High-Performance Spatial Data Analytics: Systematic R&D for Scale-Out and Scale-Up Solutions from the Past to Now

Fusheng Wang
Stony Brook University
fusheng.wang@stonybrook.edu

Rubao Lee
Freelance
lee.rubao@ieee.org

Dejun Teng
Shandong University
teng@sdu.edu.cn

Xiaodong Zhang
The Ohio State University
zhang.574@osu.edu

Joel Saltz
Stony Brook University
joel.saltz@stonybrookmedicine.edu

ABSTRACT

We released open-source software Hadoop-GIS in 2011, and presented and published the work in VLDB 2013. This work initiated the development of a new spatial data analytical ecosystem characterized by its large-scale capacity in both computing and data storage, high scalability, compatibility with low-cost commodity processors in clusters and open-source software. After more than a decade of research and development, this ecosystem has matured and is now serving many applications across various fields. In this paper, we provide the background on why we started this project and give an overview of the original Hadoop-GIS software architecture, along with its unique technical contributions and legacy. We present the evolution of the ecosystem and its current state-of-the-art, which has been influenced by the Hadoop-GIS project. We also describe the ongoing efforts to further enhance this ecosystem with hardware accelerations to meet the increasing demands for low latency and high throughput in various spatial data analysis tasks. Finally, we will summarize the insights gained and lessons learned over more than a decade in pursuing high-performance spatial data analytics.

PVLDB Reference Format:
Fusheng Wang, Rubao Lee, Dejun Teng, Xiaodong Zhang, and Joel Saltz. High-Performance Spatial Data Analytics: Systematic R&D for Scale-Out and Scale-Up Solutions from the Past to Now. PVLDB, 17(12): 4507-4520, 2024.
doi:10.14778/3685800.3685912

1 INTRODUCTION

At the beginning of the twenty-first century, "the big data" became a serious reality following years of rapid advancements of the Internet and extensive development in computer architecture and storage systems. The volume of data generated by various applications began to grow at an unprecedented rate, challenging traditional data analytics methodologies and limited scalability of commercial databases that could not be maintained at affordable costs for most users. For many years, mainstream R&D of computer systems development focused on scale-up, or vertical scaling, which involves increasing the capacity of a powerful and centralized computer system by adding more hardware and software resources. While the scale-up development results in high-end computers, it comes at a high cost. This approach has the merit of simple system management. However, there are physical and technological limits to how much a high-end computer system can be scaled up.

The big data applications, facing these limitations, desperately sought for much more effective solutions. In 2004, Google published its MapReduce algorithm [31], which enabled the parallel and distributed processing of large data sets across clusters of many commodity processors. This was a turning point in managing big data. Following this, in 2006, the release of Hadoop [15], an open-source implementation of MapReduce, provided a foundational technology for big data processing. In contrast to the scale-up approach, MapReduce and Hadoop utilize the scale-out, or horizontal scaling method. This approach increases system capacity by adding more commodity computers or nodes for parallel and distributed computing across many nodes. While the scale-out approach introduces challenges due to increased system complexity of managing and coordinating a large number of computer nodes, it is much more cost-effective compared to scale-up systems. Moreover, its scalability effectively addresses the needs of big data applications, as there is virtually no limit to the number of nodes that can be added. We have analyzed the scale-out approach using a matrix model called DOT to demonstrate its scalability [40].

On top of Hadoop, various big data management systems for major applications could be developed, and Apache Hive [2] is one of the early and successful systems. Hive is a large-scale SQL relational database and its execution engine runs on Hadoop across clusters of computers. Our contributions to Hive include two critical components. (1) The creation of RCFile [38] and its optimized version ORCFile [41] has enhanced the efficiency of data storage and retrieval in large cluster systems. (2) YSmart [45], is an optimization tool designed to improve the efficiency of SQL queries in Hive. YSmart optimizes query plans, identifies common sub-expressions to minimize the number of MapReduce jobs, and automatically translates SQL queries into optimized MapReduce jobs. For detailed development of Hive, readers may refer to [39].

Meanwhile, the proliferation of mobile phones, Internet of Things (IoT), collaborative data collection projects, ubiquitous sensory measurement technologies and scientific instruments have contributed to generating multidimensional spatial data at unprecedented rate and scale. The need for low-latency data intensive spatial frameworks has become increasingly important to businesses, daily users

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685912

as well as scientific applications ranging from geo-marketing and social engineering to biomedical research and clinical diagnosis. Spatial big data presents unique challenges due to its multidimensional complexity, geometric computations, and sheer volume.

Our group had been active in development of spatial data database methods since the late 1990s [28]; our initial work was motivated by earth science and digital pathology applications. Over the ensuing years, we went on to develop application-agnostic spatial database systems to optimize spatial data subsetting, aggregation, and query execution in parallel and distributed environments [25], [44], [43], [26]. These frameworks, called the Active Data Repository (ADR) and Data Cutter, used Hilbert curves to organize spatial data to optimize the efficiency of subsetting and aggregation operations and to efficiently index multidimensional datasets. This first generation of parallel spatial database systems were developed using the High Performance Computing community MPI message passing interface [37]. The ADR framework provided a SQL-like interface for executing range queries and aggregation operations on large datasets. Several years later, a number of the methods developed in this earlier work were extended and adapted to Hadoop in the early days of the big data era, and this work motivated us to develop the spatial data analytical system called Hadoop-GIS [19]. The contributions and impact of the Hadoop-GIS project have been recognized by the 2024 VLDB Test of Time Award [18]. In this paper, we provide background information, overview its software architecture, and discuss its unique technical contributions, evolution, and lasting legacy. Today, big data challenges are evolving due to the advancements of various hardware devices and the increasing demand for low-latency data processing. We will present our solutions to address these new challenges for spatial data analytics in the paper.

1.1 The Hadoop-GIS Project

The Hadoop-GIS Project has been pivotal in transforming spatial data analytics from traditional commercial parallel database systems to large-scale clusters of commodity processors using open-source software. As one of the earliest Hadoop-based spatial data analytical systems, it has inspired and influenced numerous subsequent projects and systems in both academia and industries.

The work was motivated by extreme-scale spatial data derived from whole slide images in digital pathology [30]. High resolution digital pathology images (up to 100,000x100,000 pixels per slide) provide rich information at micro-anatomic levels, such as cells, fats, ducts and blood vessels. These spatially centric objects can be extracted with image segmentation algorithms [32, 54], to enable novel and more effective ways of screening for disease, classifying disease states, understanding disease progression, and evaluating the efficacy of therapeutic strategies. Much of digital pathology image analysis is GIS-like, such as point-in-polygon, containment, nearest neighbor, and spatial join.

In the beginning of the project, we used a commercial parallel database system, called DBMS-X, to manage and query spatial objects from pathology images with 30 partitions [63, 64]. We encountered four significant challenges during the process. First, setting up and programming on the DBMS-X system was extremely complex, requiring specialized support from the vendor's development team. Second, spatial data loading was a major bottleneck, and it took days to load spatial data from about 500 images. Third, making it scalable to a large number of partitions is very difficult, and there is lack of effective spatial partitioning methods for load balancing. Finally, licensing costs further constrained the system's scalability and affordability for broader use.

Faced with these challenges, we decided to develop a new spatial data warehousing system based on open-source software. This system aimed to be low-cost while offering high performance, high throughput, and high scalability through massive parallel processing across a cluster of commodity processors. By employing this disruptive approach, we were concerned about whether we could achieve acceptable performance, given the potential latency issues with parallel task scheduling, network communication, and synchronization on clusters of computers. With our consistent effort, we were pleased with the performance achieved with the initial release of Hadoop-GIS in 2011 [1].

Following the release of the Hadoop-GIS open-source software and its publication in VLDB in 2013, we have observed numerous academic research projects and open-source software developments that have refined the design, algorithms, and implementations of Hadoop-GIS. In addition to having received a high number of research citations, the Hadoop-GIS paper has also been cited by more than 20 industry patents for their technical inventions in spatial data analytical systems. Notably, some of these follow-up open-source software applications have matured to the point of being widely adopted and used as production systems. Apache Sedona [16] is a prime example of such a system.

1.2 Hadoop-GIS's Impact

Hadoop-GIS has influenced the design and implementation of both spatial data management and system development in large cluster systems. We will provide an overview of its impact on these two aspects.

1.2.1 Impact on Spatial Data Management. From its inception, the Hadoop-GIS project was designed to efficiently support high-performance queries on large volumes of spatial data using a shared-nothing architecture. One of its key innovations is an on-demand spatial query engine approach, which enables the execution of spatial queries across as many partitions as needed. This marked a paradigm shift from traditional spatial data management systems. Through dynamic on-demand indexing and effective boundary handling, Hadoop-GIS achieves highly scalable spatial queries at extreme scales.

Additionally, Hadoop-GIS was the first system to implement a declarative spatial query language on top of MapReduce through integration with Apache Hive, inspiring various implementations in subsequent years, e.g. [16]. Essentially, Hadoop-GIS demonstrated the potential of combining spatial data analysis with scalable distributed computing frameworks. It demonstrated that high performance and scalability in managing and querying massive spatial data could be achieved without specialized hardware, marking a significant milestone in the evolution of spatial data warehousing systems. The Hadoop-GIS project set a new direction for the development of a new ecosystem in spatial computing and GIS fields.

1.2.2 Impact on System Development. The methodologies, design, and implementation of Hadoop-GIS in its open-source formats of programs and the VLDB 2013 paper have influenced the development of many follow-up open-source systems that are widely used in spatial data processing operations now. Commercial GIS software vendors such as Pitney Bowes have either migrated Hadoop-GIS into their own software packages or developed commercial tools that shared similar methodologies.

Two spatial join algorithms (point-in-polygon and point-to-polyline distance join) were implemented in SpatialSpark [70] with a similar methodology as Hadoop-GIS. SpatialSpark is based on Apache Spark and Apache Impala with a contribution of additional spatial partitioning methods. GeoSpark [71] extended Spark for a generalized set of spatial data types (Spatial RDDs) and utilized the same idea of the Hadoop-GIS's on-demand indexing for accelerating spatial queries for parallel processing in Spark. Magellan [17] is a Spark based open-source library for geospatial analytics, which also provides a SQL interface for spatial queries by extending Spark SQL. It takes the same on-demand indexing approach as in Hadoop-GIS.

Pitney Bowes, known for its MapInfo GIS software and location analytics products, has closely collaborated with Stony Brook University to transfer technologies from Hadoop-GIS and its follow-up system, SparkGIS. This collaboration was facilitated through two industry gifts, summer internships, and the recruitment of one of the original developers of Hadoop-GIS to join the company for continued development. MapInfo Pro location analytics [3] used the same processing engine in updating routing. The Spectrum Spatial software [4] provides location intelligence solution, which manages and queries spatial data in a similar data warehousing and querying architecture as that of Hadoop-GIS, with additional nearest neighbor search methods. Pitney Bowes' presorting business processes over 17 billion mail pieces per year, utilizing matching techniques that employ similar partitioning and parallelization methods as those used in Hadoop-GIS. The dynamic weather services [5] was implemented by extending Hadoop-GIS and our consequent work SparkGIS [22] for spatial-temporal support.

Apache Sedona [16] is an Apache Top Level project evolved from GeoSpark. Apache Sedona is currently the most downloaded open-source software for large scale geospatial data processing. Wherobots, a startup company providing end-to-end geospatial data analytics solutions, is among the major contributors to Apache Sedona. Heavily influenced by Hadoop-GIS, Sedona has been downloaded over 12 million times, and is used by major companies worldwide for their busy spatial data analytical tasks, including Amazon, Apple, Databricks, Meta Platforms, Mercedes-Benz, T-Mobile, Twitter, Uber, and many others.

Esri, known for its widely used GIS software ArcGIS, develops the GeoAnalytics Server [67], which implements spatial and spatial-temporal queries on Apache Spark and is part of the ArcGIS Enterprise platform. It employs the same on-demand local indexing approach as Hadoop-GIS, using an in-memory quadtree instead of an R*-Tree, with additional support for spatial-temporal data.

1.3 On the Role and Structure of This Paper

This paper aims to present the evolution of spatial data analytical ecosystems, and provide a comprehensive overview of our contributions to high-performance spatial data analytics, addressing the following key aspects:

(1) A summary of our high-performance spatial analytics research beyond the Hadoop-GIS project: Over the past decade, our collaborative team has driven advancements in high-performance spatial data analytics. Our project focused on developing scalable solutions that leverage both scale-out and scale-up techniques to handle vast amounts of spatial data efficiently. We introduced efficient algorithms and optimized system frameworks that significantly improved data processing performance and accuracy. We will summarize major milestones, including key publications, software releases, and practical implementations.

(2) Current progress and new challenges in the field: The field of spatial data analytics has evolved rapidly, with significant progress in various areas such as machine learning integration, real-time processing, and cloud-based solutions. However, new challenges have emerged, including handling larger and more complicated datasets, ensuring data privacy and security, and integrating heterogeneous data sources. Computing hardware devices are increasingly customized to best fit different classes of applications, moving away from the general-purpose and one-size-fits-all approach. Therefore, the gap between algorithms design at a logical level and their implementations on complex hardware devices at a physical level increases rapidly. Narrowing this gap is also a critical task in the R&D of high-performance spatial data analytics. We will outline the current state of the domain, highlighting recent advancements and the pressing challenges that researchers and practitioners are facing today.

(3) Lessons learned and experiences to share: Reflecting on our extensive research and development journey, we have gained valuable insights and learned critical lessons. These include the importance of interdisciplinary collaboration, the need for flexible and adaptable frameworks, and the benefits of fostering a strong user community for feedback and improvement. We will share these lessons and experiences to provide guidance and inspiration for future researchers and developers in the field of high-performance spatial data analytics.

By addressing these points, this paper not only commemorates our past achievements but also contributes to the ongoing discourse in the field, offering practical insights and fostering further innovation. The rest of this paper is organized as follows. In Section 2, we discuss the initial motivations behind the development of Hadoop-GIS. Section 3 provides a brief introduction to the Hadoop-GIS software and its performance. In Section 4, we present key advancements that optimize Hadoop-GIS from both systems and algorithms perspectives. Section 5 shifts the focus from scale-out to scale-up studies, detailing two accelerator-based scale-up solutions. Section 6 summarizes the major lessons and experiences gained from developing Hadoop-GIS and related projects. We conclude the paper in Section 7 and offer acknowledgments to our collaborators in Section 8.

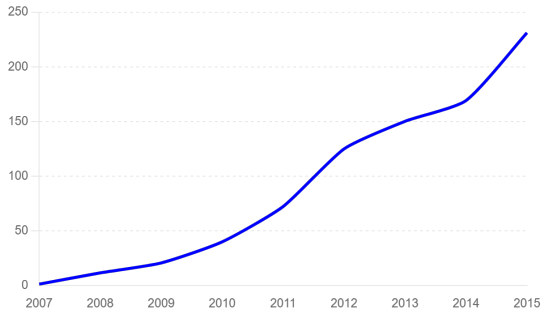


Figure 1: Unit sales of the Apple iPhone worldwide from 2007 to 2015 (in millions) [6]

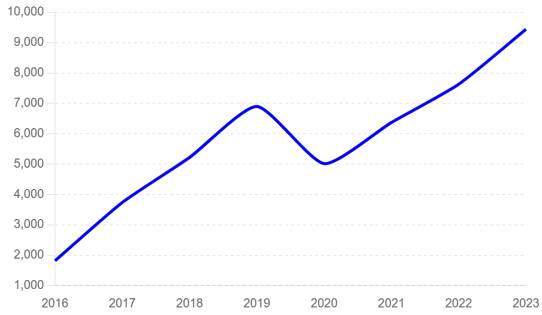


Figure 2: Number of Annual Trips on Uber (2016 - 2023) (in millions) [7]

2 WHY DID WE DEVELOP HADOOP-GIS?

In this section, we will provide an archived summary to explain why we needed to develop a MapReduce-based spatial data analytical solution over a decade ago. The key insight is that this project originated from user requirements in digital pathology and clinical applications. Our solution was developed because no existing tools could meet our unique requirements at the time.

2.1 Explosion of Big Spatial Data

The Hadoop-GIS project was created around 2010, during an unprecedented era in human history when vast amounts of spatial data were being generated at an accelerating pace each year. This explosion of big spatial data presented new challenges to existing database solutions, which struggled to even conduct normal operations, such as to store, query, index, and mining these complex spatial datasets.

One of the primary factors contributing to the surge in big spatial data, as well as big data in other areas, is the rise of mobile internet, driven by the invention and widespread adoption of smartphones. This technological advancement has brought billions of users online, each smartphone equipped with GPS, transforming them into continuous geo-rich data generators, producing geo-tagged tweets, mobile trajectories, and more. Consequently, we have entered an unprecedented era of big spatial data. Additionally, the proliferation of location-oriented services has exponentially increased the volume of spatial data, far beyond what could have been imagined 20 years ago. According to an Esri report, location information is included in 80% of all data in the world [24]. Figures 1 and 2 illustrate the annual growth curves of iPhone sales and Uber trips during a period of rapid expansion, highlighting the challenges of managing the flood of big data and underscoring the critical need for scalable, cost-effective and high-performance infrastructure. Furthermore, GIS data such as remote sensing and satellite imaging, along with tissue imaging data (digital pathology and multiplexed imaging) have contributed significantly to the spatial data deluge. Hadoop-GIS was developed in response to these challenges, embodying a scalability-oriented approach to support the demands of big spatial data applications.

The initial application targeted by Hadoop-GIS is pathology imaging data management, which presents far more challenging tasks than conventional spatial data management systems typically handle. The main reason is that pathology imaging data can be significantly larger than geographic data. To illustrate this size comparison, consider the following example quoted from [33]. *A real dataset extracted from OpenStreetMap representing map data from the whole world contains 164 million polygons.* In contrast, high-resolution microscopy images from digital slide scanners provide extensive information about spatial objects and their features. For instance, whole-slide images (WSI) produced by scanning microscope slides at diagnostic resolution can be immense, often containing 100,000 x 100,000 pixels. Each image can encompass millions of objects, with hundreds of features extractable per object. Studies often involve hundreds to thousands of such images collected from large groups of subjects. For large-scale interconnected analyses, numerous algorithms with various parameters may produce many different result sets that need to be compared and integrated. Consequently, data derived from a single study's images can reach tens of terabytes. A moderately sized hospital might generate thousands of WSIs daily, resulting in several terabytes of derived analytical results per day and potentially accumulating petabytes of data within a year.

2.2 Why Did Parallel Database Solutions Fail

In our original Hadoop-GIS paper, we documented our experiences with the failure of using a commercial parallel database system (DBMS-X) for managing pathology data. We briefly summarize and highlight the two primary and determining reasons why we abandoned the parallel database solution in favor of a MapReduce-based approach. Our intention is not to reignite the long-standing debate between the database and MapReduce paradigms, a topic extensively covered in numerous papers over the past decades [53][55]. Instead, we aim to share our first-hand and practical insights into why MapReduce was necessary for our needs.

Excessively long data loading times. Our experiences have shown that loading the results from a single whole slide image into a Spatial DBMS can range from a few minutes to several tens of minutes. In a medical and clinical research environment, we utilize a broad spectrum of software and algorithms to process an image in various ways. For large-scale, interconnected analyses, numerous algorithms with different parameters can produce a multitude of result sets that need to be compared and integrated. All these results, represented as spatial data (i.e., polygons), must be

loaded into a database system. During the algorithm development stage, various parameter tuning experiments can generate a substantial amount of temporary data requiring analysis. This data deluge ultimately made the data loading time an intolerable factor, hindering our continued use of that database system.

Complex usage and high licensing costs. Using DBMS-X in our environment was challenging due to its complexity in configuration, tuning, and optimization, despite having vendor support and professional customer service engineers. One issue stemmed from the difficulties in configuring various parameters. Additionally, spatial partitioning across nodes in a cluster requires specific techniques and optimizations, such as boundary handling, making performance tuning a tedious task. This is especially problematic given the experimental and research-oriented nature of our pathology applications. Furthermore, the high licensing costs also deterred us from adopting this commercial solution.

Although these two reasons have been summarized in other comparative studies, such as [53], our spatial data-focused scenarios prompted us to reconsider whether the combination of spatial data and a parallel database system is the optimal choice for large-scale spatial analytics. Given these challenges, we turned to MapReduce as the underlying engine to implement a simple-to-extend, simple-to-manage, high-performance spatial solution: Hadoop-GIS.

3 INSIDE HADOOP-GIS

With the goal of building an easy-to-use, highly scalable, and high-performance spatial analytical system, we carefully examined and evaluated various design principles and optimization techniques in the Hadoop-GIS system. In this section, we first provide an overview of the most important aspects of Hadoop-GIS, followed by a brief performance summary. Detailed implementation and evaluation results can be found in the original paper [19].

3.1 A Trinity of Architecture, Engine, and Interface

Architecture. Figure 3 illustrates the overall architecture of Hadoop-GIS. In essence, SQL-based spatial queries are executed by the MapReduce-based execution engines on partitioned datasets stored in the Hadoop Distributed File System (HDFS). The engine, named RESQUE (Real-time Spatial Query Engine), is implemented in C++ as a shared library to ensure high performance. We extended Hive with spatial functionalities to parse and optimize user SQL queries, which then invoke the MapReduce framework and the RESQUE engine.

Engine. The RESQUE engine is designed to achieve high performance in both data loading and query execution. Unlike conventional relational data, mapping spatial data processing onto the MapReduce framework presents unique challenges due to the complexities of geometric objects. We highlight three key techniques in the engine:

- **On-Demand In-Memory Indexing and Engine Execution** A critical contribution of Hadoop-GIS is the on-demand based RESQUE query engine. Instead of loading data and creating indexes at the data loading stage, Hadoop-GIS "eliminates" this step by proposing an on-demand based method with the observation that the indexing time is negligible with fast growing computer power for complex spatial queries. What Hadoop-GIS does is to keep only "global indexing" at the region level; and at the object level, no indexing will be created at the data loading stage. Original spatial data are stored in HDFS as they are after loading. When a query is submitted, the RESQUE engine will be launched to query datasets of interest through preliminary spatial filtering based on data partitioning, and on-demand in-memory indexing for objects in the same partition. This significantly reduces the data loading time, and makes it possible for scaling out spatial query processing through executing as many instances of RESQUE engine as needed.
- **Recursive Spatial Data Partitioning.** To address data skew, where same-sized tiles can contain different amounts of spatial objects, Hadoop-GIS recursively splits tiles into smaller ones by selecting an optimal direction based on a threshold C_{max}, which represents the maximum number of objects allowed in a tile. As effective partitioning can directly impact the queries, we have developed and evaluated partitioning methods systematically [60], as discussed later in Section 4.2.
- **Multi-assignment Boundary Handling.** Due to partitioning, spatial objects often intersect tile boundaries, which creates a challenge for all parallel spatial processing applications. Hadoop-GIS takes a simple yet elegant approach by replicating these boundary-intersected objects into adjacent tiles and uses a post-normalization step to remove duplicates. This technique ensures the correctness of query results with minimal overhead.

Interface. The spatial SQL interface of Hadoop-GIS is a key feature that allows it to be used like a relational database, rather than a programmable MapReduce framework. This interface incorporates major operators and functions from ISO SQL/MM Spatial, ensuring familiarity and ease of use for users. We enriched Hive with an extensive set of spatial features, extending the query language, query optimizer, and query engine to support a wide range of spatial data types, spatial functions, and spatial query operators (e.g., spatial relationship comparisons including intersects, touches, overlaps, contains, within, disjoint). Additionally, we implemented several spatial data accessing methods for efficient query processing, such as R*-Tree [23], Hilbert R-Tree [42], and Voronoi Diagram [20].

3.2 Performance Overview

We re-report Hadoop-GIS's performance evaluation results under two setups, as presented in the original paper. Our aim in presenting these results again is to provide convincing evidence that Hadoop-GIS effectively addressed our pathology application challenges at tha

Figure 4 shows the performance of the Hadoop-GIS RESQUE engine in executing single-node data loading, indexing, and spatial join query execution, compared to PostGIS and DBMS-X. Data loading is inherently parallelizable, meaning each node can independently load its assigned datasets. Therefore, the performance of single-node loading can accurately represent the overall loading

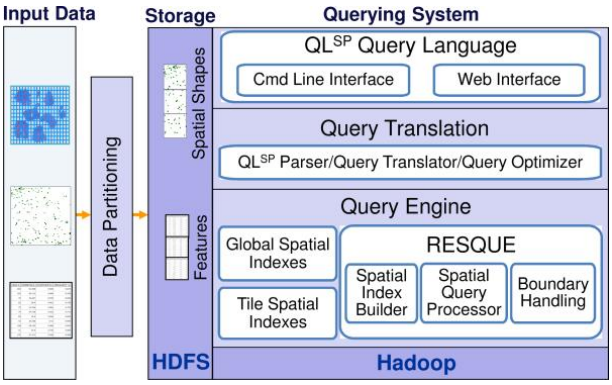


Figure 3: An overview of the Hadoop-GIS software system.

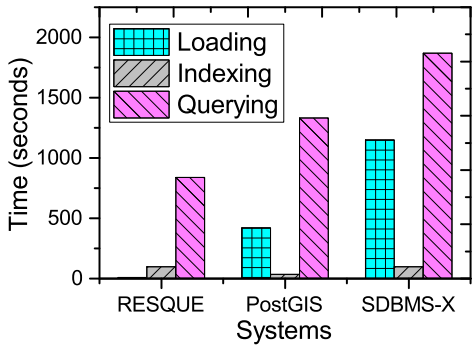


Figure 4: Standalone performance comparison of data Loading, index building, and query execution times across three different spatial data analytical engines

performance. As shown in Figure 4, RESQUE's loading time is minimal compared to the other database solutions. The dataset in Figure 4 represents a single image. Given the long loading times of the two database solutions, loading multiple images could easily become a multi-hour or even multi-day task, causing significant delays in the entire application pipeline due to unfinished data loading.

Figure 5 shows the query execution performance of Hadoop-GIS and DBMS-X when executing the same spatial join query, with the number of parallel units increasing from 5 to 30. It is evident that Hadoop-GIS demonstrates a performance advantage over the parallel database solution for this complex query. Our original paper also indicated that for other simpler queries, Hadoop-GIS exhibited similar or even lower performance compared to the database solution. However, the performance gap and absolute query execution times for these simpler queries were not very critical. Nonetheless, for join-heavy queries, which are crucial for our pathology applications, Hadoop-GIS clearly outperformed the traditional database solution.

Our conclusion from the performance results is that Hadoop-GIS met our application requirements by (1) significantly reducing data loading times and complex query execution to manageable levels, and (2) ensuring that any performance loss in simpler queries is tolerable and outweighed by the benefits in other critical aspects. While Hadoop-GIS can be seen as a specially optimized system for aspects crucial to our pathology application, the general philosophy holds true: properly curated data enables rapid analysis. This approach has proven effective in real-world applications.

4 CONTINUED ADVANCEMENT

In this section, we will provide an overview of our further development and ongoing progress in Hadoop-GIS. This includes upgrading the system to an in-memory computing infrastructure, utilizing RDMA for low-latency remote memory access, advanced spatial data partitioning, improved methods for handling complex polygons, and transitioning spatial data processing from 2D to 3D.

4.1 Further Scale-Out by In-memory Computing

Building on the foundation laid by Hadoop-GIS, our subsequent research efforts have continued to advance the scale-out paradigm by leveraging new system advancements in underlying systems. We have focused on enhancing distributed spatial data processing by in-memory computing and high-performance networking technologies. By integrating systems like Apache Spark and RDMA, we have developed solutions that significantly improve the efficiency, scalability, and performance of spatial queries, addressing the limitations of earlier systems and keeping pace with the evolving landscape of big data processing.

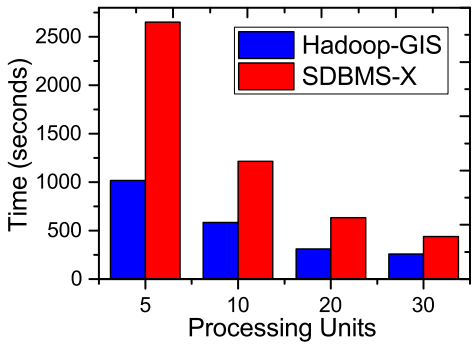


Figure 5: Spatial join performance comparisons between Hadoop-GIS and DBMS-X while increasing the number of parallel units

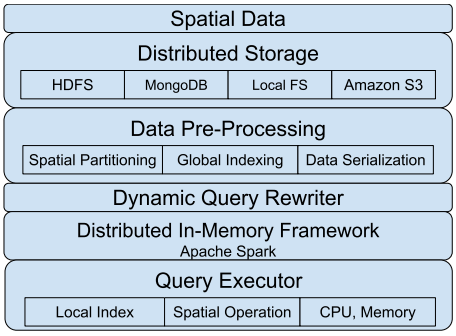


Figure 6: An overview of the SparkGIS system

Our project SparkGIS [22] focuses on the need for in-memory processing due to the high inter-job data movement costs associated with disk I/O in traditional Hadoop-based systems, which is inefficient for iterative spatial data processing. Apache Spark is chosen for its documented 10x to 100x performance gains over Hadoop due to in-memory processing capabilities and efficient handling of iterative tasks. The major design of SparkGIS includes resource-aware query rewriting, customizable spatial data partitioning, and multilevel in-memory indexing to ensure efficient query processing and optimal memory usage. In particular, SparGIS proposes dynamic query rewriting to mitigate the memory shortage problem for resource-intensive query pipelines. SparkGIS significantly outperforms existing Spark-based spatial frameworks in handling large datasets and memory-intensive workflows, providing higher throughput and lower latency, especially under limited resource settings.

Another spatial data analytics project is Catfish [68, 69], an RDMA-enabled R-tree platform designed to optimize spatial data processing in distributed systems. It addresses performance bottlenecks in R-tree processing caused by imbalanced workloads between server and client CPUs and network bandwidth. Catfish employs two RDMA mechanisms: fast messaging for low-latency queries and RDMA offloading to distribute tree traversal workloads to clients. An adaptive scheme dynamically switches between these methods to balance server load and network usage. Experimental results demonstrate that Catfish significantly outperforms conventional R-tree implementations using TCP/IP and other RDMA methods in both latency and throughput, especially in scenarios with high query volumes and diverse workload distributions.

4.2 Statistical- and Cost-Model based Spatial Partitioning Optimization

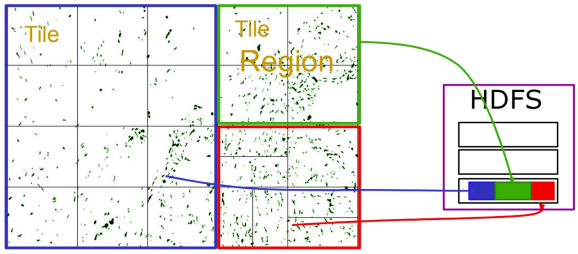


Figure 7: The SATO framework for partitioning optimization

Spatial partitioning determines how spatial objects are clustered for local processing, playing a crucial role in query efficiency in distributed spatial data processing. In [61], we proposed the *SATO* framework to optimize spatial data partitioning. With SATO, partitioning a large spatial dataset is executed in four phases. In the sampling phase, a subset of objects is sampled for subsequent optimizations. During the analysis phase, the statistics of the sampled subset are analyzed, and a global partitioning schema is generated. In the tear phase, the global partition is further refined, and the entire dataset is partitioned for local processing. Finally, in the optimization phase, the statistics collected during the previous phase are used to map the tiles in the partitioning schema to the files stored in distributed file systems such as HDFS. As shown in Figure 7, a global spatial index can be built after one round of processing.

4.3 Improved Methods for Complex Polygons

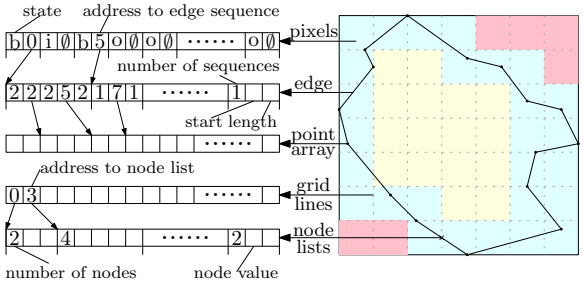


Figure 8: The data structure of the IDEAL hybrid model

A critical issue in processing real-world polygons is the significant variation in complexity, measured by the number of edges. For example, the edge count of polygons in the OpenStreetMap (OSM) dataset ranges from 3 to over 400, 000 [35] The time complexity for determining spatial relationships between objects can be $O(N)$, $O(NlogN)$, or even $O(N^2)$, where N is the number of edges. The presence of complex objects not only increases the overall geometric computation but also causes a long-tail issue across processing units, as tiles with more complex objects require significantly more time to process. To address this issue, we proposed IDEAL, a Hybrid Vector Raster Model to represent spatial objects [56, 57]. As illustrated in Figure 8, in IDEAL, each vector-based polygon is indexed with the corresponding raster model. With as low as a 10% storage overhead, the IDEAL representation reduces the time complexity from $O(N)$ to $O(C)$ for ray casting and from $O(N^2)$ to $O(N)$ for line segment pair evaluations. This significantly enhances the efficiency of Hadoop-GIS and SparkGIS in evaluating spatial datasets with complex objects. As shown in Figure 9, replacing the GEOS library in Hadoop-GIS with IDEAL in the refinement phase significantly reduces overall latency across all processing units, effectively solving the long-tail issue.

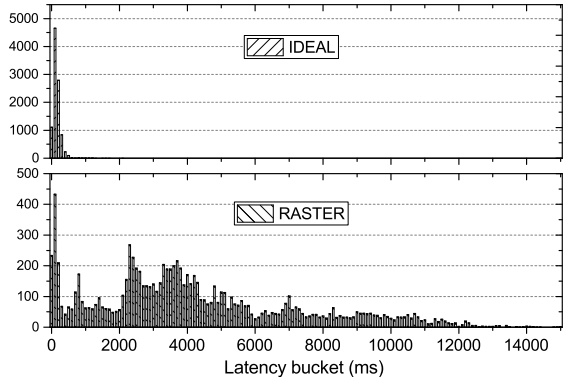


Figure 9: The efficiency of IDEAL in supporting distributed spatial data processing

4.4 Moving from 2D to 3D

With the development of 3D data modeling techniques and massive 3D data generated from human tissues and 3D GIS, the management of 3D spatial data has become increasingly important [29]. 3D data is further challenged by the increased complexity from complex 3D structures such as bifurcation. To support large-scale 3D spatial data management, we proposed *iSPEED*, an efficient and scalable spatial query processing system for large-scale 3D data [48, 59, 62]. Figure 10 provides an overview of its software system. iSPEED inherits the MapReduce-based distributed spatial data processing paradigm from Hadoop-GIS. It facilitates spatial partitioning by distributing spatial objects in different regions into various processing units and duplicating boundary objects to ensure query correctness. Furthermore, iSPEED addresses the unique challenges of processing 3D spatial data, particularly the complexity of spatial object representation models, through multi-level spatial indexing including inter-object indexing and intra-object indexing - structural indexing, in-memory spatial query engine, and 3D data compression.

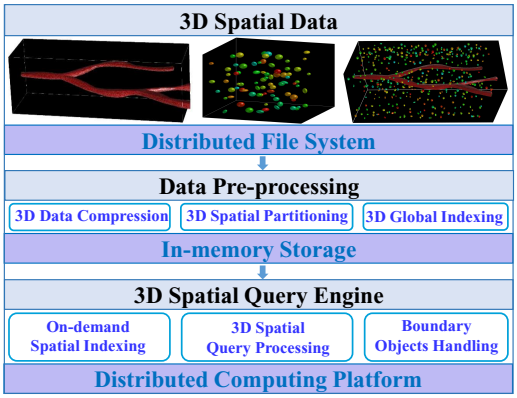


Figure 10: An overview of the iSPEED system

Polyhedrons are used to represent spatial objects in 3D space. Compared to polygons used in 2D space, polyhedrons better capture the details of spatial objects in 3D space with much more complex representations. As a result, processing polyhedrons requires more computationally intensive geometric operations. iSPEED addresses this issue by balancing query efficiency and accuracy through two approaches: skeleton-based approximation to build structural indexes and geometry simplification-based approximation. The skeleton-based approach approximates spatial objects with complex shapes using skeleton points to estimate distances between spatial objects, which are then used to determine spatial relationships. iSPEED adapts the Mean Curvature skeleton algorithm to extract the skeleton points [27]. The geometry simplification-based approach exploits simplification algorithms to reduce the complexity of spatial objects into polyhedrons with multiple resolutions, known as levels of detail (LOD). As shown in Figure 11, low-LOD polyhedrons represent the same spatial objects with significantly fewer surface elements, thereby reducing the geometric computations required for processing. This method is particularly

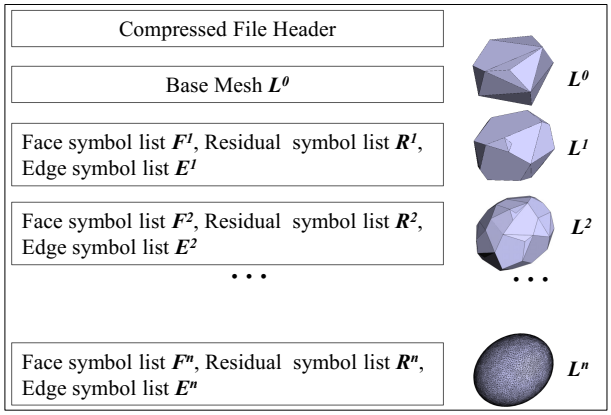


Figure 11: The storage of multi-LOD polyhedrons in a single compacted format

useful for querying scenarios where a tradeoff of accuracy for speed is necessary.

iSPEED relies on approximations to balance query accuracy and efficiency, which may not be suitable in scenarios requiring strict query accuracy. In response, we propose the 3DPro system, designed for accurate querying using data with lower level of details (LOD) from from compressed spatial data [58]. It introduces a customized geometric simplification algorithm called Progressive Protruding-Vertex Pruning (PPVP), which selectively prunes vertices during the decimation process to ensure that the simplified low-LOD representation covers a subset of the original spatial objects. These low-LOD polyhedrons serve as progressive approximations for filtering. Building upon the PPVP algorithm, 3DPro proposes a filter-progressive-refine paradigm where polyhedrons at increasing levels of detail are evaluated iteratively as needed, guaranteeing correct results are returned while minimizing computation.

5 EMERGING SCALE-UP BY ACCELERATORS

The traditional scale-up method adheres to a fundamental principle of multilevel abstraction in computer architecture and system development. Each level of abstraction conceals the intricate operations of the level below it, forming a hierarchical stack that facilitates increasingly detailed operations from top to low levels. For example, in a conventional data processing system, SQL is used at the top of this stack to allow users without programming training to make data processing requests. These requests traverse multiple software levels, ultimately reaching the bottom level where they are executed through detailed hardware operations by complex processors and a deep memory hierarchy. Computer software and hardware architects focus on improving their respective levels, collectively creating a one-size-fits-all, application-independent ecosystem where billions of users can easily run their application programs.

However, this general-purpose computing ecosystem has faced increasing challenges related to performance efficiency, power efficiency, and scalability. The emerging scale-up approach involves

building specialized computing devices tailored to specific applications, such as GPUs, FPGAs, TPUs, and ray tracing (RT) cores. Meanwhile, the demand for low-latency, real-time, and highly flexible data processing for various applications, including spatial data analytics, is on the rise. In this section, we will present two case studies highlighting our new scale-up efforts for spatial data processing applications.

5.1 A GPU Scale-up Case: PixelBox

Although there is ongoing debate in the chip industry about whether Moore's Law is truly dead, advancements in semiconductor manufacturing technologies over the past decades have enabled the production of significantly smaller transistors. For instance, Apple's A7 chip, launched in 2013, contained 1 billion transistors manufactured using a 28nm process. In stark contrast, the Apple A17 chip, launched in 2023, boasts 19 billion transistors produced with a 3nm process [9]. This dramatic increase in transistor density exemplifies the progress in chip manufacturing. However, it is an established fact that Dennard Scaling ceased to be effective before the 2010s. This means that smaller transistors alone cannot improve frequency and, thus, do not directly enhance single-thread performance. Therefore, the only way to achieve proportional performance improvements is by increasing parallelism, leading to the development of many-core CPUs and massively parallel GPUs.

Figure 12 shows the maximum number of GPU shaders for AMD and NVIDIA over the past two decades. A shader, akin to a NVIDIA CUDA core, is a fundamental unit for GPU parallel computing. Significant performance gains for applications running on such parallel devices require maximizing the parallelism inherent in the problems and algorithms supporting these applications. The parallel hardware growth curve evidently illustrates the scale-up opportunities available to applications that can perfectly align their parallelism with the hardware capabilities. However, designing and implementing GPU parallel algorithms remains a challenging task.

Our first scale-up effort is a GPU-based solution for the pathology cross-matching problem. This project was conducted concurrently with the Hadoop-GIS project for the same objective with a different approach. We will briefly present the motivation, core algorithm ideas and major performance results, while leaving detailed explanations to a VLDB 2012 paper [65].

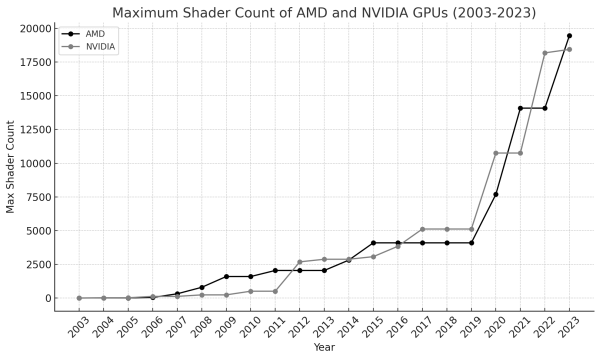


Figure 12: Maximum Shader Count of AMD (ATI before 2006) and NVIDIA GPUs (2003-2023) [8]

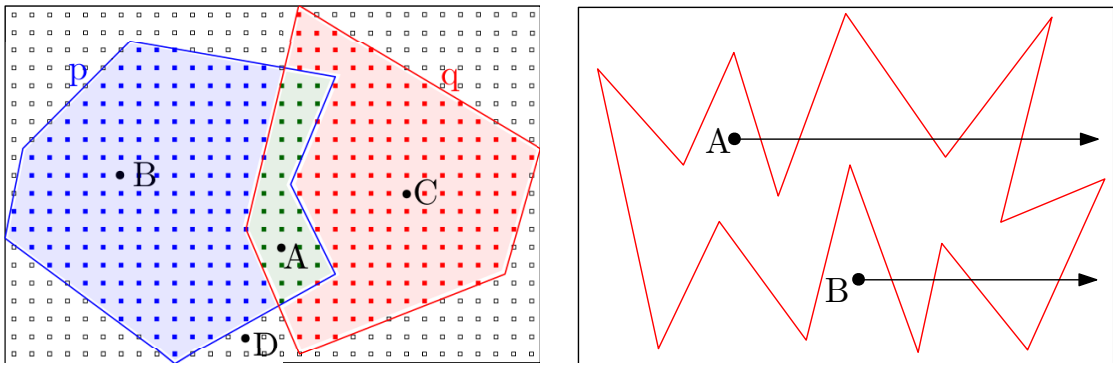
5.1.1 Performance Profiling: Where Does Time Go? The core challenge was the development of efficient pathology imaging analysis algorithms. Specifically, we need fast methods to cross-compare millions of spatial boundaries of segmented micro-anatomic objects using the Jaccard Similarity metric. This metric fundamentally requires calculating the ratio of the intersection area to the union area between two groups of polygons. The SQL can be written as below.

```
SELECT AVG(ratio)
FROM (
    SELECT
        ST_Area(ST_Intersection(p.the_geom, q.the_geom)) /
        ST_Area(ST_Union(p.the_geom, q.the_geom)) AS ratio
    FROM
        dataset1 AS p, dataset2 AS q
    WHERE
        ST_Intersects(p.the_geom, q.the_geom)) AS tmp
WHERE ratio > 0;
```

The above query can be accelerated and optimized by testing MBR (Minimum Bounding Rectangle) intersections using indexes and by indirectly calculating the union area. Consequently, one core calculation is the combo operator $ST\_Area(ST\_Intersection())$, which calculates the intersection area. Our performance profiling results in PostGIS indicate that 90% of execution time is spent on this operator, while the remaining 10% is used for index building, search, and other area calculations. Therefore, our research focus is on how to accelerate this combo operator using GPUs.

5.1.2 The PixelBox Algorithm. We chose not to parallelize the conventional sweep line algorithm in computational geometry for calculating polygon intersections due to its sequential nature and branch-intensive complexity in handling various situations. Instead, aiming to achieve optimal parallelism, we developed the PixelBox algorithm for GPUs to calculate the intersection area. The core idea, illustrated in Figure 13, is to independently test a point's position relative to a polygon. Intuitively, the intersection area correlates with the number of points that fall within both polygons. More importantly, the position testing for each point is independent of other points. Using a ray casting approach, as illustrated in Figure 13, creates an execution pattern for numerous independent points, enabling maximal parallelism. The accuracy of this design relies on the rectilinear nature of polygons extracted from medical images, which have axis-aligned edges and integer-valued vertices. As its name implies, PixelBox includes a major optimization that uses sampling boxes covering multiple points (pixels) to eliminate redundant point-in-polygon tests, switching to point tests only on boundary boxes. The algorithm is ultimately integrated into a CPU-GPU hybrid framework for the entire workflow of spatial cross-comparison.

As an initially customized solution for pathology imaging, the PixelBox algorithm needs to be extended for more complex geographic data due to its limitations in handling geospatial polygons with floating-point coordinates and varying orientations. To address these limitations and make PixelBox a general solution, our subsequent research [21] introduces an adaptive scaling strategy



(a) A pixelized view of polygon intersection and union. **(b) Determining a pixel's position relative to a polygon.**

Figure 13: Illustrating the core idea of using ray casting to execute numerous independent point-in-polygon tests.

that converts floating point-valued vertices into integer-valued vertices, making it suitable for GPU processing using a pixelization method. The paper discusses a CPU-GPU hybrid platform, Geo-PixelBox, designed to accelerate the spatial cross-matching of general geospatial datasets using CUDA and OpenACC.

5.1.3 Performance and Impact. The PixelBox solution demonstrated a strong performance advantage over PostGIS in our 2012 comparison using real-world pathology data. On a heterogeneous platform featuring an Intel Core i7 860 CPU and an NVIDIA GTX580 GPU (costing $820), PixelBox achieved a 13-44x speedup compared to a manually-tuned, parallelized PostGIS solution running on two Intel X5570 CPUs (costing over $2000).

Fixstars Solutions Inc. [10] has developed the Geometric Performance Primitives (GPP) Library [11], a high speed computational geometry engine that incorporated the PixelBox algorithm. The GPP Library is a part of the NVIDIA Developer community [12], and has been included in the NVIDIA's suite of GPU-accelerated libraries.

The PixelBox solution exemplifies a typical scale-up strategy, enabling automatic performance improvements by upgrading to faster hardware devices without modifying software or applications. As shown in Figure 12, between 2012, when the PixelBox paper was published, and 2023, the number of cores in a single GPU device increased from 2,688 cores (NVIDIA Tesla K20X) to 19,456 cores (AMD Radeon Instinct MI300X). This means GPU capacity has automatically increased more than seven-fold for the same PixelBox algorithm in GPP. Effective utilization of massive parallelism is crucial for developing and implementing efficient scale-up solutions for algorithms. Notably, the PixelBox effort ignited a series of subsequent research projects aimed at mapping database queries to advanced hardware. Our major publications on this topic include [34, 46, 47, 66, 72, 73].

5.2 An RT Core Scale-up Case: RayJoin

In the domain-specific architecture era, GPUs have evolved into heterogeneous devices containing multiple cores that serve different applications. Exploiting these new hardware capabilities brings both performance opportunities and technical challenges for application and algorithm designers. Figure 14 illustrates an example

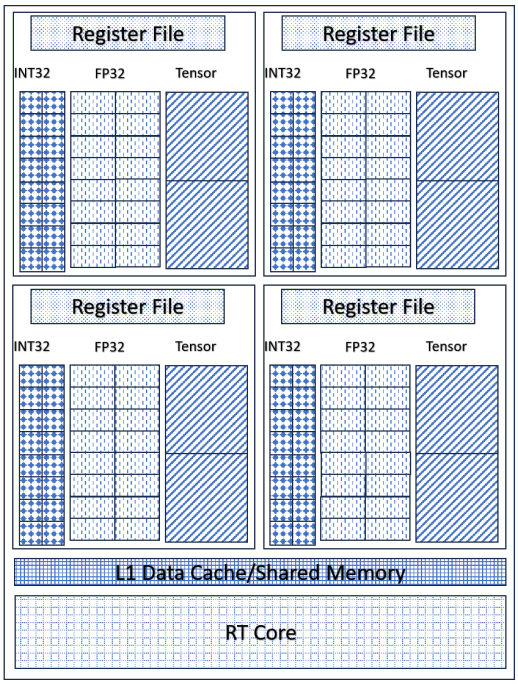


Figure 14: Illustration of the NVIDIA Turing architecture, showing the composition of a Streaming Multiprocessor (SM).

of the NVIDIA Turing architecture, showing the composition of a Streaming Multiprocessor (SM). This architecture exemplifies how modern GPUs integrate diverse cores, such as the RT Core, to accelerate ray tracing and other specific tasks, offering substantial performance gains while introducing new complexities in system design and optimization.

Exploiting new hardware for spatial problems by re-purposing its usage is both potentially beneficial and risky. In this section, we will briefly introduce how ray-tracing hardware can be harnessed to accelerate spatial join operations. For detailed explanations and comprehensive results, please refer to our original paper [36].

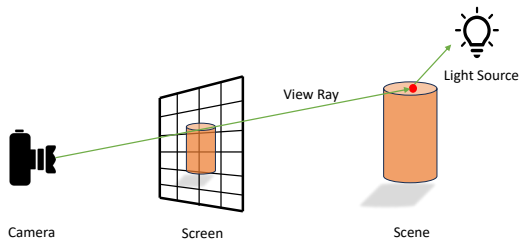


Figure 15: Ray Tracing Illustration.

5.2.1 What is Ray Tracing? Ray tracing is a technique primarily used in gaming and rendering to simulate how light interacts with geometric primitives, producing highly realistic images. It works by tracing the paths of rays from the camera through pixels in an image plane and simulating their interactions with the primitives in a scene, as shown in Figure 15. The core operation involves computing ray-primitive intersections, which can be computationally expensive. To enhance performance, a Bounding Volume Hierarchy (BVH) tree is used to accelerate the intersection finding by narrowing down the search space. The RT core is crucial in this process, executing the performance-critical task of BVH traversal and determining whether a ray intersects a primitive. The BVH on the RT core, akin to a hardware R-tree, manages the relationships between multiple shapes, allowing for quick filtering of unnecessary shapes during traversal. This hierarchical organization significantly reduces the number of intersection tests, making ray tracing efficient for real-time rendering and potentially beneficial for spatial data processing tasks. Recent advancements have expanded the use of ray tracing hardware to general-purpose applications, offering a powerful environment for diverse computational tasks.

5.2.2 RayJoin: The Algorithm and Implementation. RayJoin leverages ray tracing hardware to perform spatial join operations. The core idea involves re-purposing the ray-primitive intersection capabilities of ray-tracing hardware to efficiently compute spatial joins between geometries. The implementation of RayJoin incorporates several core techniques:

Algorithms for Converting Spatial Join into Ray Tracing: RayJoin formulates spatial join problems as ray tracing tasks by transforming key spatial join queries into RT-friendly algorithms. Specifically, it focuses on two critical spatial join queries: line segment intersection (LSI) and point-in-polygon (PIP) tests. These queries are converted into ray tracing operations using the AnyHit and ClosestHit shaders supported by RT Cores. For instance, in the case of LSI, rays are cast to detect intersections between line segments from different datasets, leveraging the BVH structure for efficient traversal and intersection testing.

Bridging the Precision Gap between Hardware Capabilities and Application Demands: One of the significant challenges in using RT Cores for spatial joins is the hardware's limited precision, typically supporting only single-precision floating-point (FP32). However, GIS applications often require higher precision. RayJoin addresses this by employing a conservative representation method that ensures exact query results. This method builds the BVH in low precision Axis-Aligned Bounding Box (AABB) but applies adjustments to maintain high precision in computations, thereby meeting the stringent requirements of GIS applications.

System Optimization for Significant Overhead Reduction: RayJoin implements several optimizations to minimize computational and memory overhead. One key optimization is the introduction of an adaptive grouping technique that significantly reduces the BVH construction cost. By grouping spatially close line segments into the same AABB, the system reduces the number of primitives used in the BVH, thereby decreasing construction time and memory consumption while maintaining high query performance. Additionally, RayJoin combines the logic of intersection testing and result collection into a single shader, eliminating the invocation overhead associated with separate AnyHit and ClosestHit shaders.

RayJoin's effective approach and optimizations ensure efficient and high-performance spatial join processing, leveraging the advanced capabilities of modern ray tracing hardware.

5.2.3 Performance Analysis for the RT Scale-up. In [36], we have reported the detailed performance advantages of RayJoin compared to the best existing solutions utilizing different algorithms (plane-sweep, tree-based, grid-based, and learned spatial index) and hardware choices (CPU and GPU) for underlying PIP and LSI queries, as well as complex polygon overlay computations. These comparisons were conducted using open and real-world datasets, such as the spatial data of water areas in North America and worldwide lakes/parks. The overall conclusions of the performance comparison are twofold: (1) RayJoin achieves performance speedups ranging from 2.1x to 22.2x over the best existing solutions. (2) RayJoin can execute all queries at a sub-second level, enabling real-time spatial join execution, a capability unattainable by previous solutions.

Table 1: RT Core Count Across Different NVIDIA GPU Generations. Note: The performance of each RT Core has also increased with each generation, not just the core count.

Microarchitecture	Year	GPU Name	RT Core Count
Turing [49]	2018	TU102	72
Ampere [50]	2020	GA102	84
Ada Lovelace [51]	2022	AD102	144

We highlight that RayJoin represents a new scale-up solution for high-performance spatial analytics, fundamentally leveraging the performance advantages provided by the number of RT Cores equipped in a GPU. Table 1 shows the RT Core counts across three generations of NVIDIA GPUs since the inception of hardware ray-tracing solutions. This table also illustrates a trend of increasing hardware-level parallelism over the years, similar to the evolution of GPU cores. By utilizing RT Cores, RayJoin can achieve automatic performance gains through hardware upgrades alone, without the need for algorithm or software redevelopment.

6 LESSONS AND INSIGHTS

In this section, we summarize our major experiences in tackling high-performance spatial data analytical problems since the inception of the Hadoop-GIS project over a decade ago. We present several key lessons learned, highlighting the insights and effective strategies that have guided our efforts. The points are ordered by the importance we have observed through our experiences.

Lesson 1: Targeting Real-World Problems. The Hadoop-GIS project and its related research efforts were initially focused on solving the pathology imaging data management task. A major advantage of conducting basic research on actual problems is the immediate recognition of the gap between existing academic efforts and real-world requirements. Moreover, real-world requirements provide a concrete goal, allowing for phase-by-phase verification and validation of research results. This approach ensures that each phase of research is both feasible and practical, leading to more effective and impactful outcomes.

Lesson 2: Simplicity in Implementation vs. Interface. A long-standing debate in computer software design is how simplicity should be understood and enforced in both interface and implementation aspects. This debate has been framed historically as the New Jersey style versus the MIT approach [13, 14]. Our experience in building Hadoop-GIS taught us to quickly deliver a usable system for users. Thus, the implementation simplicity should be prioritized. Hadoop-GIS utilized SQL as the interface to widely attract users, following the design principle of using C++ to integrate existing spatial libraries. This approach allowed us to rapidly develop the system to an operational state.

Lesson 3: Balancing Load, Index, and Query Execution. Different applications assign varying levels of value to the data loaded into a database, ranging from highly valuable to one-time use or somewhere in between. Traditional database solutions often invest heavily in data loading processes, including extensive indexing, which may not always be necessary. Hadoop-GIS addresses this issue by implementing multiple design factors, such as its two-level and on-demand tile indexing. This approach effectively balances the conflicting goals of fast data loading and efficient query execution.

Lesson 4: Deep Customization before Generalization. The era of one-size-fits-all system solutions is gradually coming to an end. Regardless of our preference, achieving computing performance gains today requires a deep integration of application requirements and available hardware features. The Hadoop-GIS system, our PixelBox solution, and the later RT solution of RayJoin were all built on an insightful understanding of the specific application needs and the capabilities of the underlying system. This deep customization ensures that the solutions are finely tuned to leverage hardware features effectively, resulting in superior performance. It is more evident in this domain-specific architecture era that vertical customization inspires new architecture designs based on application needs, as many recent R&D results show.

Lesson 5: No Conflicts between Scale-out and Scale-up. Effective performance optimization requires leveraging both horizontal (scale-out) and vertical (scale-up) scaling strategies. Hadoop-GIS exemplifies this by combining distributed computing capabilities with powerful hardware acceleration. Scale-out strategies, such as distributed data processing across multiple nodes, ensure that the system can handle large datasets efficiently. Simultaneously, the new scale-up approach that involves utilizing advanced GPUs for intensive computations, enhances performance for complex tasks. This two-dimensional scaling approach maximizes resource utilization and ensures robust, high-performance solutions.

7 CONCLUSION

In this paper, we have presented over 10 years of research and development efforts dedicated to building high-performance spatial data analytical systems. Driven by the initial requirements of clinical researchers and practitioners, we started the Hadoop-GIS project to develop scale-out solutions and promptly began our GPU-based acceleration studies to create scale-up solutions. Our efforts have played critical roles in establishing a useful and effective ecosystem for both users and academic researchers, inspiring subsequent research and development work. Our experience in solving the pathology imaging problem exemplifies how a concrete software and hardware solution must evolve in tandem with the rapid development of IT infrastructures, seizing potential disruptive opportunities to implement applications with each new generation.

At the end of the paper, we would like to quote a sentence from Dr. Jim Gray who was a pioneer in the field of computer systems. During an interview with Professor David Patterson on the evolution of storage and databases [52], Dr. Jim Gray gave the following simple rule for system development.

> At a certain point, most people who bought the relational stuff; they bought it for the usability, not for the price performance. They were getting new applications, and they wanted to get their applications up quickly.

This quote aligns with the design and implementation principles of the Hadoop-GIS project and our other data processing initiatives.

ACKNOWLEDGMENTS

Ablimit Aji, Qiaoling Liu and Hoang Vo are three co-authors of the original Hadoop-GIS paper, who were graduate students at Emory University in 2013. We thank for their early contributions in the project. Inspired by the Hadoop-GIS project, multiple Ph.D. students completed their Ph.D. dissertations in the areas of high-performance data processing. These students include Ablimit Aji from Emory University; Hoang Vo, Yanhui Liang, Furqan Baig and Dejun Teng from Stony Brook University; and Yin Huai, Kaibo Wang, Yuan Yuan, and Sofoklis Floratos from The Ohio State University. We would like to thank Professor Mohamed Mokbel at University of Minnesota for his valuable suggestions to our R&D efforts in high-performance spatial data analytics. We appreciate Dr. Jia Yu, a co-founder of Whereobots Inc., for his dedicated efforts and contributions in transforming open-source software Sedona into a production system, making it widely usable around the world.

Our research efforts have been supported in part by the National Science Foundation in grants of IIS-1350885, ACI-1443054, CCF-1629403, IIS-1718450, CCF-2005884, CCF-2210753, OAC-23105010, and CCF-2312507 and National Institutes of Health in grant of U01CA242936. We are also grateful for several industrial gifts to support our research from eBay, IBM, OpenStack, Pitney Bowes, and Samsung.

REFERENCES

[1] 2011. Hadoop-GIS. https://github.com/StonyBrookDB/hadoopgis
[2] 2024. https://hive.apache.org/
[3] 2024. https://www.precisely.com/product/data-integrity/precisely-data-integrity-suite/spatial-analytics
[4] 2024. https://www.precisely.com/product/precisely-spectrum-spatial/spectrum-spatial
[5] 2024. https://www.precisely.com/data-guide/products/dynamic-weather-data
[6] 2024. https://www.statista.com/statistics/276306/global-apple-iphone-sales-since-fiscal-year-2007/
[7] 2024. https://backlinko.com/uber-users
[8] 2024. https://www.techpowerup.com/gpu-specs/
[9] 2024. https://en.wikipedia.org/wiki/Transistor_count
[10] 2024. https://www.starsolutions.com/
[11] 2024. https://developer.nvidia.com/geometric-performance-primitives-gpp/
[12] 2024. https://developer.nvidia.com/
[13] 2024. https://en.wikipedia.org/wiki/Worse_is_better
[14] 2024. https://www.dreamsongs.com/WIB.html
[15] 2024. Apache Hadoop. https://hadoop.apache.org/.
[16] 2024. Apache Sedona. https://sedona.apache.org/1.6.0/
[17] 2024. Magellan: Geospatial Analytics Using Spark. https://github.com/harsha2010/magellan
[18] 2024. VLDB Test of Time Award. https://www.vldb.org/awards_10year.html
[19] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. 2013. Hadoop-GIS: A high performance spatial data warehousing system over MapReduce. In *Proceedings of the VLDB endowment international conference on very large data bases*, Vol. 6. NIH Public Access.
[20] Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* 23, 3 (1991), 345–405.
[21] Furqan Baig, Chao Gao, Dejun Teng, Jun Kong, and Fusheng Wang. 2020. Accelerating spatial cross-matching on cpu-gpu hybrid platform with cuda and openacc. *Frontiers in big Data* 3 (2020), 14.
[22] Furqan Baig, Hoang Vo, Tahsin Kurc, Joel Saltz, and Fusheng Wang. 2017. Sparkgis: Resource aware efficient in-memory spatial query processing. In *Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems*. 1–10.
[23] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. 322–331.
[24] Lauren Bennett and Trisalyn Nelson. 2022. Five Reasons Every Data Science Team Needs a Geographer. https://www.esri.com/about/newsroom/arcuser/five-reasons-every-data-science-team-needs-a-geographer/
[25] Michael Beynon, Chialin Chang, Umit Catalyurek, Tahsin Kurc, Alan Sussman, Henrique Andrade, Renato Ferreira, and Joel Saltz. 2002. Processing large-scale multi-dimensional data in parallel and distributed environments. *Parallel Comput.* 28, 5 (2002), 827–859.
[26] Michael D Beynon, Tahsin Kurc, Umit Catalyurek, Chialin Chang, Alan Sussman, and Joel Saltz. 2001. Distributed processing of very large datasets with DataCutter. *Parallel Comput.* 27, 11 (2001), 1457–1478.
[27] CGAL. 2024. *The Computational Geometry Algorithms Library*. http://www.sfcgal.org/
[28] Chialin Chang, Anurag Acharya, Alan Sussman, and Joel Saltz. 1998. T2: A customizable parallel database for multi-dimensional data. *ACM SIGMOD Record* 27, 1 (1998), 58–66.
[29] Lu Chen, Dejun Teng, Tian Zhu, Jun Kong, Bruce W Herr, Andreas Bueckle, Katy Börner, and Fusheng Wang. 2022. Real-time spatial registration for 3D human atlas. In *Proceedings of the 10th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*. 27–35.
[30] Lee A. D. Cooper, Alexis B. Carter, Alton B. Farris, Fusheng Wang, Jun Kong, David A. Gutman, Patrick Widener, Tony C. Pan, Sharath R. Cholleti, Ashish Sharma, Tahsin M. Kurc, Daniel J. Brat, and Joel H. Saltz. 2012. Digital Pathology: Data-Intensive Frontier in Medical Imaging. *Proc. IEEE* 100, 4 (2012), 991–1003. https://doi.org/10.1109/JPROC.2011.2182074
[31] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, Eric A. Brewer and Peter Chen (Eds.). USENIX Association, 137–150. http://www.usenix.org/events/osdi04/tech/dean.html
[32] Hongyi Duanmu, Fusheng Wang, George Teodoro, and Jun Kong. 2021. Foveal blur-boosted segmentation of nuclei in histopathology images with shape prior knowledge and probability map constraints. *Bioinformatics* 37, 21 (06 2021), 3905–3913. https://doi.org/10.1093/bioinformatics/btab418 arXiv:https://academic.oup.com/bioinformatics/article-pdf/37/21/3905/57196073/btab418.pdf
[33] Ahmed Eldawy and Mohamed F Mokbel. 2015. Spatialhadoop: A mapreduce framework for spatial data. In *2015 IEEE 31st international conference on Data Engineering*. IEEE, 1352–1363.
[34] Sofoklis Floratos, Mengbai Xiao, Hao Wang, Chengxin Guo, Yuan Yuan, Rubao Lee, and Xiaodong Zhang. 2021. NestGPU: Nested query processing on GPU. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1008–1019.
[35] OSM Foundation. 2024. OpenStreetMap. http://www.openstreetmap.org.
[36] Liang Geng, Rubao Lee, and Xiaodong Zhang. 2024. RayJoin: Fast and Precise Spatial Join. In *Proceedings of the 38th ACM International Conference on Supercomputing* (Kyoto, Japan) *(ICS '24)*. Association for Computing Machinery, New York, NY, USA, 124–136. https://doi.org/10.1145/3650200.3656610
[37] William Gropp, Ewing Lusk, and Anthony Skjellum. 1999. *Using MPI: portable parallel programming with the message-passing interface*. Vol. 1. MIT press.
[38] Yongqiang He, Rubao Lee, Yin Huai, Zheng Shao, Namit Jain, Xiaodong Zhang, and Zhiwei Xu. 2011. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 1199–1208.
[39] Yin Huai, Ashutosh Chauhan, Alan Gates, Gunther Hagleitner, Eric N Hanson, Owen O'Malley, Jitendra Pandey, Yuan Yuan, Rubao Lee, and Xiaodong Zhang. 2014. Major technical advancements in apache hive. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1235–1246.
[40] Yin Huai, Rubao Lee, Simon Zhang, Cathy H. Xia, and Xiaodong Zhang. 2011. DOT: a matrix model for analyzing, optimizing and deploying software for big data analytics in distributed systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (Cascais, Portugal) *(SOCC '11)*. Association for Computing Machinery, New York, NY, USA, Article 4, 14 pages. https://doi.org/10.1145/2038916.2038920
[41] Yin Huai, Siyuan Ma, Rubao Lee, Owen O'Malley, and Xiaodong Zhang. 2013. Understanding insights into the basic structure and essential issues of table placement methods in clusters. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1750–1761.
[42] Ibrahim Kamel and Christos Faloutsos. 1994. Hilbert r-tree: An improved rtree using fractals. In *VLDB*, Vol. 94. Citeseer, 500–509.
[43] Tahsin Kurc, Umit Catalyurek, Chialin Chang, Alan Sussman, and Joel Saltz. 2001. Visualization of large data sets with the active data repository. *IEEE Computer Graphics and Applications* 21, 4 (2001), 24–33.
[44] Tahsin Kurc, Chialin Chang, Renato Ferreira, Alan Sussman, and Joel Saltz. 1999. Querying very large multi-dimensional datasets in ADR. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*. 12–es.
[45] Rubao Lee, Tian Luo, Yin Huai, Fusheng Wang, Yongqiang He, and Xiaodong Zhang. 2011. Ysmart: Yet another sql-to-mapreduce translator. In *2011 31st International Conference on Distributed Computing Systems*. IEEE, 25–36.
[46] Rubao Lee, Minghong Zhou, Chi Li, Shenggang Hu, Jianping Teng, Dongyang Li, and Xiaodong Zhang. 2021. The art of balance: a RateupDB™ experience of building a CPU/GPU hybrid database product. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2999–3013.
[47] Mengbai Xiao Dongxiao Yu Rubao Lee Li, Xin and Xiaodong Zhang. 2024. Ultra-Precise: A GPU-based Framework for Arbitrary-Precision Arithmetic in Database Systems. In *Proceedings of the 40th IEEE International Conference on Data Engineering (ICDE 2024)*.
[48] Yanhui Liang, Hoang Vo, Jun Kong, and Fusheng Wang. 2017. ISPEED: an efficient in-memory based spatial query system for large-scale 3D data with complex structures. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.
[49] Nvidia. 2018. Nvidia Turing GPU Architecture. (2018).
[50] Nvidia. 2021. Nvidia Ampere GA102 GPU Architecture. (2021).
[51] Nvidia. 2022. Nvidia Ada GPU Architecture. (2022).
[52] Dave Patterson. 2003. A conversation with Jim Gray. *ACM Queue* 1, 4 (2003), 53–56.
[53] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J Abadi, David J DeWitt, Samuel Madden, and Michael Stonebraker. 2009. A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 165–178.
[54] Mousumi Roy, Fusheng Wang, Hoang Vo, Dejun Teng, George Teodoro, Alton B. Farris, Eduardo Castillo-Leon, Miriam B. Vos, and Jun Kong. 2020. Deep-learning-based accurate hepatic steatosis quantification for histological assessment of liver biopsies. *Laboratory Investigation* 100, 10 (01 Oct 2020), 1367–1383. https://doi.org/10.1038/s41374-020-0463-y
[55] Michael Stonebraker, Daniel Abadi, David J DeWitt, Sam Madden, Erik Paulson, Andrew Pavlo, and Alexander Rasin. 2010. MapReduce and parallel DBMSs: friends or foes? *Commun. ACM* 53, 1 (2010), 64–71.
[56] Dejun Teng, Furqan Baig, Zhaohui Peng, Jun Kong, and Fusheng Wang. 2023. Efficient spatial queries over complex polygons with hybrid representations. *GeoInformatica* (2023), 1–39.
[57] Dejun Teng, Furqan Baig, Qiheng Sun, Jun Kong, and Fusheng Wang. 2021. IDEAL: a vector-raster hybrid model for efficient spatial queries over complex polygons. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 99–108.

[58] D Teng, Y Liang, F Baig, J Kong, V Hoang, and F Wang. 2022. 3DPro: Querying Complex Three-Dimensional Data with Progressive Compression and Refinement.. In *Advances in Database Technology: proceedings. International Conference on Extending Database Technology*, Vol. 25. 104–117.

[59] Dejun Teng, Yanhui Liang, Hoang Vo, Jun Kong, and Fusheng Wang. 2022. Efficient 3D spatial queries for complex objects. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 8, 2 (2022), 1–26.

[60] Hoang Vo, Ablimit Aji, and Fusheng Wang. 2014. SATO: a spatial data partitioning framework for scalable query processing. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Dallas, Texas) *(SIGSPATIAL '14)*. Association for Computing Machinery, New York, NY, USA, 545–548. https://doi.org/10.1145/2666310.2666365

[61] Hoang Vo, Ablimit Aji, and Fusheng Wang. 2014. SATO: a spatial data partitioning framework for scalable query processing. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*. 545–548.

[62] Hoang Vo, Yanhui Liang, Jun Kong, and Fusheng Wang. 2018. iSPEED: a Scalable and Distributed In-Memory Based Spatial Query System for Large and Structurally Complex 3D Data. *Proceedings of the VLDB Endowment* 11, 12 (2018).

[63] Fusheng Wang, Jun Kong, Lee Cooper, Tony Pan, Tahsin Kurc, Wenjin Chen, Ashish Sharma, Cristobal Niedermayr, Tae W Oh, Daniel Brat, Alton B Farris, David J Foran, and Joel Saltz. 2011. A data model and database for high-resolution pathology analytical image informatics. *J. Pathol. Inform.* 2, 1 (July 2011), 32.

[64] Fusheng Wang, Jun Kong, Jingjing Gao, Lee A D Cooper, Tahsin Kurc, Zhengwen Zhou, David Adler, Cristobal Vergara-Niedermayr, Bryan Katigbak, Daniel J Brat, and Joel H Saltz. 2013. A high-performance spatial database based approach for pathology imaging algorithm evaluation. *J. Pathol. Inform.* 4, 1 (March 2013), 5.

[65] Kaibo Wang, Yin Huai, Rubao Lee, Fusheng Wang, Xiaodong Zhang, and Joel H Saltz. 2012. Accelerating pathology image data cross-comparison on cpu-gpu hybrid systems. In *Proceedings of the VLDB endowment international conference on very large data bases*, Vol. 5. NIH Public Access, 1543.

[66] Kaibo Wang, Kai Zhang, Yuan Yuan, Siyuan Ma, Rubao Lee, Xiaoning Ding, and Xiaodong Zhang. 2014. Concurrent analytical query processing with GPUs. *Proceedings of the VLDB Endowment* 7, 11 (2014), 1011–1022.

[67] Randall T Whitman, Bryan G Marsh, Michael B Park, and Erik G Hoel. 2019. Distributed spatial and spatio-temporal join on apache spark. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 5, 1 (2019), 1–28.

[68] Mengbai Xiao, Hao Wang, Liang Geng, Rubao Lee, and Xiaodong Zhang. 2019. Catfish: Adaptive RDMA-enabled R-Tree for low latency and high throughput. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 164–175.

[69] Mengbai Xiao, Hao Wang, Liang Geng, Rubao Lee, and Xiaodong Zhang. 2022. An RDMA-enabled In-memory Computing Platform for R-tree on Clusters. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 8, 2 (2022), 1–26.

[70] Simin You, Jianting Zhang, and Le Gruenwald. 2015. Large-scale spatial join query processing in cloud. In *2015 31st IEEE international conference on data engineering workshops*. IEEE, 34–41.

[71] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. 2015. Geospark: A cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*. 1–4.

[72] Yuan Yuan, Rubao Lee, and Xiaodong Zhang. 2013. The Yin and Yang of processing data warehousing queries on GPU devices. *Proceedings of the VLDB Endowment* 6, 10 (2013), 817–828.

[73] Kai Zhang, Kaibo Wang, Yuan Yuan, Lei Guo, Rubao Lee, and Xiaodong Zhang. 2015. Mega-kv: A case for gpus to maximize the throughput of in-memory key-value stores. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1226–1237.