Interactive and Scalable Exploration of Big Spatial Data - A Data Management Perspective

(Invited Industrial Paper)

Mohamed Sarwat
School of Computing, Informatics, and Decision Systems Engineering,
Arizona State University, Tempe, AZ
699 South Mill Ave, Tempe, AZ 85281, USA
msarwat@asu.edu

Abstract—**Recently, the volume of available spatial data increased tremendously. For instance, in November 2013 NASA announced the release of hundreds of Terabytes of its earth remote sensing dataset. Such data includes but not limited to: weather maps, socioeconomic data, vegetation indices, geological maps, and more. Making sense of such spatial data will be beneficial for several applications that may transform science and society – For example: (1) Space Science: that allows astronomers to study and probably discover new features of both the earth and the outer space, (2) Socio-Economic Analysis: that includes for example climate change analysis, study of deforestation, population migration, and variation in sea levels, (3) Urban Planning: assisting government in city planning, road network design, and transportation engineering, (4) Disaster Planning: that helps in assessing the impact of natural disasters. The main aim of this paper is to investigate novel data management techniques that enable interactive and scalable exploration of big spatial data.**

The paper envisions novel system architectures that provide support for interactive and spatial data exploration, as follows: (1) The paper suggests extending data analytics frameworks, e.g., Apache Spark, to support spatial data types and operations at scale. The resulting framework will serve as a scalable backbone for processing spatial data exploration tasks. (2) It also sketches novel structures and algorithms that leverage modern hardware, e.g., SSDs, and in-memory data processing techniques to efficiently store and access spatial data. Second, the paper proposes extending spatial database systems to support an exploration-aware spatial query evaluation paradigm through three novel components: (1) Spatial Query Steering: that allows the user to slightly modify the query conditions online (zooming in/out) and retrieve the new results in very low latency. (2) Recommendation-Aware Spatial Querying: that injects the recommendation functionality inside classical spatial query executors to support spatial data recommendation. It leverages recommendation algorithms to predict what spatial objects/areas the user would like based on her past interactions with the system. (3) Spatial Query Approximation: That aims at achieving interactive performance by studying the tradeoff between approximate spatial data exploration and query response time.

I. Introduction

Spatial data represent objects that possess spatial attributes which denote the spatial locations and/or geometrical shape of these objects. In the past decade, the volume of available spatial data increased tremendously. For instance, in November 2013 NASA announced the release of a 500 TeraBytes of its earth dataset generated by remote sensing and satellite imagery technologies [17]. Such data includes but not limited to: weather maps, socioeconomic data, vegetation indices, geological maps, and more. Furthermore, NASA mentions that the released data represents just an initial collection and that they plan to publish more datasets through the Open NASA Earth Exchange (OpenNEX) program. On the other hand, novel technology allows hundreds of millions of users to frequently use their mobile devices to access their healthcare information and bank accounts, interact with friends, buy stuff online, search interesting places to visit on-the-go, ask for driving directions, and more. In consequence, everything we do on the mobile internet leaves breadcrumbs of spatial digital traces, e.g., geo-tagged tweets, venue check-ins.

Making sense of such spatial data will be beneficial for several applications that may transform science and society – For example: (1) Space Science: that allows astronomers to study and probably discover new features of both the earth and the outer space, (2) Socio-Economic Analysis: that includes for example climate change analysis [13], study of deforestation [64], population migration [12], and variation in sea levels [59], (3) Urban Planning: assisting government in city/regional planning, road network design, and transportation / traffic engineering [30], (4) Disaster Planning: That helps in assessing the impact of natural (e.g., Hurricanes) and man-made disasters, (5) Commerce and Advertisement [8], [15], [61], [67]: that includes for instance point-of-interest (POI) recommendation services in which we analyze the user spatial behavior and recommends POIs accordingly. Many of the aforementioned applications needs a powerful data management platforms to handle the large volume of spatial data such applications deal with.

A. Challenges for Big Spatial Data Exploration

Data scientists are in deep need of a system that allows them to make sense of big spatial data. Despite the abundance of spatial data, most of it is not well-leveraged yet mostly due to the following system challenges:

- **Challenge I: Exploratory Interface.** In many cases, the user (data scientist/analyst) does not know exactly what kind of information he needs to extract from the spatial data at hand. A user would need a data management system that allows her to explore, and not only to query spatial data using classical query processing techniques.
- **Challenge II: System Scalability.** The massive-scale of available spatial data hinders making sense of it using traditional spatial query processing techniques. Moreover, big spatial data, besides its tremendous storage footprint, may be extremely difficult to manage and maintain. The underlying data management system must be able to digest Petabytes of spatial data, effectively stores it, and allows applications to efficiently retrieve it when necessary.
- **Challenge III: Interactive Performance.** In a spatial data exploration session, the user will not tolerate delays introduced by the underlying spatial data management system to execute queries efficiently. Instead, the user needs to see useful information quickly and interactively change her query if necessary. Hence, the underlying spatial database system must figure out effective ways to process user's request in a sub-second response time.

B. GeoExpo - Vision

This paper investigates novel data management techniques for interactive and scalable exploration of big spatial data. To tackle the aforementioned challenges, this paper proposes GeoExpo a system that consists of the following two main components:

The first component investigates novel system architectures that provide support for interactive and spatial data exploration (Challenges I & II). To build this component, we propose the following two main research tasks: (1) The first task addresses the problem of extending Apache Spark (a scalable data analytics framework) to support spatial data types and operations. The resulting framework will serve as a scalable backbone for processing spatial data exploration tasks. (2) Existing spatial database structures [28], [41], [36], [33] and algorithms (e.g., spatial range, spatial join [10], [48], [51]) deal with the implicit assumption that the underlying secondary storage is the mechanical hard disk (HDD). HDD notoriously exhibits poor data access performance, especially when interactive performance is deemed a priority. On the other hand, modern non-volatile memory devices (i.e., Solid-State-Drives) exhibits orders of magnitude better I/O speed and throughput than HDD. This paper proposes tailored spatial data structures and algorithms that leverage modern non-volatile memory devices to efficiently store and access spatial data residing on each spatial data node. Furthermore, this paper aims at leveraging state-of-the-art in-Memory data processing techniques to achieve sub-second response time necessary for spatial data exploration workload. The ultimate goal is to harness the maximum capacity of the memory hierarchy for storing and accessing spatial data to achieve interactive performance (Challenge III).

To address the problem of injecting exploration-awareness in the classical spatial querying paradigm (Challenge I), the paper envisions an exploration-aware spatial query execution component would provide the following main functionalities: *(1) Spatial Query Steering and Speculation*: Speculate what kind of spatial data and/or spatial operations the user might be interested in based on her query context and history of interactions with the system. This task investigates how the spatial query executor could help the user to slightly modify the query qualifications and retrieve the new results in sub-second response. A slight modification might be a small change to the designated spatial range or the proximity condition, and zooming in/out the initial spatial area of interest. *(2) Recommendation-Aware Spatial queries:* This research task studies the integration of recommender systems functionalities with state-of-the-art spatial query execution engines. In such case, the user might provide feedback by specifying whether she liked (or not) the spatial exploration results retrieved so far. The system then quickly learns the user's preferences and retrieve more relevant results during the same or future spatial data exploration session. *(3) Spatial Query Approximation:* Existing spatial query executors utilize a lot of the system resources (and still take too much time) to generate an exact spatial query answer whereas an approximate answer might sometimes be good enough. This paper studies the tradeoff between result accuracy and performance in spatial join and spatial aggregate analytics and proposes a framework for real-time approximate query execution over large-scale geospatial data. The main goal is to generate approximate spatial query answers in sub-seconds response time (Challenge III) while minimizing the accuracy loss.

II. GeoExpo Overview

GeoExpo has two main components (see Figure 1): (1) novel system architectures to support scalable spatial computing operations (Section III). (2) investigates extending state-of-the-art spatial database systems to support exploration-aware spatial query processing techniques (Section IV).

III. Scalable Spatial Data Management Platform

The goal of this component is two-fold: (1) Extending state-the-art scalable cluster computing paradigms to support spatial computations (Section III-A) and (2) Crafting novel spatial data management techniques on modern hardware devices (Section III-B).

A. Extending Apache Spark to support Spatial Data

Existing spatial database management systems, e.g, PostGIS [49], extends classical relational database systems with new data types, functions, operators, and index structures to handle spatial operations based on the Open Geospatial Consortium. Even though such systems sort of provide full support for spatial data storage and access, they suffer from a scalability issue. Based upon a relational database system, such systems are not scalable enough to handle more than Gigabytes or at most few terabytes of spatial data, and hence

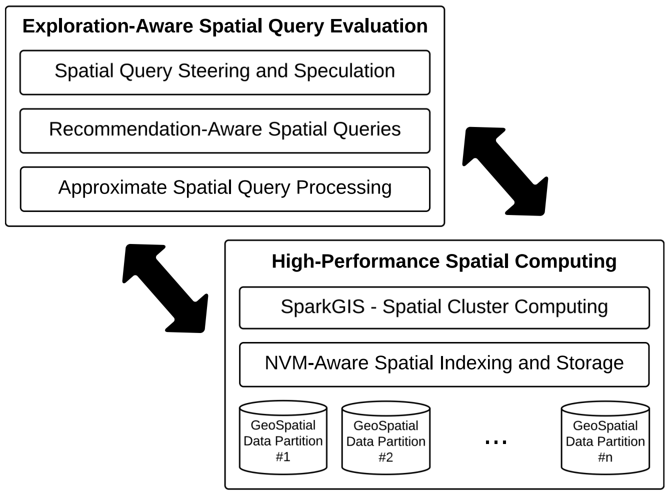
Exploration-Aware Spatial Query Evaluation
Spatial Query Steering and Speculation
Recommendation-Aware Spatial Queries
Approximate Spatial Query Processing
High-Performance Spatial Computing
SparkGIS - Spatial Cluster Computing
NVM-Aware Spatial Indexing and Storage
GeoSpatial Data Partition #1
GeoSpatial Data Partition #2
...
GeoSpatial Data Partition #n

Fig. 1. GeoExpo Overview.

are not able to perform large-scale analytics over big spatial data. Recent works (e.g., [5], [6], [18], [19], [45], [65]) extend the Hadoop [29] ecosystem, e.g., MapReduce [14] and Hive [58], to perform spatial analytics at scale. The Hadoop-based approach indeed achieves high scalability. However, these systems though exhibits excellent performance in batch-processing jobs, they show poor performance handling applications that require interactive performance. For instance, consider the case when a data scientist wants to analyze spatial data with exploration intent and hence cannot tolerate hours to get the analysis results.

Apache Spark [57], [63], on the other hand, is a in-memory cluster computing system. Spark provides a novel data abstraction called resilient distributed datasets (RDDs) [62] that are collections of objects partitioned across a cluster of machines. Each RDD is built using parallelized transformations (filter, join or groupBy) that could be traced back to recover the RDD data. For fault tolerance, Spark rebuild lost data on failure using lineage: each RDD remembers how it was built from other datasets (through transformations) to recover itself. RDDs allow Spark to outperform existing models (Hadoop MapReduce) by up to two orders of magnitude in multi-pass analytics. Spark is easy to use and program as it provides an intuitive API for users to write their data analytics tasks on RDDs. Unfortunately, Spark does not provide native support for spatial data and spatial operations. Hence, users need to perform the tedious task of programming their own spatial data exploration jobs on top of Spark.

This module addresses the problem of extending Apache Spark to support spatial data types and operations. The main aim is to extend the resilient distributed datasets (RDDs) concept to support spatial data. This problem is quite challenging due to the fact that (1) spatial data may be quite complex, e.g., rivers' and cities' geometrical boundaries, (2) spatial (and geometric) operations (e.g., Overlap, Intersect, Convex Hull, Cartographic Distances) [50] cannot be easily and efficiently expressed using regular RDD transformations and actions. GeoExpo investigates the problem of how spatial data could represented using RDDs to form spatial RDDs (SRDDs). More specifically, this task addresses the problem of efficiently partitioning SRDD data elements across machines and introduces novel parallelized spatial (geometric operations that follows the Open Geosptial Consortium (OGC) [46] standard) transformations and actions (for SRDD) that provide a more intuitive interface for users to write spatial data exploration and analytics programs on-top of Spark. This task also aims at extending Spark SQL engine (Shark / Spark SQL) [20], [60] to support spatial query evaluation. SharkGIS would consists of a set of SQL User-Defined-Functions (UDFs) that maps to spatial data types and proximity constraints. The input/output of these UDFs would be quite similar to UFDs defined in PostGIS, an extension to PostgreSQL [1] that provides a SQL interface for users to express spatial operations on geographical data. SharkGIS (the proposed Shark Spatial Extension) will build on the SRDDs concept (mentioned earlier) to decide how spatial object-relational tuples could be stored, indexed, and accessed using SRDDs. Furthermore, One of the goals of SharkGIS is to craft efficient methods to process, spatial range, KNN (K-Nearest Neighbors), and spatial join queries [35], [34], [48] inside Apache Spark.

B. Spatial Data Management on Modern Hardware

Existing spatial data management algorithms deal with the implicit assumption that the underlying secondary storage is the mechanical hard disk (HDD). HDD notoriously exhibits poor data access performance, especially when interactive performance is deemed a priority [27]. GeoExpo investigates novel spatial data structures and algorithms that leverage modern non-volatile memory devices (e.g., Flash Memory Solid-State-Drives (SSD), Phase Change Memory (PCM)) to efficiently store and access spatial data. For instance, an SSD device is block-oriented, i.e., pages are clustered into a set of blocks. Moreover, SSDs are also able to parallelize I/O operations and hence increase the overall I/O throughput. Thus, it has fundamentally different characteristics, compared to the conventional magnetic disks, especially for the write operations.Such a popularity of SSDs is mainly due to its superior characteristics that include smaller size, lower power consumption, and faster read performance [9], [26], [40], [66].

This module presents novel spatial data structures and algorithms that leverage modern Non-Volatile Memory devices to efficiently store and access spatial data. Spatial tree indexes, e.g., R-Tree and its variants [36], [37], [56], are crucial for efficient spatial data retrieval. Existing solutions are build for one-dimensional B^+-tree like indexes and hence are not well adapted to handle spatial data workloads [4], [44], [43], [39]. GeoExpo distinguishes itself from previous approaches in two main aspects: (1) Rather than focusing on a specific index structure or building a new index, We aim to build a generic framework, similar to the GIST framework [31], that can be applied to a wide variety of tree index structures, including R-tree along with its variants. (2) GeoExpo aims at achieving both efficiency *and* durability in the same design. For efficiency, the proposed approach buffers all the incoming updates in memory while employing a *flushing policy* that evicts selected updates from memory to minimize the cost of writing to the flash storage. In the mean time, we need guarantee durability by sequentially logging each in-memory update and by employing an efficient *crash recovery* technique. In our preliminary study, we designed FAST [54], [55]; a framework for Flash-Aware Spatial Tree index structures. Algebraic cost models and experiments show that the proposed approach provided 80 % boost in performance for range queries over road-network spatial datasets. We plan to extend this idea by investigating the use of a richer memory hierarchy that includes: RAM, SSD, and HDD [16]. Our initial study shows that we could classify spatial regions as hot or cold based on the spatial operations performed on these regions. Therefore, the system caches hot spatial regions on SSDs and cold spatial regions on HDD to decrease the overall I/O latency. Another goal of this task is extending spatial database systems to harness the I/O parallelism in SSDs. To our knowledge, no previous works investigated leveraging internal SSD parallelism to increase spatial database I/O throughput. There exist multiple levels of parallelism in flash memory SSDs. For instance, flash memory packages are connected to an SSD controller through multiple channels. We can control the spatial operations' I/O workload such that each channel can be operated independently and simultaneously. Moreover, since each flash memory package can be operated independently, spatial I/O operations on a package attached to the same channel can be interleaved so the bus utilization can be optimized. Moreover, we propose storing col-located spatial data on the same die of a flash memory package such that each die can execute independent spatial commands which increases the overall spatial I/O throughput. This task also investigates spatial data requests re-scheduling to exploit internal parallelism of SSDs to further boost the overall performance of spatial data exploration tasks.

IV. Exploration-Aware Spatial Query Evaluation

Existing spatial databases exhibit ill-Support for an exploration workloads – They do not provide a querying interface to its users that facilitates spatial data exploration. They instead necessitates that users must know (in advance) what spatial data they need and allows them to issue queries accordingly. GeoExpo aims at crafting an exploration-aware spatial querying paradigm through the following research tasks.

A. Spatial Query Steering and Speculation

This task investigates query speculation for spatial data exploration. For example, assume the user issues a spatial query [47] that retrieves spatial objects that are within a specific rectangular range R. Assume the user, in exploration mode, then decided to slightly expand, shrink, or move the original rectangular range query R to $R^{'}$. If the system could predict $R^{'}$, it might speculatively pre-fetch the answer to $R^{'}$ so that the user gets the answer to $R^{'}$ very fast when needed. To achieve that, the system needs to employ a smart query speculation algorithm that is able to speculate what kind of queries the user might issue in her spatial data exploration session. This task is challenging for the following reasons: (1) The number of spatial query variations might be endless and hence speculatively computing the answer to all possible variations leads to huge system overhead. (2) Even if the number of speculative queries is reasonable, the user might wind up not using any of the speculatively calculated answers and hence the amount of work spent on speculation and data pre-fetching would be a waste.

This module investigates the problem of predicting the user's future interactions (modification to original query) with the system during her current exploration session. When the user issues an ad-hoc spatial query (e.g., Q_1 : Return Air Quality data within Tempe, Arizona), the system might for instance determine a set of speculative queries that returns air quality in geospatial regions that are of close proximity to Tempe (e.g., Q_2 : Return Air Quality Data within Scottsdale, AZ), bigger regions that contain Tempe (e.g., Q_3 : Return Air Quality Data within the Phoenix Metropolitan area), or smaller neighborhoods within Tempe (e.g., Q_4 : Return Air Quality Data around the Arizona State University Campus). The system processes these speculative queries in the background, and materializes the returned spatial objects in a data structure called the SQ_Cache. The query speculation algorithm relies on two main components to take its decision, briefly described as follows: (1) Offline Modeling Component: that takes as input: (a) $\langle$*User,Object*$\rangle$ Set: this represents the set of spatial objects (polygons or points) that appeared in the answer of past spatial queries issued by each system user. (b) Spatial Exploration Constraints: the possible set of exploration actions available for the system users to change the qualifications of the spatial queries. Therefore, the system builds a query speculation model using a Hidden Markov Model in which the hidden state is the successive spatial queries in the exploration session. (2) Spatial Query Steering: this component implements a deterministic finite state automaton (DFA) that represents the spatial exploration session: each DFA state denotes a spatial exploration action (Panning, Zooming In/Out) as well as user feedback states. This component also performs progressive processing of spatial queries where the system produces the result for speculative queries based on partially processed spatial regions. When the user submits a new ad-hoc spatial query, the system calculates the likelihood that the user would perform a specific action, generates speculative spatial queries accordingly, and returns the top-k queries that are likely to be posed by the user during the exploration session. The system pre-computes and materializes the relevant spatial data in the SQ_Cache which will be fetched later when necessary.

B. Spatial Query Approximation

Achieving real-time performance for queries issued over big spatial data may sometimes be quite challenging even when employing high performance computing and modern hardware infrastructure. Assume an environmental scientist

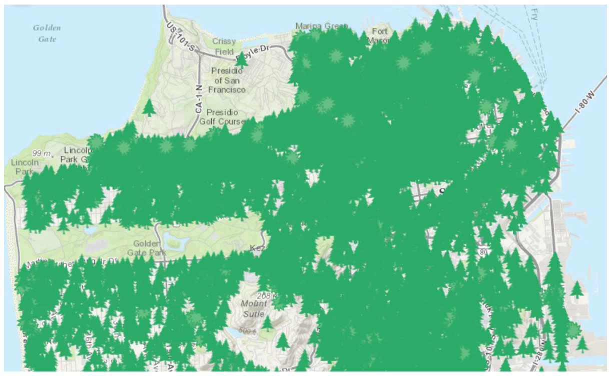

Fig. 2. San Francisco Trees

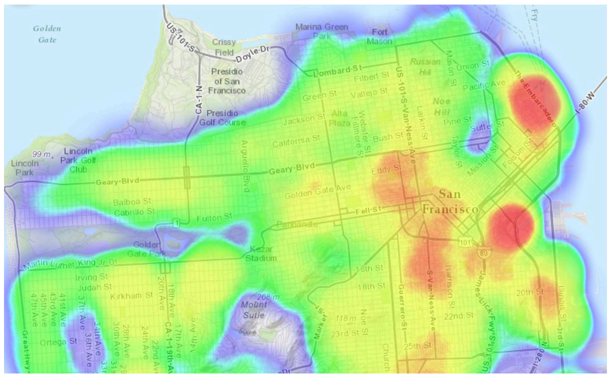

Fig. 3. San Francisco Heat Map

– studying the relationship between air quality and trees – would like to explore the trees population in San Francisco (See Figure 2[1]. As given in Fig 3, a query that just returns all trees in San Francisco would not help the user much. Alternatively, a heat map (spatial aggregate) that shows the distribution of trees in San Francisco (See Fig 3[2]) would be more helpful to the data scientist to explore the data. This spatial aggregate query (i.e., heat map) needs to count all trees at every single ⟨latitude,longitude⟩ position over the map. Now assume the user zooms-in the San Francisco trees heat map. Given that spatial data size (e.g., weather) and the U.S. geographical locations might be huge, traditional SQL aggregation techniques [24], [23], [25] may take so long to execute such interactive exploration queries. This task addresses the problem of trading interactive performance for accuracy in evaluating spatial queries. The main aim of this task is to come up with a good enough approximation to answer spatial queries in real time.

The idea of our first cut solution is to craft a novel data structure, called the spatial exploration pyramid structure instead of a simple grid structure [7], [21], as depicted in Figure 4. The pyramid decomposes the space into H levels (i.e., pyramid height). For a given level h, the space is partitioned into 4^h equal area grid cells. In each cell, we store

[1]Figure created using ArcGIS JavaScript API

[2]Figure created using ArcGIS JavaScript Heatmap Overlay API

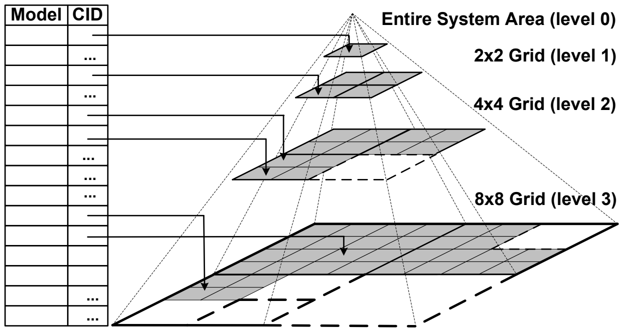

Fig. 4. Exploration Pyramid

spatial aggregation models (i.e., Count, Average, Sum) using *only* the spatial objects with user locations contained in the cell's spatial region. A spatial object may contribute to up to H aggregation models; one per each pyramid level. Note that the root cell (level 0) of the pyramid represents the aggregate values built using all spatial data in the system. Levels in the pyramid can be incomplete, as we will periodically merge or split cells. For example, in Fig 4, the four cells in the upper right corner of level 3 are not maintained (depicted as blank white squares).

This pyramid structure is advantageous over a simple grid structure for two main reasons: (1) We will be maintaining much less number of cells than the gird structure as not every cell in every level is maintained, and (2) Users can freely explore different levels (Zoom in/out) of the pyramid by setting a preferred *zoom level* indicating how localized they would like to receive their spatial aggregate measure. A zoom level of zero means that the user would like to get her aggregate measure from the root level. The higher zoom level, the more localized is the answer. We aim to build on our initial ideas to: (1) build accurate cost models and quality of service measures for the shape of the spatial exploration pyramid to study the trade-off of merging four neighbor grid cells to only one maintained at a higher level. Merging children cells will reduce the system overhead as less cells will need to be maintained, however, this may affect how the accuracy of more localized spatial aggregation, and (2) design an online decision model that decides on merging and splitting pyramid cells online. A merging decision can be taken if an good approximate (with reasonable accuracy) answer at an individual (child) cell could be easily inferred using the spatial aggregate measure stored at a merged (parent) cell. Also, a splitting decision can be taken if the system finds that more localized answer can be obtained, and that is worth the extra overhead. Since existing approximate query processing techniques do not take into account the characteristics of complex spatial [3], [2], this task also studies spatial data sampling techniques to process spatial aggregation with guaranteed error bounds.

C. *Recommendation-Aware Spatial Query Processing*

Many spatial operations, e.g., Spatial Overlap, Spatial Join, relies on spatial data indexes (e.g., R-tree, Grid) as a backbone to efficiently store and access spatial data. However, existing spatial data index structures allows users to lookup spatial data that matches exactly the designated query. For example, Query 2 (given below) returns all hotels (store in the Hotels Table in PostGIS) that lie within the spatial extents of the Phoenix metropolitan area. Such classical spatial queries clearly do not provide support for spatial data recommendation.

Query 1: Retrieve all Hotels that exist in the 'Phoenix' urban area.

```
Select H.name From Hotels as H, City as C
Where C.name = 'Phoenix'
AND ST_Contains(C.geom, H.geom)
```

Recommendation [11], [32], [38], [52] is the process of suggesting useful data to the user with based on a large pool of historical data. GeoExpo aims to extend existing spatial data index structures to support spatial data exploration through recommendation. In order to support spatial data exploration, we propose extending existing spatial data indexes and query execution engines to support spatial data recommendation. The problem is to come up with an efficient way to store spatial data, filter it, predict its relevancy and recommend it.

The straightforward approach considers a set of data points of interest in space and then constructing a spatial index, e.g., R-tree, on all the spatial data objects. Similarity between different items/data points in the database can be calculated by using the cosine distance. When the user asks for a set of spatial objects in a certain boundary region, the system first searches for all the points lying within the region. It is not feasible to go through all the data point outcomes from the search. For this purpose, the system filters huge data ignoring all those points that may not interest him based on his previous interactions with the system. Then, only the items with top-k recommendation scores are displayed to the user. The disadvantages of this method is as follows: (1) If the spatial search gives returns many results, recommendation score calculation has to be done for all those data points. Hence, this approach is advisable only for highly selective queries. (2) In case the total number of spatial objects exceeds the number of objects explored by the user, this method may not be adequate. (3) If a user did not rate many spatial data points in the given region, then recommendations may not be appropriate.

Some of these disadvantages can be overcome by calculating recommendation score of items rated prior to searching them on the R-tree. The data points with recommendation scores are looked up in the spatial index to check whether they belong to the designated location. All those objects belonging to the desired region will be sorted and the top-k will be eventually displayed to the user. This problem can be further optimized to give faster recommendations by improving the storage of spatial data. GeoExpo investigates the problem of augmenting existing spatial indexes, e.g., R-Tree, to filter spatial data using both their spatial attributes and their recommendation scores. This research also extends existing spatial query execution engines with an additional Spatial-Recommend SQL operator. Consider the following scenario: Alice plans to visit 'Phoenix' for business and she looks for Hotels in the area. Query 2 predicts how much user 1 (i.e., Hotel) would like unexplored hotels using the `RECOMMEND` operator (based on the singular value decomposition algorithm [22]). The query also leverages the `ST_Contains()` function (provided by PostGIS) to recommend only hotels that lie within the extent of the 'Phoenix' area. In this scenario, the system first needs to retrieve hotels that lie within the 'Phoenix' area. Therefore, the system could predict what hotels Alice would be interested in based on her past interactions as well as other users' interactions with the system [53], [42].

Query 2: Predict How much user 1 like Hotels that exist in the 'Phoenix' urban area.

```
Select H.name, R.ratingva
From HotelRatings as R, Hotels as H, City as C
Recommend R.iid To R.uid On R.ratingVal Using SVD
Where R.uid=1 AND R.iid=H.vid
AND C.name = 'Phoenix'
AND ST_Contains(C.geom, H.geom)
```

As presented above, GeoExpo aims at seamlessly integrating the recommendation functionality within the core of a spatial database engine.

V. Conclusion

The paper proposes GeoExpo; a system that tackles data management challenges that lie ahead of enabling scalable and interactive spatial data exploration. More specifically, the paper lists three main system challenges that faces spatial data exploration applications: (1) An Exploratory Interface, (2) System Scalability, and (3) Interactive Performance. To address these challenges, GeoExpo envisions two main components: (A) High-Performance Spatial Computing Platform: this component aims at leveraging state-of-the-art cluster/parallel computing paradigms (i.e., Apache Spark) to support complex geospatial operations at scale. It also extends existing spatial data stores to harness the uniques characteristics of Non-Volatile Memory devices to boost spatial data access performance. (B) Exploration-Aware Spatial Querying Paradigm: This component re-thinks existing spatial querying paradigms to incorporate user exploration intent. More specifically, this component investigates speculative query execution and query steering strategies to provide interactive performance during the user's spatial data exploration section. This component also studies the tradeoff between the accuracy of the spatial query result and the system responsiveness and proposed an efficient approximate spatial query execution approach. Moreover, this component integrates state-of-the-art recommendation systems to allow users to explore spatial data based on their personal preferences and their past interactions with the system.

References

[1] PostgreSQL. http://PostgreSQL:http://www.postgresql.org.

[2] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. I. Jordan, S. Madden, B. Mozafari, and I. Stoica. Knowing when you're wrong: building fast and reliable approximate query processing systems. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 481–492, 2014.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42, 2013.

[4] D. Agrawal, D. Ganesan, R. K. Sitaraman, Y. Diao, and S. Singh. Lazy-Adaptive Tree: An Optimized Index Structure for Flash Devices. *PVLDB*, 2009.

[5] A. Aji, X. Sun, H. Vo, Q. Liu, R. Lee, X. Zhang, J. H. Saltz, and F. Wang. Demonstration of hadoop-gis: a spatial data warehousing system over mapreduce. In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems, ACM GIS*, 2013.

[6] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. H. Saltz. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *Proceedings of the VLDB Endowment, PVLDB*, 6(11):1009–1020, 2013.

[7] W. G. Aref and H. Samet. Efficient Processing of Window Queries in the Pyramid Data Structure. In *Proceedings of the ACM Symposium on Principles of Database Systems, PODS*, pages 265–272, Nashville, TN, Apr. 1990.

[8] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and Preference-aware Recommendation Using Sparse Geo-social Networking Data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, Proceedings of the ACM Symposium on Advances in Geographic Information Systems, ACM GIS, 2012.

[9] L. Bouganim, B. Jónsson, and P. Bonnet. uFLIP: Understanding Flash IO Patterns. In *Proceedings of the International Conference on Innovative Data Systems Research, CIDR*, 2009.

[10] T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient Processing of Spatial Joins Using R-Trees. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 237–246, 1993.

[11] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 1997.

[12] C. Chen, M. Burton, E. Greenberger, and J. Dmitrieva. Population migration and the variation of dopamine D4 receptor (DRD4) allele frequencies around the globe. *Evolution and Human Behavior*, 20(5):309–324, 1999.

[13] N. R. C. Committee on the Science of Climate Change. *Climate Change Science: An Analysis of Some Key Questions*. The National Academies Press, 2001.

[14] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of ACM*, 51:107–113, 2008.

[15] S. Dhar and U. Varshney. Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5):121–128, 2011.

[16] J. Do, D. Zhang, J. M. Patel, and D. J. DeWitt. Fast peak-to-peak behavior with SSD buffer pool. In *Proceedings of the IEEE International Conference on Data Engineering, ICDE*, 2013.

[17] Earth Science Data on AWS With NASA / NEX Public Data Sets. http://aws.typepad.com/aws/2013/11/process-earth-science-data-on-aws-with-nasa-nex.html.

[18] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan. CG_Hadoop: computational geometry in MapReduce. In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems, ACM GIS*, 2013.

[19] A. Eldawy and M. F. Mokbel. A demonstration of spatialhadoop: An efficient mapreduce framework for spatial data. *Proceedings of the VLDB Endowment, PVLDB*, 6(12):1230–1233, 2013.

[20] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: fast data analysis using coarse-grained distributed memory. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 689–692, 2012.

[21] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[22] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[23] L. I. Gómez, S. A. Gómez, and A. A. Vaisman. A generic data model and query language for spatiotemporal OLAP cube analysis. In *Proceedings of the International Conference on Extending Database Technology, EDBT*, pages 300–311, 2012.

[24] L. I. Gómez, S. Haesevoets, B. Kuijpers, and A. A. Vaisman. Spatial aggregation: Data model and implementation. *Information Systems*, 34(6):551–576, 2009.

[25] L. I. Gómez, B. Kuijpers, and A. A. Vaisman. A data model and query language for spatio-temporal decision support. *GeoInformatica Journal*, 15(3):455–496, 2011.

[26] J. Gray and B. Fitzgerald. Flash Disk Opportunity for Server Applications. *ACM Queue*, 2008.

[27] J. Gray and G. Graefe. The five-minute rule ten years later, and other computer storage rules of thumb. *SIGMOD Record*, 26(4):63–68, 1997.

[28] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 47–57, 1984.

[29] Apache. Hadoop. http://hadoop.apache.org/.

[30] P. Hall. *Cities of Tomorrow: An Intellectual History of Urban Planning and Design Since 1880*. John Wiley & Sons, 2014.

[31] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized search trees for database systems. In *Proceedings of the International Conference on Very Large Data Bases, VLDB*, 1995.

[32] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems, TOIS*, 22(1):5–53, 2004.

[33] E. G. Hoel and H. Samet. Data-Parallel R-Tree Algorithms. In *International Conference on Parallel Processing, ICPP*, pages 47–50, 1993.

[34] E. G. Hoel and H. Samet. Benchmarking spatial join operations with spatial output. In *VLDB*, pages 606–618, 1995.

[35] E. H. Jacox and H. Samet. Spatial join techniques. *ACM Transactions on Database Systems, TODS*, 32(1):7, 2007.

[36] I. Kamel and C. Faloutsos. Hilbert R-tree: An Improved R-tree using Fractals. In *Proceedings of the International Conference on Very Large Data Bases, VLDB*, pages 500–509, 1994.

[37] N. Katayama and S. Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 1997.

[38] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. FlexRecs: Expressing and Combining Flexible Recommendations. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 745–758, Providence, RI, July 2009.

[39] S. Lee, B. Moon, and C. Park. Advances in Flash Memory SSD Technology for Enterprise Database Applications. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2009.

[40] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim. A case for Flash memory SSD in Enterprise Database Applications. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2008.

[41] S. Leutenegger, M. Lopez, and J. Edgington. STR: A Simple and Efficient Algorithm for R-Tree Packing. In *Proceedings of the International Conference on Data Engineering, ICDE*, pages 497–506, 1997.

[42] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. LARS: A Location-Aware Recommender System. In *Proceedings of the International Conference on Data Engineering, ICDE*, 2012.

[43] Y. Li, B. He, Q. Luo, and K. Yi. Tree indexing on Flash Disks. In *Proceedings of the IEEE International Conference on Data Engineering, ICDE*, 2009.

[44] Y. Li, B. He, R. J. Yang, Q. Luo, and K. Yi. Tree Indexing on Solid State Drives. *PVLDB*, 3(1), 2010.

[45] J. Lu and R. H. Guting. Parallel Secondo: Boosting Database Engines with Hadoop. In *International Conference on Parallel and Distributed Systems*, pages 738 –743, 2012.

[46] Open Geospatial Consortium. http://www.opengeospatial.org/.

[47] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the International Conference on Very Large Data Bases, VLDB*, pages 802–813, 2003.

[48] J. M. Patel and D. J. DeWitt. Partition based spatial-merge join. In *ACM SIGMOD Record*, volume 25, pages 259–270. ACM, 1996.

[49] PostGIS. http://postgis.net.

[50] F. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[51] D. Rotem. Spatial Join Indices. In *Proceedings of the International Conference on Data Engineering, ICDE*, pages 500 –509, 1991.

[52] M. Sarwat, J. Avery, and M. F. Mokbel. RecDB in Action: Recommendation Made Easy in Relational Databases. *PVLDB*, 6(12):1242–1245, 2013.

[53] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. LARS*: An Efficient and Scalable Location-Aware Recommender System. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 26(6):1384–1399, 2014.

[54] M. Sarwat, M. F. Mokbel, X. Zhou, and S. Nath. Fast: A generic framework for flash-aware spatial trees. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases, SSTD*, 2011.

[55] M. Sarwat, M. F. Mokbel, X. Zhou, and S. Nath. Generic and efficient framework for search trees on flash memory storage systems. *GeoInformatica*, 17(3):417–448, 2013.

[56] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *Proceedings of the International Conference on Very Large Data Bases, VLDB*, 1987.

[57] Spark. https://spark.apache.org.

[58] A. Thusoo, J. S. Sen, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: A Warehousing Solution over a Map-Reduce Framework. *Proceedings of the VLDB Endowment, PVLDB*, pages 1626–1629, 2009.

[59] P. L. Woodworth, M. Menéndez, and W. R. Gehrels. Evidence for century-timescale acceleration in mean sea levels and for recent changes in extreme sea levels. *Surveys in geophysics*, 32(4-5):603–618, 2011.

[60] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: SQL and rich analytics at scale. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, pages 13–24, 2013.

[61] M. Ye, P. Yin, and W.-C. Lee. Location Recommendation for Location-based Social Networks. In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems, ACM GIS*, pages 458–461, San Jose, CA, Nov. 2010.

[62] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation, NSDI*, pages 15–28, 2012.

[63] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: fault-tolerant streaming computation at scale. In *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles, SOSP*, pages 423–438, 2013.

[64] N. Zeng, R. E. Dickinson, and X. Zeng. Climatic Impact of Amazon Deforestation?A Mechanistic Model Study. *Journal of Climate*, 9, 1996.

[65] C. Zhang, F. Li, and J. Jestes. Efficient Parallel kNN Joins for Large Data in MapReduce. In *Proceedings of the International Conference on Extending Database Technology, EDBT*, pages 38–49, 2012.

[66] N. Zhang, J. Tatemura, J. M. Patel, and H. Hacigümüs. Re-evaluating designs for multi-tenant OLTP workloads on SSD-based I/O subsystems. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD*, 2014.

[67] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative Location and Activity Recommendations with GPS History Data. In *Proceedings of the International Conference on World Wide Web, WWW*, pages 1029–1038, Raleigh, NC, Apr. 2010.