Scalable Harmonization for Efficient Exploration of Heterogeneous Spatiotemporal Datasets

Tyson O'Leary, Everett Lewark, Federico Larrieu, Nathan Orwick, Paige Hansen,
Sangmi Lee Pallickara, and Shrideep Pallickara
Department of Computer Science, Colorado State University, Fort Collins, CO, USA
Email: {tmoleary, elewark, flarrieu, nathan.orwick, p.hansen, sangmi, shrideep}@colostate.edu

***Abstract*—Spatiotemporal data volumes have increased alongside a proliferation in their diversity. As a precursor to analysis, data must be reconciled and *harmonized*. Differences in encoding formats, spatial and coordinate reference systems, data types (points, shapes, rasters, grids), and storage strategies disrupt analytic workflows. The challenge grows more complex when analyses require "layered" datasets, in which multiple heterogeneous sources must be integrated to yield consistent insight. Here, we present a methodology that systematically harmonizes such datasets and enables their layering into federated collections. Our approach provides an ecosystem of interoperable services that perform dimensionality reduction, query evaluation, correlation analysis, normalization, and visualization. These services can be chained together, operate on distributed datasets, and integrate directly with existing large-scale computation frameworks such as Spark and MapReduce.**

***Index Terms*—big data, harmonization, wrangling, visual analytics, federation, metadata**

I. Introduction

Underpinned by improvements in the quality and capacity of networks, disks, and sensing equipment, there has been an exponential growth in the amount of data that are generated every year. This growth in data volumes has occurred alongside an increase in the sheer variety of the data. Together, this increase in data volumes and their accompanying diversity means that data wrangling increasingly dominates analysis times [1]. Consequently, large amounts of the data are siloed and not in use – a phenomenon referred to as "dark data" [2].

The datasets we consider are spatiotemporal. In spatiotemporal datasets data are geocoded and timestamped to identify where and when the individual data item was recorded. Spatial datasets account for the vast majority of the datasets that are generated annually. These include satellite imagery, social media data, ecological, and environmental monitoring data. Spatiotemporal datasets continue to be made available in domains encapsulating terrestrial, agricultural, atmospheric, ecological, and environmental processes.

Increases in the volumes and proliferation of spatiotemporal datasets have occurred alongside an increase in their diversity. The data are generated using different encoding formats, spatial referencing and coordinate systems, etc. Another challenge that arises is the units in which measurements are recorded. For example, the International System of Units (SI) and customary units are used extensively by the scientific community. An exemplifying situation occurs with monitoring systems that rely on sensing equipment. These may report measurements using diverse unit systems, such as temperature reported in Kelvin, Celsius, or Fahrenheit.

Consider a researcher who would like to analyze how weather patterns have contributed to increased concentrations of nutrients in lakes over a particular spatial extent. This would involve combining three data sources; meteorological variables that are available in gridMET (gridded time-series datasets) [3], water-quality variables from time-series data published by the EPA (a time-series point set), and vector shapes for lakes offered by the USGS national hydrology database (defined by shapefiles) [4]. Furthermore, these datasets have measurements in different units (SI in the case of gridMET and customary units in the case of the EPA) and different coordinate and spatial projection systems. The encoding formats in these cases are also diverse, including netCDF, CSV, and XML. Processing without reconciling data diversity can produce results that are incorrect and inconsistent.

In large-scale distributed systems, data arrives from many different sources each with its own formats, coordinate systems, and units that precludes meaningful analysis. Consider, for example, a researcher faced with several multivariate datasets: without harmonization, their visualizations distort, their measurements misalign, and the results become unreliable. This problem is exacerbated in large-scale distributed systems where the sheer volume of data is compounded by its diversity—not only in the data itself but also in the variety of storage frameworks employed to house it, an approach known as polyglot persistence [5], [6].

The crux of this study is to support fast, expressive explorations over spatiotemporal datasets via harmonization at scale. At the heart of our approach lies the *scaffold*, a software abstraction that unifies heterogeneous spatiotemporal data at scale. By reconciling disparate coordinate references, spatial projections, unit systems, and data types, the scaffold creates a single, coherent view of the underlying information. Building on this foundation, we design a suite of services to work seamlessly atop the scaffold. By simplifying and enhancing analysis, we posit that users can extract accurate insights from clean, consistent information encapsulated within scaffolds.

This research was supported by the National Science Foundation (1931363, 2312319), the National Institute of Food Agriculture (COL014021223), an NSF/NIFA Artificial Intelligence Institutes AI-CLIMATE Award [2023-03616], and the Clare Boothe Luce Professorship.

A. Challenges

There are several challenges in ensuring effective data reconciliation and harmonization at scale.

- Researchers are often interested in performing analyses that cross-cut different types of data: points, shape files, and raster/gridded datasets.
- The data are encoded in a diversity of encoding formats, measurement units, and coordinate referencing systems.
- Data are voluminous, spanning vast spatial (country, continental, and global) and temporal (years, decades) scales at high frequencies and resolutions.

B. Research Questions

The overarching objective of this study is to support effective explorations at scale over diverse datasets. Specific research questions that we explore include:

RQ-1: *How can we reconcile data diversity?* In addition to data being encoded in different formats, the data may be represented as points, vector shapes, and rasters/grids.

RQ-2: *How can we support effective analyses and a rich ecosystem of services over diverse datasets?* Such services could include dimensionality reduction and visualizations.

RQ-3: *How can we allow users to efficiently explore and identify spatiotemporal scopes that are relevant to their analyses? And how can we ensure that such exploration interoperates with frameworks that users already employ?*

C. Approach Summary

Our methodology facilitates interactive and batch analytics at scale over diverse spatiotemporal data. This includes (1) creation of a software-powered data abstraction that is amenable to layering, querying, and transformation, and (2) a suite of services built around this abstraction that simplify data analysis tasks.

We introduce a novel software-powered data abstraction, the *scaffold*. Each scaffold includes metadata and a suite of transformations that allow it to be layered with other scaffolds. The metadata includes various information about the dataset including spatial extents for which data are available, the temporal bounds associated with the data, the data type, per-variable precomputed summary statistics, the unit system associated with the recorded observations, and the extracted spatial and coordinate referencing system for the dataset.

The scaffold abstraction is space-efficient and computationally light. The scaffold will compute summary statistics for the data in an online fashion, allowing efficient normalization of the data. The scaffold maintains pointers to the relevant portions of the relevant dataspace and is highly amenable to serialization for transfers. Scaffolds allow data to be transformed and wrangled, staging for expressive and efficient analyses. Each scaffold supports transformation of the spatial and coordinate reference system to any of the over 7,000 different coordinate reference systems in the EPSG database [7]. Similarly, the observations can be transformed into one of 11 supported unit systems. Transformations on scaffolds are launched with data locality to ensure fast completion times.

Scaffolds can be layered. Our methodology performs reconciliation across scaffolds. A scaffold that is layered over an existing base scaffold automatically inherits the spatial and coordinate referencing system, the units of measurement, and temporal representations from the base scaffold.

We also design a suite of services and capabilities around the scaffold abstraction. This includes operations such as masking, query evaluations, dimensionality reduction, correlation analysis, and visualization. Because individual services are designed around scaffolds, they can be designed independently and daisy-chained into more complex services. We also include support for expressive queries backed by a heuristic-based query plan optimization in order to support user explorations.

Because the scaffold ensures that every data item shares a common frame of reference, researchers can explore large, complex datasets quickly and with confidence. This allows researchers to gain insights previously obscured by the raw, unharmonized data. Our benchmarks demonstrate that this harmonization accelerates exploratory analysis with complex operations performed efficiently and at high throughput. By reducing the computational overheads associated with reconciliations and alignment, scaffolds free up resources for deeper analytical tasks.

Our methodology also supports interoperation with computational frameworks such as Apache Spark; scaffolds are agnostic to the specifics of any given engine. Our methodology does not preclude extensions to other or future computational frameworks. As a result, researchers can work directly with scaffolds (and the harmonized datasets they encapsulate) within computational frameworks they already know and trust.

D. Paper Contributions

Scaffolds provide a unifying abstraction that reconciles the persistent discrepancies of encoding formats, coordinate reference systems, and units of measurement; barriers that frustrate reuse and analysis. Their compact, memory-efficient structure allows them to act as surrogates, bridging across datastores and computational libraries without adding burdens for the user. We demonstrate this in practice through native interoperation with Spark, Sedona, and GeoPandas. By design, scaffolds do not replace a user's chosen computational framework; they fit within it, extending its reach and capabilities. This abstraction enables queries that cut across disparate datasets and produces results that can be materialized directly within familiar environments. The result is a coherence of design that makes possible new forms of interactive and scalable analytics. In harmonizing the underlying complexity while preserving the user's ability to work within established tools, scaffolds allow researchers to concentrate on discovery rather than reconciliation.

- Scaffolds can represent very voluminous datasets.
- Scaffolds allow targeted operations over the data, allowing researchers to focus their analysis over a particular spatial extent and temporal bounds without triggering extensive data accesses or movements.

- Services are built around the scaffold abstraction, giving each new dataset immediate access to existing capabilities. Because some services can themselves return scaffolds, they may be chained together to create progressively richer functionality.
- Finally, our methodology does not place restrictions on the data encoding formats and the data storage frameworks that are used to represent and store the data.

E. Paper Organization

In section II we discuss related work, section III describes our methodology. We describe our experimental setups and empirical benchmarks in section IV. Finally, section V outlines our conclusions and future work.

II. Related Work

The scaffold abstraction targets geospatial data with the goal of simplifying data processing. Apache Spark uses a related concept of a resilient distributed dataset (RDD) to provide a high-level abstraction of a distributed data model, and enables transformations and actions to perform distributed computational operations [8]. Spark has been highly influential as a basis for multiple other frameworks. For example, the GeoTrellis framework applies Spark for distributed processing of geospatial raster datasets in particular, as demonstrated by Kothari et al. for computing properties including Normalized Difference Vegetation Index (NDVI) [9]. The Apache Sedona framework, formerly known as GeoSpark [10], extends Spark with geospatial data structures including R-trees, Quad-trees, and KDB-trees to decrease processing time [11]. Similar to Sedona, Geopandas [12] provides geospatial data processing for Pandas in Python (although not distributed).

Prior works in raster-vector data processing focus on specific optimizations in storing or joining data. Silva-Coira et al. propose a framework [13] for jointly querying raster and vector datasets, leveraging the k2-raster compact data structure for efficient storage and querying in compressed form. This is unlike our approach, which emphasizes harmonizing diverse datasets with scaffolds, a metadata structure. Their work primarily addresses efficiency in spatial queries without integrating multi-layered data transformations or raster-vector masking. Villarroya et al. present a distributed system [14] for efficient spatial joins between large-scale vector and raster data, termed MapalSpark, and found that they were able to achieve better performance than GeoSpark, SpatialSpark, and LocationSpark in this task. Our framework focuses on modular scaffolding for extensible spatial analysis workflows on raster and vector data across disparate distributed data stores. Additionally, our implementation leverages GDAL [15] to perform raster-vector masking.

Other user-facing applications exist for performing similar visual analytics. Two such GUI-based systems, JMP [16] and Spotfire [17], host a variety of statistical services that run directly on a user's machine. Both include support for spatiotemporal data analytics; however, their capabilities are not as strongly tied to this type of data and therefore cannot accomplish the task as seamlessly as our system. They also require a higher level of expertise, as a goal of the scaffolds ecosystem is to hide computation details from the user to hold the final product as the focus. JMP and Spotfire are primarily batch processing tools used for one-time data analytics. This is a valid use case for scaffolds, but we also support interactive querying ready for a production environment including high integration with state-of-the-art database systems and query optimization.

Nanocubes [18] and HashedCubes [19] are frameworks that aim to store large-scale data cubes with a low memory footprint, and support real-time interaction. Similar to our methodology, they focus on efficient geospatial data processing. Scaffolds do not inherently grant the benefits gained from these frameworks; however, there is nothing that precludes the use of these frameworks alongside scaffolds. Our system is adaptable and can easily integrate with other geospatial frameworks.

Managing data and metadata [20] for large scientific collections remains a persistent challenge; one that is addressed across diverse computational landscapes, including web services, large-scale peer-to-peer grids [21]–[23], and collaborative environments [24], [25]. Efforts to refine spatiotemporal data handling have produced systems such as Galileo [26], Trident [27], and Atlas [28], [29], which do far more than simply store information: they enable sophisticated querying, whether ad hoc [30], geometry-constrained [31], or highly expressive [32]. To interpret these immense datasets, researchers have developed algorithms that transform data [33] into data sketches—compact, approximate representations that balance efficiency with accuracy. Examples include Synopsis [34], Pebbles [35], and Gossamer [36]. SCAFFOLD is complementary, and nothing in our methodology precludes interoperation with such systems as data sources.

Butenuth et al. [37] discuss the application of federated databases for raster and vector geospatial data, and systems for cataloging data relationships and performing operations such as raster-vector masking. Their effort was largely focused on the algorithms and processes required to combine heterogeneous data, which we build upon in our work, with a data abstraction.

III. Methodology

Here, we describe our scaffold abstraction, outline the capabilities and surrounding ecosystem of services designed to support it, and detail the refinements implemented to ensure the effective evaluation of queries.

A. The Scaffold [RQ-1]

We propose representing datasets with scaffolds to *harmonize* diverse data for inter-dataset analyses. We address harmonization by reconciling data properties such as coordinate reference systems (CRSs) and unit systems, aligning spatial and temporal dimensions, and supporting data store federation.

To maximize generalizability, our methodology models a dataset as a collection of records with spatial location and

timestamps. Spatiotemporal data is often represented as gridded rasters, such as gridMET (meteorological data covering the US) [3]. To support interoperation, our methodology partitions the data into tiles, which are treated as records.

A *scaffold* is an abstraction over the dataset that encapsulates extensive metadata including structure of the data, storage location, statistics, labels and descriptions, and other necessities for performing queries, aggregations, and transformations. In environments with high data diversity, scaffold metadata underpins identifying differences between datasets and informs reconciliation strategies. Crucially, creation of a scaffold from a dataset enables interoperation with other datasets.

Scaffolds can be built from any spatiotemporal dataset, whether the data is purely spatial, purely temporal, or both. For example, individual demographic data points from the United States Census Bureau are uniquely identified by spatial FIPS codes, which can be associated with counties or other subdivisions. While the census is recorded every ten years, a single census is regularly considered a standalone dataset and would thus be temporally singular.

1) Constructing Scaffolds: Scaffold construction is fully automated, with supported data stores, but allows users to configure and guide the process as needed. Scaffolds are largely inferred alongside data ingestion, acquiring information like identifiers, variable names, and data types directly from the source data. Structured data, such as from a SQL database, are entirely inferred, while NoSQL data sources, such as MongoDB, offer less inference capability but still provide enough information to populate basic properties of a scaffold.

The base unit system for a scaffold is determined by the most frequently used units among the underlying variables; ties are broken in favor of SI and U.S. customary units in that order. Each variable in the dataset carries its own measurement unit within its scaffold. When units are not packaged with the data, scaffolds will be unable to infer them; an example where a domain expert may need to guide the scaffold construction.

A scaffold is constructed with harmonization for downstream tasks, such as analysis and visualization, in mind. Construction is oriented around collecting necessary metadata for automatically handling the necessary conversions and adjustments, ensuring the data end up in the expected form.

2) Transforming Scaffolds: Scaffold transformations are modular and amenable to chaining, as shown in the pseudocode example given in Fig. 1. Because scaffold transformations take scaffolds as input and return scaffolds as output, they can be daisy-chained using fluent interfaces, ensuring that every new transformation can interoperate with existing transformations. The crux of scaffolds is their high level fluidity. Triggering operations over the full data is resource intensive, so scaffolds minimize repeated I/O operations and mutation redundancy by performing the changes first logically, allowing pipeline optimization in a lightweight setting. Upon evaluation, this lightweight "recipe" is used directly to request the data from source, perform each operation in the pipeline, and stream the final transformed values to the user.

```
# Retrieve prebuilt scaffolds
tmmn = load("gridmet_tmmn")
tmmx = load("gridmet_tmmx")
bodies = load("water_bodies")
temp = load("water_temperature")

# Transform
respite = tmmx.align(tmmn, SPATIOTEMPORAL)
              .subtract()

bodies.keyJoin(temp) # one-to-many
      .convertCRS("EPSG:3857")
      .boundary(NE_USA)
      .align(respite, SPATIAL)
      .mask()
      .rasterAverage("resp_avg")
      .correlate("wtemp", "resp_avg")
      .visualize(color="correlation")
```

Fig. 1. Scaffolds are loaded from persistent storage, modified with transformations (filters, mappings, and layering), then finalized with visualization. Transformations on the scaffolds occur through fluent interfaces making powerful operations easily accessible and adaptable.

For example, converting a dataset from the SI system to the imperial system immediately changes the new scaffold's metadata to reflect the new system– including converting the tracked summary statistics to imperial units. Direct unit conversions are enabled by the QUDT Ontology [38]. This ensures a just-in-time approach for transformations: no wasted computation, no unnecessary storage, and just the right operation when needed.

This design has two notable advantages. First, it abstracts away the details of execution. In particular, a user need not concern themselves with the orchestration details of whether a transformation runs sequentially on a local machine or as a distributed job, as long as the results are accurate and timely. This separation of concerns also allows optimizations across transformation operations be performed independently of each other. Second, delayed scaffold transformations keep memory consumption in check. In big data environments, loading an entire dataset into memory is often impractical. By delaying computations until they are needed and streaming results as they are processed, we minimize the memory footprint while leaving room for pipeline optimizations down the line.

3) Layering Scaffolds: A holistic view of contributing factors is crucial to insightful data analysis, which means bringing data domains together. However, these data are generally structured heterogeneously. Dataset differences such as storage framework, encoding, or format, often require specialized code to combine datasets. Subtler challenges arise, like mismatched unit systems, which must be manually reconciled to ensure valid comparisons. Scaffolds solve the problem of combining datasets, or layering, by encapsulating all information about a dataset needed for reconciliation and combination, and defining strategies for combining scaffolds that are logically equivalent to combining datasets.

We support multiple types of layering (extension, alignment, and join), each addressing ways datasets can be combined. Before dataset layering can occur, dataset attributes must be reconciled ensuring that key properties such as unit systems

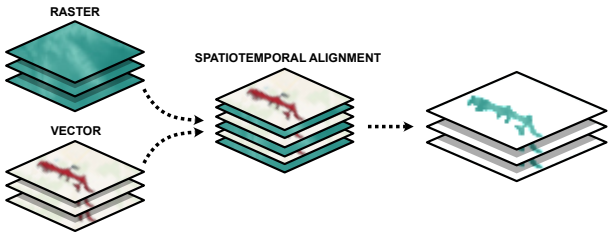


Fig. 2. Spatiotemporal alignment and masking of gridMET max temperature over Horsetooth Reservoir in Colorado. Tiles and polygons in the same location at the same point in time are associated.

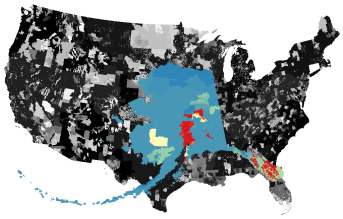


Fig. 3. Incorrect reprojection of map coordinates causes spatial misalignment. In this QGIS-based example, USDA SSURGO data using an Alaska-specific Albers projection is intentionally misinterpreted as continental US Albers.

and coordinate references systems (CRSs) match. In our context, this is called reconciliation: the process of negotiating the differences between scaffolds (datasets).

Layered scaffolds take their cue from the primary scaffold, the first added, ensuring consistency across all datasets in unit system, CRS, and spatiotemporal scope. Unit systems are a key aspect of reconciliation; without aligning them, calculations and comparisons lose their meaning. The system automatically applies a unit conversion transformation to all layered scaffolds, ensuring accuracy and coherence across datasets.

Reprojection is a fundamental process in geospatial data analysis and visualization, and is another important part of reconciliation. It ensures spatial datasets align correctly when integrating multiple sources. An example of an incorrect projection can be seen in Fig. 3. Different datasets, whether vector, point, or raster, often originate from varying coordinate systems, which can lead to distortions or misalignments when analyzed together. Reprojection involves transforming spatial data from one CRS to another, maintaining spatial accuracy.

Extension: A special case of layering data is extending a dataset to include a closely related dataset. Eligible datasets contain the same type of information but differ in attributes that require reconciliation. For example, our water quality dataset [39] includes water temperature values, which are stored in degrees Fahrenheit (°F) and Celsius (°C) indicated by a field in each record. We split this into single-unit datasets, which are then reconciled, converting the °F data to °C. The new scaffold logically describes the °C data extended with the reconciled output. Another example is illustrated with US Census data, which is published as individual datasets, but can logically be viewed as a timeseries. Scaffolds of these separate datasets are combined via extension to represent a single multi-decade dataset. The full scaffold can later be extended with new census data as available.

Spatiotemporal Alignment: Some common spatiotemporal operations require computation with two (or more) data points from different datasets together. For this analysis to be meaningful, those points must describe the same location in both space and time. The process of associating points, spatiotemporal alignment, supports downstream analysis across spatiotemporal datasets. A scaffold encodes information about where its data exists geographically and temporally. We use this information to determine the scope of the resulting combined scaffold and to reduce the search space when the layered data are requested. Spatiotemporal alignment is a necessary step towards joining data on spatial or temporal properties: a full join would include all points while an inner join would exclude points (points in space and time that do not have data from both datasets). Spatiotemporal alignment is depicted in Fig. 2.

Joining Information: Our methodology supports joining layers with procedures similar to traditional relational databases. The traditional join can only be supported for commonly keyed datasets, such as those with the same associated geometry identifier. For instance, joining two scaffolds that use county FIPS codes is straightforward. Joining one of these with a scaffold identified by 12-digit hydrologic unit codes is a more complex task. For this case, we provide a more advanced joining technique that associates data points with a custom equivalence relation, such as geographic intersection or nearness.

Performing a join of a scaffold representing records without direct geospatial components (record scaffold) and a raster scaffold is more difficult than joining a record scaffold with a vector scaffold. It still requires that both use identifiers from the same set of keys, but a raster image's identifiers exist in one of its bands. If the raster has no bands with identifier keys, then its dataset and therefore its scaffold can only be joined with some custom equivalence relation. In the case of rasters, a single record is a tile, which would contain thousands of pixels and therefore thousands of identifiers, although all may not be unique. The closest operation to a join in this case would be to add new raster bands for each joined variable, and assign values to the pixels based on their identifier.

Vector-to-raster joins imply a many-to-many relationship between records because raster tiles can overlap many vector shapes, and these shapes can span multiple tiles. Equivalence between a raster tile and vector shape is determined by whether they intersect spatially. The contents of the final stream of data can either be vector shapes or raster tiles. Since vector shapes are represented by a single measurement, raster measurements within the vector boundary must be aggregated. For raster tiles, the measurements from vector shapes are duplicated into pixels within the boundary. In both cases, we convert vectors to masks to "cut" the relevant pixels from the raster. This masking operation works in real time and allows associating values between raster tiles and vector shapes in a way that

simultaneously supports visualization and continued analysis.

B. Ecosystem of Services [RQ-2]

We have designed an ecosystem of services around our scaffold abstraction, each designed to work seamlessly within its framework. These capabilities include tailored visualizations for identifying spatial variation of data, a set of charting operations, kernel density estimations, and principal component analyses for dimensionality reduction. A key advantage of supporting these services with scaffolds stems from the easy inferencing of parameters based on scaffold attributes. By leveraging scaffolds, the services are also able to perform their computations with data across all supported formats, encodings, representations, units, and storage frameworks. The output of these services are also scaffolds, so it is important to note that these values can be visualized the same as any other.

1) Visualization: Data visualization is one of many final steps supported in the scaffold ecosystem. Our methodology includes built-in tools for rendering choropleth maps and various chart types, but its real strength lies in its flexibility: scaffolds streamline the process to prepare data according to visualization format standards.

We demonstrate scaffold interoperability with visualization frameworks using *deck.gl* [40] for rendering on a map and *recharts* [41] for charting. The only requirements are a scaffold describing the data to be visualized and an implementation for communicating with the scaffold ecosystem. Because of this simplistic design and standardized visualization formats, other existing visualization frameworks can be substituted directly.

Raster tiles, vector shapes, and points can all be rendered on a map. Commonly, map rendering frameworks render a raster dataset by sending one request per rendered tile image, parameterized by the tile's coordinate. We support this by easily splitting the relevant scaffold into sub-scaffolds representing tiles. In contrast, both vector shapes and points are streamed from the scaffold system directly to the map in the form of GeoJSON. We support simplification of vector shapes using the Ramer-Douglas-Peucker [42], [43] or Visvalingam [44] algorithm to reduce storage consumption and speed up rendering times. Results can seamlessly populate a chart, a table, or be rendered on the map.

2) Statistical Applications: Support for statistical computations is a common necessity in big data computing and management frameworks, and our system is no exception. By leveraging scaffolds, these statistical services can simply be applied in the same way as our transformations, i.e. through fluent interfaces, and they inherit scaffold benefits including automatic reconciliation, data source federation, and distributed processing. We walk through a selection of our services.

To explore and compare distributional characteristics, we support kernel density estimations (KDEs) including automatic bandwidth selection via Scott's Rule [45] or Silverman's Rule [46]. To support exploration of spatial variation in the correlation between datasets, we compute Pearson correlation coefficients within and across scaffolded datasets, space, and time. Layering scaffolds can quickly explode the feature space, so we support principal component analysis (PCA) for dimensionality reduction, including scree plots and component selection based on count or variance threshold. Finally, to support downstream machine learning, scaffolds can be automatically split into train, validation, and test scaffolds.

C. Query Evaluations [RQ-3]

A query consists of a scaffold and a pipeline. The scaffold defines the structure and properties of the data and encapsulates the transformation pipeline. The pipeline defines transformations on the input data, which could include element-wise mapping operations, filters, or joins. An action can be included to define finalizing operations applied to the resulting collection of data. Potential actions include rendering raster tiles, limiting to a single result, or calculating summary statistics.

The query system is designed around supporting declarative queries. The goal is to keep things simple for the user: a user need only specify the data of interest and the form of the results. The scaffold captures the relevant data while the action defines the output format. Everything else (e.g., *where* the data comes from, *how* it is combined and reconciled, and *when* reconciliation happens) remains behind the scenes, allowing users to focus on their analysis instead of the mechanics.

1) Data Federation: Data federation allows data to be placed into a multitude of storage frameworks based on the specialties of the data stores. For example, MongoDB supports spatial indexing which can speed up queries targeting specific geographic regions. The scaffold ecosystem is neutral to underlying storage frameworks, so strengths of data stores can be utilized for the datasets that suit them best, while maintaining all transformation and finalizing capabilities.

2) Query Optimization: Users should not need expertise in pipeline optimization to get timely results, especially considering the variety of storage frameworks our methodology accommodates. Our methodology incorporates a query optimizer, which embeds the necessary knowledge into heuristics, ensuring queries run efficiently without requiring users to fine-tune them manually. The optimization process is comprised of (1) reorganizing filters, (2) combining steps, (3) removing unnecessary transformations, and (4) pushing compatible transformations to the direct data source query.

Part one of the optimization process, *reorganizing filters*, aims to eliminate redundant operations during query evaluation by filtering out as many records as possible before any transformations are applied. Transforming a record only to discard it later is wasteful: it is better to filter it out upfront and skip the unnecessary processing. For example, if a pipeline has a CRS transform followed by a spatial bounding box filter, the bounding box will reduce the amount of data needing transformation, and should thus be done as early as possible. However, to be valid, the bounding box must be transformed from the destination CRS to the source CRS.

The optimizer attempts to move filters as close to the beginning of the pipeline as possible, which can involve moving past map (independent operations on each item) or join steps along the way. A filter is easily moved past the map step if the filter does not use any fields affected by the map step. If it depend on affected fields, the values in the filter can be *reversed* by the map step. Reversing the values involves performing the *inverse* action defined in the map step on the values in the filter in order to make it valid for the state of the data before the map step is applied to it. In the context of the CRS bounding box example above, this would mean performing the inverse CRS reprojection on the coordinates of the bounding box, thereby defining the bounding box in the original CRS.

Part two, *combine steps*, reduces logically equivalent sequences of pipeline steps into single steps. The number of operations on the data is reduced by recognizing and merging transformations. In a given set of conversions, it is not uncommon for steps to cancel each other out. In a very simple example, if the pipeline included steps for adding 3, then subtracting 3, the optimizer combines these steps to adding 0. Further, this combination is readily recognized as a no-op. Even if the combination does not result in a no-op, the operations over the data are still reduced. In a more applicable example, if one step converts a WGS84 CRS to web mercator, then the following step converts back to WGS84, effort is wasted. This is especially true for the expensive CRS conversion operations, but if the steps are combined it can be recognized as a no-op and later removed altogether. Consider there could be any number of other steps in between the two conversions. A user could have chosen to convert the coordinates to allow the use of a web mercator bounding box, then converted back to WGS84 because it is their desired result format.

Part three, *removing unnecessary operations*, is a crucial step. Besides the case discussed previously when steps are combined and recognized as a no-op, users may reuse a scaffold they have created to align with their analyses. Imagine a user starts with a scaffold representing vector geometry, applies a CRS reprojection, adds several more transformations, and streams the results to be visualized on a map. Later, they use that same scaffold to compute summary statistics. Since summary statistics do not depend on the CRS, and reprojection can be costly, the optimizer recognizes this and removes the unnecessary transformation, saving time and resources.

Finally, in part four, *pushing computations*, operations required for transformations are pushed to the underlying data source query. Here, the optimizer makes use of database-specific drivers to convert the transformations in the pipeline into database queries. For example, when filter steps are done in the database, existing indexes can be used and only a subset of data be pulled out. Each data source has its own strengths and weaknesses, and each support different feature sets, so it is important to allow for handling them differently. By encoding pipeline computations into data source queries, we take advantage of data locality and specialized indexing.

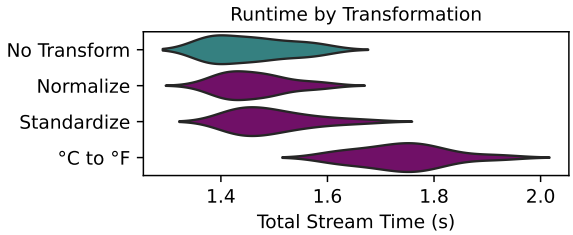


Fig. 4. There is a marginal difference in time between each transformation and the raw data stream, showing that the time is more affected by I/O and transformations add low overhead.

3) Distributed Processing: Spatial data are large and complex, requiring significant computational power for transformations such as CRS reprojections. By distributing the workload across multiple nodes, we can significantly reduce processing time. The scaffold system itself is inherently distributable as the scaffold can be partitioned into n smaller scaffolds, each being processed on a different node. This supports the implementation of a micro-service with the key advantage that scaffolds serve as the primary method of communication. This uniformity allows servers to coordinate efficiently without additional complexity. Our implementation shows scaffolds are amenable to distributed work and are able to be integrated with existing distributed frameworks, such as Apache Spark.

IV. Benchmarks & Performance Evaluations

At its core the scaffold is designed to reconcile diverse datasets into cohesive datasets while supporting interoperation with computing environments. We assess our methodology by profiling several dimensions of the scaffold's capabilities – measuring computational overheads, latencies, and throughput. Because we support a mix of interactive and batch processing tasks, our benchmarks are not aimed at matching isolated peak-performance measures, but rather to illustrate the effectiveness of scaffolds for a diverse array of computing tasks. This ensures that scaffolds can underpin *both* real-time applications and batched analytical pipelines alike; capabilities that are critical for several data-intensive computing workloads.

We orient our benchmarks around use cases of scaffolds in the analysis of water bodies using three datasets: the USGS National Hydrography Dataset (NH) for water body shapes (vector data), EPA's Water Quality Dataset (WQ) for measurements (point data), and GridMET (GM) daily meteorological data (raster data). NH and WQ share a CRS of ESPG:4326 (CRS-1) in MongoDB; GM uses EPSG:3857 (CRS-2) in Cassandra.

Benchmarks were run on a cluster of 26 HP DL60 g9 servers (64 GB of RAM, eight 2.1GHz CPU cores) running AlmaLinux 8.10 and OpenJDK 11.0.25. The java-based system uses *GDAL* [15] and *GeoTools* [47] for spatial data operations.

A. *Transformations on One Scaffold* [RQ-1]

We explore transformation overhead on streamed data by applying normalization, standardization, and unit conversion to 100,000 WQ water temperature measurements. We stream the

TABLE I
Performance of CRS transformations on vector, point, and raster datasets, highlighting high throughput and low latency across all dataset types. CRS reprojection is efficiently executed, satisfying performance for high data volumes in a batch processing context.

	Point		Vector		Raster	
	μ	σ	μ	σ	μ	σ
Throughput (records/s)	1,999	89	56	2	77	4
Latency (ms)	0.50	0.02	17.87	0.81	13.03	0.62
Records	991,780		37,110		21,228	
Conversions	991,780		30,842,780		1,332,624,666	

unaltered scaffold-defined dataset as baseline. Results in Fig. 4 demonstrate retained interactivity with low transformation overhead.

B. *CRS Reconciliation* [RQ-1]

To evaluate the cost of CRS reprojections, we reproject each of our datasets to a different CRS. Our three datasets represent three feature classes: point, vector, and raster. We project the data stored as CRS1 to CRS2 and vice versa. Highlighted in Table I, because scaffolds support working in the data's native format, vector and raster benefit from parallelism.

C. *Distributed CRS Reconciliation* [RQ-3]

We evaluate the effectiveness of distributed CRS reprojection of over 37,000 vector water bodies, for a total of 30,842,780 coordinate transformations. To support distribution, the scaffold representing the entire dataset is split into n scaffolds representing $1/n$ of the work based on spatial identifiers (e.g., `GridCode`). Figure 5 illustrates the linear speedup as workers are added. Data volume is unchanged, so disk retrieval I/O remains consistent, with some added overhead to receive results from workers. This approach enables efficient parallel execution while preserving the metadata and transformation capabilities inherent to each scaffold, allowing seamless recombination of results after processing.

The workload orchestration described here was developed independent of a cluster manager [48], [49]. This design decision (intentional though constrained) has direct implications for the effectiveness of our workload placement strategy. Our orchestration scheme lacks access to essential cluster state, and therefore does not achieve the same level of optimization as those made with full context. This limitation stands in contrast to our scaffold integration with Apache Spark, discussed in the following section, which more fully leverages cluster context to improve performance.

D. *Interoperation with frameworks and libraries* [RQ-3]

A central contribution of scaffolds lies in their ability to interoperate with diverse computational frameworks, even when those frameworks differ in language, design, or data model. The same abstraction that reconciles formats and units

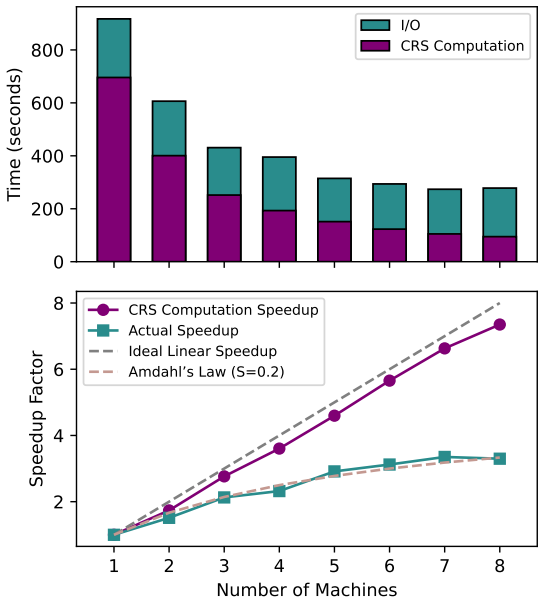


Fig. 5. Distributed processing of CRS transformations on vector data improves with more resources, enhancing interactivity.

also enables data to flow naturally between libraries such as GeoPandas and JVM-based systems such as Sedona and Spark.

We allow materialization of scaffolds as a Sedona dataset (vector or raster) for efficient distributed processing or a GeoPandas dataframe for familiar Python operations, before re-entering the scaffold environment. At each stage, scaffolds preserve their own advantages: automatic reconciliation, federation across sources, and access to the broader ecosystem of services. This design allows users to perform processing on scaffolds in the language of their choice, whether Python, Java, or Scala. The result is not a replacement of existing tools, but a framework in which they coexist. Scaffolds make transitions between libraries seamless giving users the full power of interoperability while keeping them grounded in the computational environments they already know and trust.

Because frameworks such as Spark, Sedona, and GeoPandas are engineered for computational efficiency within their own domains, the operations they perform there often surpass what we can achieve in our native implementations. However, their strength depends upon the foundation we provide. The central contribution of scaffolds lies in memory-efficient, harmonized representations of disparate data. Our goal is not to compete with these frameworks but to enable them, through memory-efficient and harmonized representations of disparate data. Without this reconciliation of formats, units, and coordinate systems, their analytic strengths would remain unusable. Scaffolds make their efficiencies usable by ensuring that the data they consume is coherent, compatible, and ready for computation.

We demonstrate the interoperability in Table II by implementing the same task as described in §IV-C within the Apache Spark distributed computing framework using 8 machines and

TABLE II
CRS transformations using the scaffold ecosystem compared to the addition of state-of-the-art tools. Scaffolds interoperability is powerful due to combining optimized frameworks with seamless harmonization of data.

Tool	Data Type	Native scaffold	Tool with scaffold
Spark	Vector	300s	160s
Sedona	Raster	30s	19s
GeoPandas	Vector	660s	500s

in GeoPandas on a single thread. We also implemented a CRS conversion of 108 raster tiles using Apache Sedona on 8 machines. All three cases performed better than scaffolds' native implementation and were made simpler using scaffolds than the same operations in the respective tools alone.

E. Query Optimization [RQ-3]

To profile our optimization, we formed a pipeline following a realistic use case: calculating the mean of measurements in a geographically bounded region. The scenario is the following:

Imagine a user wants to find the mean water temperature of bodies of water in the northeast United States. Illustrated in Fig. 6a, first, the user selects the water temperature dataset, measured in degrees Celsius. Consider a user based in the United States, who prefers to view temperatures in degrees Fahrenheit—so they add a unit transformation. Realizing the need to narrow their spatial extent, the user joins the dataset with a shape-based bodies of water dataset, linked by a common key. Since that dataset is in a different coordinate system, they add a reprojection step to convert it to `WGS84`, harmonizing it with their northeast United States boundary. And finally, they compute the mean of the temperature values.

The optimizer recognizes and resolves two inefficiencies in the user's pipeline. First, the optimizer moves the bounds filter as early as possible, reversing its CRS as it moves past the reprojection step. Second, the mean does not require a reprojection, so it is removed. With these changes, the filter is now handled by the data store, which holds the water bodies and has a spatial intersection stage. This is faster, and means the search space is reduced as early as possible. Removing the CRS reprojection step also provides speedup as it is a costly operation. The heuristically improved pipeline is shown in Fig. 6b, which ran in 30 seconds compared to 477 seconds for the unoptimized query. We achieve a **16x speedup** in query runtime by using the optimizer on the described query. This example showcases the scaffold's ability to dynamically compose and optimize complex data workflows while leveraging heuristics in response to user-driven goals.

F. Data Federation [RQ-3]

To verify the applicability of the scaffold ecosystem, we cross-reference its performance across four different database engines (Cassandra, MongoDB, PostgreSQL, and Redis) when querying the same datasets. We contrast the database performance on vector and raster data in order to highlight the flexibility of scaffolds to fit to any established data storage strategies a user may have, though storing data according to the data's properties is always preferable from an efficiency standpoint. Due to the complexity of federating data stores, users often adapt their data into a data store less suited to the data type to simplify retrieval for inter-dataset analyses.

TABLE III
Data federation of four database engines for two data types 10-run mean. Scaffolds fit to the user's data store ecosystem.

	Cassandra	MongoDB	Postgres	Redis
Vector shape data (Census Tracts, 85,528 records)				
Mean latency (ms)	**0.616**	1.255	0.694	0.699
Throughput (items/s)	**1563**	797	1451	1431
Total stream time (s)	**54.735**	107.354	59.386	59.761
Time to first (s)	1.463	0.357	0.499	**0.145**
Raster data (GridMET, 105,966 records)				
Mean latency (ms)	**0.429**	0.482	0.612	0.661
Throughput (items/s)	**2232**	2081	1636	1477
Total stream time (s)	**47.490**	51.076	64.860	71.725
Time to first (s)	0.955	**0.061**	0.651	0.431

Scaffolds support seamless federation, reducing the downstream impacts of storing data in a variety of targeted data stores. For example, MongoDB and PostgreSQL may face different load in terms of decoding as opposed to Redis and Cassandra, since the former two represent geospatial shapes natively while the latter two return them as GeoJSON strings that are parsed at retrieval time.

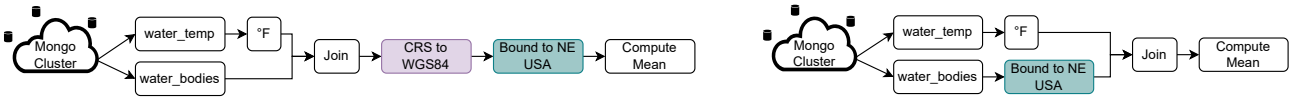
In Table III, we contrast performance first for retrieving vector shape data describing United States census tracts [50]. We also perform retrieval of raster data using five years of gridMET maximum temperature tiles.

G. High Data Diversity Correlation [RQ-2]

We evaluate the scaffold system on a complex series of operations requiring many aspects of our methodology to work together. Specifically, we combine four datasets (NH water bodies, WQ water temperature, and GM minimum and maximum temperature) as in Fig. 1.

First, we query lake boundaries from NH within a square region of the NE US (including Chesapeake Bay, Long Island Sound, and Lake Champlain), where there is high availability of water temperature measurements. Water body shapes are joined with their associated measurements from WQ. Next, we retrieve GM raster data for both minimum and maximum air temperature to compute the meteorological *respite* by subtracting them. Temporally aligning the data allows dates to be skipped where no information was found, reducing the number of tiles to be retrieved. Finally, we cut these respite tiles to each lake boundary, compute daily average respite for the lakes, and correlate these averages against daily water temperature per lake.

The bottleneck of this query is joining the raster data with the vector shapes. This process involves rasterizing the vector shapes, extracting corresponding pixels from the raster data, and finally assembling the tiles that overlap the lake. Reducing the search space of the query to limit the number of raster operations is paramount. By targeting the search space reduction via temporal alignment, we reduce the processing time from 25 hours to 8.5 minutes. The water temperature data is sparse,



(a) Before Heuristics

(b) After Heuristics

Fig. 6. Scaffold query pipelines flowcharts. 6a depicts the user defined pipeline and 6b shows the resulting pipeline after our heuristic optimizer was run. **The CRS reprojection is removed and the spatial bound is moved forward, reducing the search space for a 16x speedup**.

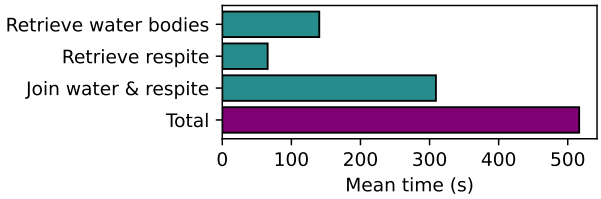


Fig. 7. Time to calculate the correlation of water and air temperatures for water bodies in the NW US, involving four datasets and several transformations. Scaffolds simplify data harmonization to support inter-dataset analyses.

illustrating the benefits of skipping unnecessary operations where there is no corresponding data. Fig. 7 shows each step's contribution to the optimized processing time, averaged over 30 runs. The time to retrieve the water temperature values (1.33 s) and perform the final correlation step (1.04 ms) were minuscule compared to the other parts of the query and as a result are not broken out from the total on the chart.

H. Materializing Scaffolds in HDFS [RQ-3]

We demonstrate the flexibility of scaffolds by materializing a scaffold derived dataset in a widely used distributed file system, Apache HDFS [51]. We materialize the transformed scaffold created in (§IV-G), with the exception of the final step (correlation), retaining the flexibility of the materialized dataset for additional analysis. Materialization causes the computed data to be placed in a file in HDFS and a new scaffold be created to represent it to include the data store. The process took roughly 8.5 minutes on average (as in the previous section), with only 4 seconds required to insert into HDFS. This example shows that scaffolds can be integrated with existing distributed storage frameworks with ease, allowing users to combine the advantages of both systems.

V. Conclusions & Future Work

We described our methodology to reconcile data diversity and a framework to support an ecosystem of services.

RQ-1: Scaffolds provide a flexible mechanism to unify diverse data regardless of encoding (§IV-A) or representational format (§IV-B). Rather than imposing limits on the types of data that can be incorporated, the scaffold abstraction provides a common framework that naturally accommodates different spatial coordinates, referencing schemes, and projection systems while simplifying unit conversions and spatiotemporal alignment. Further, scaffolds can be layered, forming new scaffolds for straightforward data harmonization. Importantly, our method remains neutral to data structure, whether as vector data, gridded datasets, or other formats.

RQ-2: Designing services around the scaffold abstraction creates a consistent interface where inputs and outputs are scaffolds. This enables easy service chaining (Fig. 1) and automatic interoperability with new services. Services work within the shared framework, without manual handling of data encoding, representation formats, unit conversions, or projection systems, simplifying development and reducing potential errors. Each service includes drill-down or finalization operations to extract meaningful results (§IV-G).

RQ-3: Our methodology enables a rich set of queries over scaffolds, supporting intersections of spatial constraints and traditional range queries. This gives users the flexibility to explore data spaces efficiently. We incorporate heuristic query optimizations (§IV-E) that dynamically reorder predicate evaluations while preserving equivalence in order to reduce search spaces and minimize latency, ensuring that queries remain both powerful and efficient. The distributed nature of the scaffold also enables server-side operations to benefit from distributed and concurrent execution (§IV-C), further improving performance. We support flexibility of use by federating storage frameworks (§IV-F) and interoperating with established distributed approaches (§IV-D, §IV-H).

As part of future work, we plan to explore distributed caches to store intermediate scaffolds, reducing cold-start costs when these scaffolds are reused in services, and integrating our distributed processing with an existing cluster manager.

References

[1] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. Van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono, "Research directions in data wrangling: Visualizations and transformations for usable and credible data," *Information Visualization*, vol. 10, no. 4, pp. 271–288, 2011.

[2] "The State of Dark Data," *Splunk Inc.*, 2019.

[3] J. T. Abatzoglou, "Development of gridded surface meteorological data for ecological applications and modelling," *International Journal of Climatology*, vol. 33, no. 1, pp. 121–131, 2013.

[4] U.S. Geological Survey, "National Hydrography Dataset," 2022.

[5] P. P. Khine and Z. Wang, "A Review of Polyglot Persistence in the Big Data World," *Information*, vol. 10, no. 4, p. 141, Apr. 2019, number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[6] P. J. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley, Aug. 2012.

[7] "The EPSG Geodetic Parameter Dataset." [Online]. Available: https://epsg.org/home.html

[8] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: a unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016.

[9] S. Kothari, J. Shah, J. Verma, S. H. Mankad, and S. Garg, "Raster Big Data Processing Using Spark with GeoTrellis," in *Computing, Communication and Learning*, S. K. Panda, R. R. Rout, M. Bisi, R. C. Sadam, K.-C. Li, and V. Piuri, Eds. Cham: Springer Nature Switzerland, 2024, pp. 260–271.

[10] "Publications - Apache Sedona™," Sep. 2025. [Online]. Available: https://sedona.apache.org/latest/community/publication/

[11] J. Yu, Z. Zhang, and M. Sarwat, "Spatial data management in apache spark: the GeoSpark perspective and beyond," *GeoInformatica*, vol. 23, no. 1, pp. 37–78, Jan. 2019.

[12] K. Jordahl, J. V. den Bossche, M. Fleischmann, J. Wasserman, J. McBride, J. Gerard, J. Tratner, M. Perry, A. G. Badaracco, C. Farmer, G. A. Hjelle, A. D. Snow, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, N. Eubank, maxalbert, A. Bilogur, S. Rey, C. Ren, D. Arribas-Bel, L. Wasser, L. J. Wolf, M. Journois, J. Wilson, A. Greenhall, C. Holdgraf, Filipe, and F. Leblanc, "geopandas/geopandas: v0.8.1," Jul. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3946761

[13] F. Silva-Coira, J. R. Paramá, S. Ladra, J. R. López, and G. Gutiérrez, "Efficient processing of raster and vector data," *PLOS ONE*, vol. 15, no. 1, p. e0226943, Jan. 2020, publisher: Public Library of Science.

[14] S. Villarroya, J. R. R. Viqueira, J. M. Cotos, and J. A. Taboada, "Enabling Efficient Distributed Spatial Join on Large Scale Vector-Raster Data Lakes," *IEEE Access*, vol. 10, pp. 29 406–29 418, 2022, conference Name: IEEE Access.

[15] GDAL/OGR contributors, *GDAL/OGR Geospatial Data Abstraction software Library*, Open Source Geospatial Foundation, 2025. [Online]. Available: https://gdal.org

[16] J. S. Discovery, "JMP Statistical Discovery."

[17] "Spotfire: Solving complex, industry-specific problems at the speed of thought." [Online]. Available: https://www.spotfire.com

[18] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for Real-Time Exploration of Spatiotemporal Datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2456–2465, 2013.

[19] C. A. L. Pahins, S. A. Stephens, C. Scheidegger, and J. L. D. Comba, "Hashedcubes: Simple, Low Memory, Real-Time Visual Exploration of Big Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 671–680, Jan. 2017.

[20] S. L. Pallickara, S. Pallickara, M. Zupanski, and S. Sullivan, "Efficient metadata generation to enable interactive data discovery over large-scale scientific data collections," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. IEEE, 2010, pp. 573–580.

[21] G. Fox, S. Pallickara, M. Pierce, and H. Gadgil, "Building messaging substrates for web and grid applications," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 363, no. 1833, pp. 1757–1773, 2005.

[22] G. Fox, H. Bulut, K. Kim, S.-H. Ko, S. Lee, S. Oh, S. Pallickara, X. Qiu, A. Uyar, M. Wang *et al.*, "Collaborative web services and peer-to-peer grids," *SIMULATION SERIES*, vol. 35, no. 1, pp. 3–12, 2003.

[23] Y. L. Simmhan, S. L. Pallickara, N. N. Vijayakumar, and B. Plale, "Data management in dynamic environment-driven computational science," in *Grid-Based Problem Solving Environments: IFIP TC2/WG 2.5 Working Conference on Grid-Based Problem Solving Environments: Implications for Development and Deployment of Numerical Software July 17–21, 2006, Prescott, Arizona, USA*. Springer, 2007, pp. 317–333.

[24] S. Lee, S. H. Ko, and G. C. Fox, "Adapting content for mobile devices in heterogeneous collaboration environments." in *International Conference on Wireless Networks*, 2003, pp. 211–217.

[25] G. C. Fox, S. H. Ko, K.-S. Kim, S. Oh, and S. Lee, "Integration of hand-held devices into collaborative environments." in *International Conference on Internet Computing*, 2002, pp. 231–250.

[26] M. Malensek, S. L. Pallickara, and S. Pallickara, "Galileo: A framework for distributed storage of high-throughput data streams," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE, 2011, pp. 17–24.

[27] M. Malensek, W. Budgaga, R. Stern, S. Pallickara, and S. L. Pallickara, "Trident: Distributed storage, analysis, and exploration of multidimensional phenomena," *IEEE Transactions on Big Data*, vol. 5, no. 2, pp. 252–265, 2018.

[28] D. Rammer, S. Lee Pallickara, and S. Pallickara, "Atlas: A distributed file system for spatiotemporal data," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019, pp. 11–20.

[29] D. Rammer, T. Buddhika, M. Malensek, S. Pallickara, and S. L. Pallickara, "Enabling fast exploratory analyses over voluminous spatiotemporal data using analytical engines," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 213–228, 2019.

[30] M. Malensek, S. Pallickara, and S. Pallickara, "Fast, ad hoc query evaluations over multidimensional geospatial datasets," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 28–42, 2015.

[31] ——, "Evaluating geospatial geometry and proximity queries using distributed hash tables," *Computing in Science & Engineering*, vol. 16, no. 4, pp. 53–61, 2014.

[32] M. Malensek, S. L. Pallickara, and S. Pallickara, "Expressive query support for multidimensional data in distributed hash tables," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing*. IEEE, 2012, pp. 31–38.

[33] S. Pallickara, J. Ekanayake, and G. Fox, "Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce," in *2009 IEEE International Conference on Cluster Computing and Workshops*. IEEE, 2009, pp. 1–10.

[34] T. Buddhika, M. Malensek, S. L. Pallickara, and S. Pallickara, "Synopsis: A distributed sketch over voluminous spatiotemporal observational streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2552–2566, 2017.

[35] T. Buddhika, S. L. Pallickara, and S. Pallickara, "Pebbles: Leveraging sketches for processing voluminous, high velocity data streams," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2005–2020, 2021.

[36] T. Buddhika, M. Malensek, S. Pallickara, and S. L. Pallickara, "Living on the edge: Data transmission, storage, and analytics in continuous sensing environments," *ACM Transactions on Internet of Things*, vol. 2, no. 3, pp. 1–31, 2021.

[37] M. Butenuth, G. v. Gösseln, M. Tiedge, C. Heipke, U. Lipeck, and M. Sester, "Integration of heterogeneous geospatial data in a federated database," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 62, no. 5, pp. 328–346, Oct. 2007.

[38] S. Ray et al., "Quantities, Units, Dimensions and Types," 2015.

[39] M. Young, S. Pallickara, and S. Pallickara, "AQUA: A Framework for Spatiotemporal Analysis and Visualizations of Water Quality Data at Scale," in *2023 IEEE International Conference on Big Data*, Dec. 2023, pp. 1555–1562.

[40] Y. Wang, "Deck.gl: Large-scale web-based visual analytics," 2019.

[41] "Recharts." [Online]. Available: https://recharts.org/en-US/

[42] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, Nov. 1972.

[43] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica*, vol. 10, no. 2, pp. 112–122, Dec. 1973, publisher: University of Toronto Press.

[44] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of the smallest area," *C.I.S.R.G. Discussion Papers (University of Hull Cartographic Information Systems Research Group)*, no. 10, Jul. 1992.

[45] D. W. Scott, "Histograms: Theory and Practice," in *Multivariate Density Estimation*. John Wiley & Sons, Ltd, 1992, pp. 47–94.

[46] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, 1st ed., ser. Monographs on statistics and applied probability. New York: Routledge, 1998.

[47] GeoTools contributors, "GeoTools The Open Source Java GIS Toolkit — GeoTools." [Online]. Available: https://geotools.org/

[48] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache Hadoop YARN: yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, ser. SOCC '13. New York, NY, USA: Association for Computing Machinery, Oct. 2013, pp. 1–16.

[49] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," in *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[50] U.S. Census Bureau, "2020 TIGER/Line Shapefiles."

[51] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. MSST '10. USA: IEEE Computer Society, May 2010, pp. 1–10.