformatica* (2019).

[5] Ahmed Eldawy and Mohamed F. Mokbel. 2015. SpatialHadoop: A MapReduce framework for spatial data. In *Proc. of ICDE*. 1352–1363.

[6] Stefan Hagedorn, Philipp Götze, and Kai-Uwe Sattler. 2017. Big Spatial Data Processing Frameworks: Feature and Performance Evaluation. In *Proc. of EDBT*. 490–493.

[7] Nikolaos Koutroumanis, Panagiotis Nikitopoulos, Akrivi Vlachou, and Christos Doulkeridis. 2019. NoDA: Unified NoSQL Data Access Operators for Mobility Data. In *Proc. of SSTD*. 174–177.

[8] Jiamin Lu and Ralf Hartmut Güting. 2013. Parallel SECONDO: Practical and efficient mobility data processing in the cloud. In *Proc. of IEEE Big Data*. 17–25.

[9] Youzhong Ma, Yu Zhang, and Xiaofeng Meng. 2013. ST-HBase: A Scalable Data Management System for Massive Geo-tagged Objects. In *Proc. of WAIM*. 155–166.

[10] Ahmed R. Mahmood and Walid G. Aref. 2019. *Scalable Processing of Spatial-Keyword Queries*. Morgan & Claypool Publishers.

[11] Stella Maropaki, Sean Chester, Christos Doulkeridis, and Kjetil Nørvåg. 2020. Diversifying Top-k Point-of-Interest Queries via Collective Social Reach. In *Proc. of CIKM*.

[12] Panagiotis Nikitopoulos, Georgios A. Sfyris, Akrivi Vlachou, Christos Doulkeridis, and Orestis Telelis. 2019. Parallel and Distributed Processing of Reverse Top-k Queries. In *Proc. of ICDE*. 1586–1589.

[13] Panagiotis Nikitopoulos, Georgios A. Sfyris, Akrivi Vlachou, Christos Doulkeridis, and Orestis Telelis. 2020. Pruning techniques for parallel processing of reverse top-k queries. *Distributed and Parallel Databases (to appear)* (2020).

[14] Panagiotis Nikitopoulos, Akrivi Vlachou, Christos Doulkeridis, and George A. Vouros. 2018. DiStRDF: Distributed Spatio-temporal RDF Queries on Spark. In *Proc. of EDBT workshops*. 125–132.

[15] Georgios M. Santipantakis, Apostolos Glenis, Christos Doulkeridis, Akrivi Vlachou, and George A. Vouros. 2019. stLD: towards a spatio-temporal link discovery framework. In *Proc. of SBD*. 4:1–4:6.

[16] MingJie Tang, Yongyang Yu, Qutaibah M. Malluhi, Mourad Ouzzani, and Walid G. Aref. 2016. LocationSpark: A Distributed In-Memory Data Management System for Big Spatial Data. *PVLDB* 9, 13 (2016), 1565–1568.

[17] Akrivi Vlachou, Christos Doulkeridis, Apostolos Glenis, Georgios M. Santipantakis, and George A. Vouros. 2019. Efficient spatio-temporal RDF query processing in large dynamic knowledge bases. In *Proc. of SAC*. 439–447.

[18] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, and Kjetil Nørvåg. 2010. Reverse top-k queries. In *Proc. of ICDE*. 365–376.

[19] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. 2016. Simba: Efficient In-Memory Spatial Analytics. In *Proc. of SIGMOD*. 1071–1085.

[20] Simin You, Jianting Zhang, and Le Gruenwald. 2015. Large-scale spatial join query processing in Cloud. In *Proc. of ICDE Workshops*. 34–41.

[21] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. 2016. A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In *Proc. of ICDE*. 1410–1413.