THESIS

RAPID INTERACTIVE EXPLORATIONS OF VOLUMINOUS SPATIAL TEMPORAL DATASETS

Submitted by

Matthew Branley Young

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2025

Master's Committee:

Advisor: Shrideep Pallickara
Co-Advisor: Sangmi Pallickara

Mazdak Arabi

ABSTRACT

RAPID INTERACTIVE EXPLORATIONS OF VOLUMINOUS SPATIAL TEMPORAL DATASETS

Spatial data volumes have grown exponentially alongside the proliferation of sensing equipment and networked observational devices. In this thesis, we describe the framework aQua for performing visualizations and exploration of spatiotemporally evolving phenomena at scale, and Rubiks, which supports effective summarizations and explorations at scale over arbitrary spatiotemporal scopes, which encapsulate the spatial extents, temporal bounds, or combinations thereof over the data space of interest. We validate these ideas in the context of data from the National Hydrology Database (NHD) and the Environmental Protection Agency (EPA) to support longitudinal analysis (53 years of data) for the vast majority of water bodies in the United States. Our methodology addresses issues relating to preserving interactivity, effective analysis, dynamic query generation, and scaling. We extend the concept of data cubes to encompass spatiotemporal datasets with high-dimensionality and where there might be significant gaps in the data because measurements (or observations) of diverse variables are not synchronized and may occur at diverse rates. We consider optimizations and refinements at the server-side, client-side, and how information exchange occurs between the client and server-side. We report both quantitative and qualitative assessments of several aspects of our tool to demonstrate its suitability. Finally, our methodology is broadly applicable to domains where visualization-driven explorations of spatiotemporally evolving phenomena are needed.

ACKNOWLEDGEMENTS

My advisor Shrideep Pallickara, my co-advisor Sangmi Pallickara. This research was supported by the National Science Foundation [OAC-1931363, ACI-1553685, CNS-231231], the National Institute of Food and Agriculture [COL0-FACT-2019], and the NSF/NIFA Artificial Intelligence (AI) Institutes AI-CLIMATE Award [2023-03616].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The past couple of decades have seen an exponential growth in the amount of data being generated [1]. This growth has been fueled in part by increases in the number of data sources, falling costs for sensing equipment, increased battery capacities, improvements in the quality and capacity of networks, and the proliferation of sensors that measure features with increased precision. Other data sources include simulations and models.

A substantial amount of data that are generated is spatial, i.e., data that have spatial coordinates associated with them. These data can either be point measurements with geotagged *[lat, long]* coordinates or N-sided polygons where each vertex is defined using a *[lat, long]* tuple. The data also have a timestamp representing when those observations were made. Data collections that are geocoded alongside timestamps are spatiotemporal datasets.

As spatiotemporal datasets proliferate, researchers are interested in tools that facilitate analyses over them [2–4]. In particular, researchers are interested in identifying the implicit characteristics associated with the spatiotemporal evolution of the feature space [5–7]. These include visualizations at vast spatiotemporal scales, spatial variation of the measure of interest and also how they vary over time for longitudinal analyses [8, 9]. However, researchers are stymied in their analyses by issues stemming from data volumes, the amount of I/O that needs to be performed, and precisely identifying the data of interest [10].

Overarching objectives of this study include 1) leveraging visualizations as a complement that undergird complex data analysis over spatiotemporally evolving phenomena where both the spatial extent and time scales under consideration are vast and 2) rapid summarizations and explorations of the spatiotemporal dataspace. A key goal is to accomplish visualization driven analyses and explorations while preserving interactivity [11].

Data cubes are considered an effective mechanism to facilitate such summarizations [12–14]: we extend this concept of data cubes to spatiotemporal data spaces where the number of obser-

vations may be very large. Crucially, we support these aggregations at scale, with low latency, alongside the ability to perform these operations along diverse spatial hierarchies (administrative, watersheds, quadtiles, etc.) We explore these ideas in the context of our research prototype, RUBIKS [15].

The analysis and visualizations work in concert with each other. Visualization is one of the primary ways to explore both the raw data and the results of the analyses. These visualizations themselves inform subsequent analyses [16]. Queries identify spatiotemporal scopes where the specified predicates hold true and can be used to identify how the analyses proceed.

We validate our ideas in the context of water quality analyses. Our study involves every water body in the United States and over 2 million monitoring stations that profile and track over 4000 chemicals in these water bodies. The research questions and methodological aspects underpinning our work are broadly applicable to other spatiotemporally evolving phenomena such as urban sustainability and planning, assessing air quality, climatic clarifications, profiling environmental and ecological impacts of manmade and natural processes.

1.1 Research Challenges

Challenges in effectively supporting visualizations over spatiotemporally evolving phenomena include:

1. Dimensionality of the datasets: The datasets we consider are high-dimensional and users may be interested in summarization and exploration capabilities across all features. Our empirical benchmarks are performed over a dataset where over 4,000 chemical pollutants are being tracked.

2. The aggregations and summarization can be performed at diverse spatiotemporal scopes i.e., users can specify arbitrary chronological bounds along spatial bounds. The spatial bounds can be based on administrative boundaries, watershed, climatic regions, quad tiles, etc.

3. Complexity, number, and diversity of the visual elements: The visual elements may have a large number of vertices each with a double precision *[lat, long]* tuple associated with them. The complexity of these shapes increases the per-element rendering overheads.

4. Input/Output (I/O) overheads: Data fetching operations are I/O bound because they entail both disk I/O (retrievals) and network I/O (transmissions).

5. Variability of Features: Each visual element may have a large number of variable features associated with it.

6. The data are continually evolving with new measurements being reported continually. Furthermore no single water body has all features being tracked.

1.2 Research Questions

The crux of this study is to enable interactive visualizations and analysis over voluminous, high-dimensional spatiotemporal data. Research questions that we explore include:

RQ-1: How can we preserve interactivity during explorations?

RQ-2: How can we limit the amount of I/O being performed?

RQ-3: How can we support rich data analyses at scale?

RQ-4: How can support effective summarizations across arbitrary spatiotemporal scopes?

1.3 Approach Summary

Our methodology involves a carefully calibrated mix of (1) data preprocessing so that computationally expensive operations are not duplicated or performed in the critical path, (2) ensuring support for expressive analyses, (3) preserving interactivity, (4) coping with issues of scale, (5) supporting exploratory analyses, and 6) a novel mix of algorithmic, statistical and systems approaches to facilitate real time data summarization at scale across user-specified spatiotemporal scopes. We consider optimizations and refinements at the server-side, client-side, and information exchange between the client and server-side.

We stage and disperse the data so that the data can be collated effectively without significant data movements. Data from moderately sized spatial extents (e.g., tracts, counties, or watershed boundaries) are collated and stored on the same machine.

Summarizations allow a researcher to spot patterns that arise at diverse spatiotemporal scopes. Summarizations provided by our data cubes include min, max, mean, median, variance, standard deviations, and distributional skew and kurtosis associated with individual features (or variables). We also supplement these measures with tracking the covariance across a set of user-specified features. RUBIKS data cubes support pivot, aggregation, and disaggregation operations. Pivots allow the data cube to be probed across a specific dimension e.g., spatial, temporal, or any of the features encapsulated within the cube. The roll-up and drilldown operations relate to aggregation and disaggregation operations across spatiotemporal scopes. For example, a user may be interested in exploring the data space at coarse scales (roll-up) or at finer scales (drilldowns). These operations allow a user to specify interest over the dataspace at progressively larger (or smaller) spatial extents, time ranges, or combinations thereof.

Rather than compute these data cubes exhaustively every time a query is issued, we perform a limited number of one-time precomputations that we then leverage to support data cube operations. The smallest unit of data summarization in the system is a ***cubelet*** representing the smallest, indivisible spatiotemporal scope at which summarizations are performed. In our methodology, the scope associated with the cubelet is configurable. Data cubes are constructed from cubelets. Data cubes may either be constructed from cublets or hierarchically constructed from other cubes. We leverage Welford's algorithm to compute the cubelets in an online, single-pass fashion. Information maintained within the data cubes are also amenable to leveraging the same online method to compute data cubes at ever coarser scales. Our methodology also allows data cubes to be constructed from non-contiguous spatiotemporal cubes.

We also consider that the measurements across different variables are often not synchronized. For example, consider the case where multiple monitoring stations are profiling a water body for chemical pollutants. The pollutants may be measured at a different timepoints and frequencies.

Correlation analysis over such measurements with non-concurrent sampling between the related attributes require special consideration and interpolation. For such irregular time-series of measurements, we implement a kernel-based weighting in the computation of correlation and covariance in our cubelets.

Our pairwise covariance computations allow a user to first identify the pairs of covariances that are of interest. To reduce the number of pairwise covariances that need to be maintained in the data cube – $O(N^2)$ for N variables – we allow users to identify the set of M covariances that are of interest. Alternatively, the covariates of interest may be domain-specific or computed dynamically by the system based on occurrence of variables in queries.

In RUBIKS, cubelets are space-efficient and persistently stored since they are used in the computation of data cubes that may span diverse spatiotemporal scopes. Persistent storage of the cubelets also precludes duplicate computations alongside repeated sweeps of the data involving I/O. The data cubes, on the other hand, are ephemeral meaning they are garbage collected after a period of time.

Our summarization capabilities are backed by a distributed cache that serves two key purposes. First, the cache is used to store data cubes that have been calculated based on user-specified queries. We also store cubelets that were used to construct these data cubes; the rationale for this is that it is often the case that users are incrementally refining queries to customize the spatiotemporal scopes of interest. As such, cubelets that are part of a query have a higher likelihood of inclusion in the refinement queries. Second, the cache can reduce duplicate processing alongside any I/O that such refinements entail. Our distributed cache relies on a LRU (least recently used) caching scheme with an additional preference for storing cubelets rather than data cubes when the cache needs to evict elements during a cache miss.

Our data preprocessing operations target creation of harmonized datasets, computing intermediate values needed for downstream analyses, and producing simplified representations of complex visual elements. Water bodies in our analyses are represented using shapefiles (N-sided polygons) from the NHD. Monitoring stations managed by the U.S. Environmental Protection Agency (EPA)

report measurements for observed concentrations of diverse chemicals alongside the station's GPS coordinates and measurement timestamps. Our tool, aQua (URL for tool @ [17]), supports explorations over data reported by 2 million monitoring stations (for the 53-year duration of the data that we consider). Our data preprocessing targets associating measurements to water bodies based on their proximity and inclusion within the shapefiles. Once the data sources, water bodies, and measurements are harmonized the data analysis can proceed over water bodies where individual data items are reported as multidimensional vectors with measurement timestamps.

To support effective analysis, we include support for tracking changes in feature values over time alongside support for drill-down and roll-up analyses. We supplement this with an emphasis on user experience that blends interactivity with simplicity of query composition, rich visualization, and graphing support to analyze spatiotemporally evolving phenomena. To effectively track and graph changes in feature values over time, we leverage discretization and binning to reduce the number of observations that need to be retrieved. Our application is designed as a Web Mercator projection system. The maps are interactive, and clicking visual elements allows users to retrieve metadata and also to analyze the data using graphing tools. The visualization allows roll-ups, drill-downs, and panning to allow users to detect patterns and interactively drive the analyses.

Query composition in aQua is dynamic and steered from the visualization engine; the range of values presented to the user per-feature are based on the max and min observed values. The queries are *declarative*; the details of the query composition and predicate formulation are shielded from the user. The framework generates these queries dynamically by constraining temporal bookends, spatial extents under consideration, and the order in which the query predicates are evaluated (based on features that have been indexed) – this allows pruning of the search spaces during query evaluations.

To ensure interactivity we rely on a mix of streaming, targeted rendering operations that account for boundary conditions, and shape simplifications. We ensure that rendering operations needed for visualization are performed in parallel; boundary conditions are accounted for and once the viewport is rendered, bounding boxes just beyond the viewport boundary are rendered next to

ensure responsiveness during panning operations. We reduce computational requirements by only rendering visual elements that exceed perceptual limits: visual elements that are excluded from rendering are included for rendering as a user zooms in.

Exploratory analyses are backed using dynamic query compositions alongside the configuration of powerful defaults for several aspects with the option to override these based on the analyses being performed. A key feature we support for exploratory analyses is similarity-based analyses. Queries such as "more like this" trigger dynamic composition of the query predicates. We support two distinct mechanisms to detect similarity: instance similarity and longitudinal similarity. Given a water body of interest (the reference), instance similarity queries focus on identifying other spatial extents with observed values for each measure of interest that are within "range" of the area of interest. Longitudinal similarity is based on identifying water bodies that exhibit similarity over much longer time scales. In this case, each spatial extent is represented by a profile vector that encapsulates the average values for each feature monitored at the particular water body. With each water body now represented using a spatial profile vector, we then cluster these vectors in N-dimensional space. We use longitudinal similarity to perform the unsupervised clustering operation to cluster the spatial profile vectors. To avoid issues stemming from the curse of dimensionality as the number of features increase, we constrain the number of dimensions in which the clustering operation is performed. We allow users to override defaults for the number of clusters to be generated, the stopping criteria for clustering, the number of iterations to be performed, and the distance measures used during clustering operations. Water bodies within the same cluster are similar with respect to the features over which the spatial extents were clustered. Clustering is performed in the critical path (i.e., the users must wait for the clustering operation to complete), and once the clustering operation is completed the water bodies are clustered based on their similarity.

We also include several refinements to ensure that the system is able to scale with increases in data sources, data volumes, and the complexity of the analysis being performed. As the number of features increases, indexing every feature is infeasible given the memory overheads associated with maintaining indices. Also, while indexing operations speed up the read-side they introduce

delays during ingestion since all indices associated with that collection need to be updated as data are ingested. We manage the competing pulls of speed vs. memory requirements of indexing operations by limiting indexing to a smaller subset of features. We supplement this by generating compound indices for features that are typically used in tandem with each other. We also incorporate mechanisms to ensure that we can cope with availability of new data both from existing and new data sources - this allows our methodology to be amenable to continuous, incremental updates.

1.4 Paper Contributions

Our methodology describes visualization driven exploratory analyses of spatiotemporally evolving phenomena and includes a mix of algorithmic and systems innovations, and a framework for summarization over voluminous, high-dimensional spatiotemporal dataspaces including:

1. Support for declarative queries and a novel framework that dynamically generates query predicates with the appropriate spatial scopes, and temporal bookends.

2. Support for discretization and binning of feature values observed at a water body. We do so while constraining arbitrary temporal bounds.

3. Support for layering and apportioning of visualization tasks so that the visualizations are amenable to parallelization and rendering using GPUs.

4. Support for interactive visualizations and exploratory analyses over a large number of spatial extents, features, data sources, temporal and spatial scopes.

5. Assimilation of new data, data sources, and features (chemicals) of interest.

6. Preserving locality to minimize data movements while reducing contention in shared clusters.

7. A scalable framework that supports continuous assimilation of data, targeted I/O, and cache-residency schemes to minimize duplicate processing.

8. Our summarizations can be performed in near real-time regardless of the spatiotemporal scopes involved. Except for the pairwise covariances where users configure variable pairs of interest, the other aspects of summarizations are available for all variables of interest.

9. Our summarization schemes are backed by a distributed caching scheme that preferentially caches cubelets and data cubes to reduce disk access times and re-computation costs.

10. Finally, our methodology places no restrictions on the storage frameworks that host the voluminous datasets.

1.5 Organization

Section 2 outlines our system architecture and methodology. Section 3 includes a discussion of the empirical benchmarks, performance profiling, and findings. Section 4 covers related work and section 5 describes our conclusions and future work.

Chapter 2

Methodology

Our methodology describes the construction of hierarchical datacubes and visualization driven analysis over voluminous spatiotemporal datasets.

Data cubes may be constructed from cubelets (this smallest, indivisible unit of aggregation in the system) or from other data cubes. Data cubes generate aggregated summarization of measurements over a spatiotemporal scope at varying levels of coarseness, based on their resolution. Here, we demonstrate our methodology for computing and analyzing data cubes over disjoint spatiotemporal extents.

To facilitate visualization driven analysis our methodology addresses aspects relating to: (1) data wrangling, (2) managing perceptual limits, (3) enabling support for expressive queries that facilitate rich analyses, (4) data streaming, (5) graphing at diverse temporal and spatial scales, and (6) analyzing adherence to water quality standards.

2.1 Data Cubes - Cubelets

In the Rubik framework, a cubelet serves as the fundamental unit of aggregation and analysis. These cubelets play a pivotal role in encapsulating aggregated values across a diverse spectrum of measurements within a well-defined spatiotemporal scope. With these Cubelets, we develop a dynamic analytical framework that facilitates iteratively identifying regions of interest that satisfy desired properties or pairwise covariences.

Our cubelets encapsulate statistical summaries such as counts, means, minimums, maximums, and standard deviations, alongside distributional skew and kurtosis, for all observations for a particular variable within the specified spatiotemporal extent. The data encapsulated within a cubelet is space-efficient and is amenable to aggregations i.e., cubelets can combined to produce a new data cube that encapsulates the aggregated measures of interest.

The ability to hierarchically aggregate cubelets (and cubes) facilitate a comprehensive exploration of spatiotemporal patterns, trends, and relationships across the entirety of the dataset's geographic and temporal domain. By orchestrating queries that target data cubes, *we aim to pinpoint regions of interest that align with specific criteria or exhibit correlated behaviors with a predefined set of features*, enabling targeted analysis of intricate spatial and temporal phenomena and extraction of nuanced insights, contributing to informed decision-making in a wide array of applications.

Cubelets are created over a configurable spatiotemporal scope. The cubelets can be aggregated hierarchically into data cubes at varying spatiotemporal resolutions. We describe the hierarchical organization of cubelets in section 2.6.

In RUBIKS, cubelets are *perennial* while the data cubes are ephemeral. Cubelets are created and persisted on stable storage (and thus are perennial) along with actual data points during ingestion. These cubelets have a predetermined resolution and constitute the lowest level of the cube hierarchy. Cubes are coarser in the sense that they are computed on an on-demand basis from cubelets or other cubes in a hierarchical fashion and ephemeral (i.e., they are not persisted to disk).

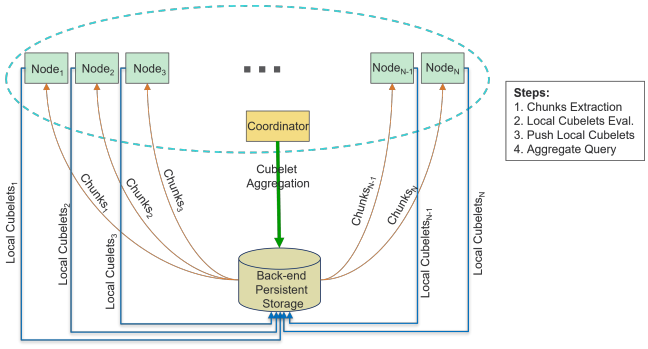

Figure 2.1: Cubelet Construction During Ingestion

2.2 Data Cubes - Cubelet Content [RQ-3, RQ-4]

Cubelets summarize data from a particular spatial extent and are constructed from persistent data stored on disk (we place no constraints on the storage framework used to store such data).

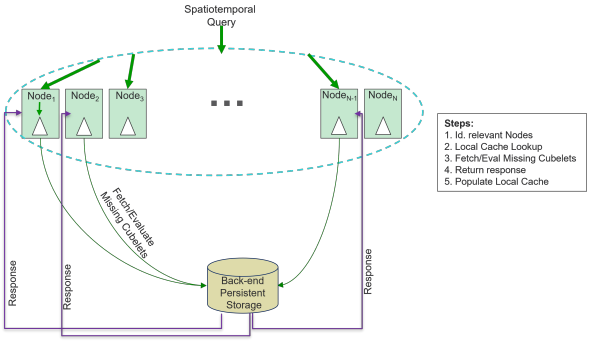

Figure 2.2: Rubik Cubelet Contruction and Fetching

Each cubelet summarizes data for a configurable but system-wide spatiotemporal scope. The cubelet comprises a set of metadata attributes recorded within that region. The supported metadata includes essential statistical measures such as count, mean, minimum, maximum, and standard deviation for each attribute.

To enhance the analytical capabilities of cubelets, for a predefined set of attribute pairs, we also maintain running covariances within each cubelet. These covariances facilitate the evaluation of Pearson correlation coefficients at runtime, enabling researchers to gain insights into the relationships between different attributes within the cubelet.

2.3 Data Cubes - Cubelet Spatiotemporal Bound [RQ-1, RQ-4]

RUBIKS offers the flexibility to construct data cubes at different spatiotemporal extents, tailored to the specific dataset, creating non-overlapping regions as the foundation for construction of data cubes. In RUBIKS, we allow data cubes to be created over varying types of disjoint geospatial bounds, such as quadtiles, Hydrologic Unit Codes (HUC), and Federal Information Processing Standards (FIPS) codes that are used by the U.S. Census Bureau. This feature enables the analysis of data with diverse spatial characteristics, accommodating datasets that might have irregular or complex geographical boundaries. By supporting multiple geospatial bounds, RUBIKS allows researchers to perform detailed analyses on localized regions while also gaining insights into broader

geographic trends, fostering a more comprehensive exploration of spatiotemporal patterns and relationships within the data.

Perennial cubelets represent the finest level of aggregation and are persisted both on-disk over our distributed storage, as well as in-memory cache that we construct over the cluster nodes. These can be hierarchically combined to create coarser aggregates – the ephemeral cubes –facilitating a multi-resolution analysis of spatiotemporal patterns and trends. This provides a powerful tool for efficient and flexible exploration of large-scale point datasets with varying granularities. Ephemeral cubes are constructed as client-queries get evaluated server-side to enable collaborative query evaluation. At the finest level, perennial cubelets are constructed and updated during data ingestion by aggregating and summarizing point data that fall within a predefined spatiotemporal extent – for instance, over a spatial bound of a HUC12 boundary and temporal bound of a single day.

2.4 Data Cubes - Distributed Ingestion: Perennial Cubelet Generation [RQ-2, RQ-4]

Perennial Cubelets are generated during data ingestion. In Fig. 2.1, we illustrate the process of generation of these cubelets. Preprocessing of incoming voluminous data in a standalone fashion can be time-consuming and compute-intensive. Rubiks relies on a distributed cluster of nodes for handling data ingestion, cubelet creation and query evaluation.

During ingestion, incoming data-points are partitioned into chunks and ingested in a distributed manner across our cluster nodes. Each node independently computes its local set of cubelets, contributing updates to a temporary set of cubelets in the distributed storage backend. Subsequently, a coordinator node initiates an aggregation query to combine local cubelets with overlapping keys, if any, into usable perennial cubelets. Only cubelets are constructed and persisted; data cubes themselves are constructed hierarchically on an on-demand basis.

2.5 Data Cubes - Cubelet Update

To adapt to the continuous updates to the underlying storage, we ensure concurrent data ingestion and the update of cubelets in persistent memory.

2.5.1 Data Cubes - Welford's Algorithm for Rapid construction/Updates [RQ-4]

To ensure efficiency and scalability within RUBIKS, we employ dynamic merging and updates of cubelets using Welford's algorithm, which provides a computationally efficient (single-pass) approach for incrementally calculating the mean and variance as new data is added or cubelets are merged. This method allows for real-time updates and analysis without the need to recompute the entire dataset, reducing both computational complexity and memory requirements.

Leveraging Welford's algorithm and associated metadata mean that our cubelets can efficiently accommodate data updates and adapt to changing input without sacrificing analytical accuracy. The algorithm's incremental, online nature makes it particularly well-suited for handling continuous data ingestion and maintaining up-to-date statistics within cubelets and across data cubes that are hierarchically constructed using cubelets and other data cubes. As a result, both cubelets and data cubes can dynamically adjust to new data points, supporting real-time analyses and ensuring a robust and scalable solution for data management and analysis. Utilizing Welford statistics for aggregation over cubelets allows us to 1) rapidly identify cubelets that require changes/creation, and 2) perform rapid, decentralized updates over our cluster.

2.5.2 Data Cubes - Correlation estimation for misaligned time series [RQ-3, RQ-4]

If two time series x and y are observed at irregular time points $\{s_i\}_{i=1}^{n_x} \neq \{t_j\}_{j=1}^{n_y}$, the empirical means $\widehat{\mu}_x, \widehat{\mu}_y$ and empirical standard deviations $\widehat{\sigma}_x, \widehat{\sigma}_y$ for the two series can be computed directly. The empirical correlation, however, can only be computed directly if the observation times are

aligned ($n = n_x = n_y$, $s_1 = t_1, s_2 = t_2, \ldots, s_n = t_n$):

$$\widehat{\rho}_{xy} = \frac{1}{n-1} \sum_{j=1}^{n} \left\{ \frac{x(t_j) - \widehat{\mu}_x}{\widehat{\sigma}_x} \right\} \left\{ \frac{y(t_j) - \widehat{\mu}_y}{\widehat{\sigma}_y} \right\}.$$

If observation times for the two series are misaligned, we use the non-rectangular kernel approach described in [18] to approximate the correlation. Let

$$K_h(s,t) = \frac{1}{h\sqrt{2\pi}} \exp\left\{ \frac{-(s-t)^2}{2h^2} \right\}$$

denote the Gaussian kernel function with bandwidth parameter h. This kernel function is used to determine which time points between the x and y series are close enough to be used in estimating the correlation, via

$$\begin{aligned} \widetilde{\rho}_{xy} \;=\; & \frac{1}{\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} K_h(s_i - t_j)} \\ & \times \left[\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \frac{x(s_i)}{\widehat{\sigma}_x} \frac{y(t_j)}{\widehat{\sigma}_y} K_h(s_i - t_j) \right. \\ & - \frac{\widehat{\mu}_y}{\widehat{\sigma}_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \frac{x(s_i)}{\widehat{\sigma}_x} K_h(s_i - t_j) \\ & - \frac{\widehat{\mu}_x}{\widehat{\sigma}_x} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \frac{y(t_j)}{\widehat{\sigma}_y} K_h(s_i - t_j) \\ & \left. + \frac{\widehat{\mu}_x}{\widehat{\sigma}_x} \frac{\widehat{\mu}_y}{\widehat{\sigma}_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} K_h(s_i - t_j) \right]. \end{aligned} \tag{2.1}$$

We compute the average distances Δ_s, Δ_t between consecutive time points in $\{s_i\}_{i=1}^{n_x}$, $\{t_j\}_{j=1}^{n_y}$, respectively, and choose $h = 0.25 \times \max\{\Delta_s, \Delta_t\}$, following [18]. As noted in [18], $\widetilde{\rho}_{xy}$ is not guaranteed to lie within $[-1, 1]$; we set it equal to the closest boundary value if it falls outside.

If information from two cubelets is to be combined, let $\{s_i^{(k)}\}_{i=1}^{n_x^{(k)}}$ and $\{t_j^{(k)}\}_{j=1}^{n_y^{(k)}}$ denote the observation time points for cubelets $k = 1, 2$. Assume that from pilot analysis a single value of h

can be determined across cubelets. Further, assume that

$$K_h\left(s_i^{(1)}, t_j^{(2)}\right) \simeq 0, \quad K_h\left(s_i^{(2)}, t_j^{(1)}\right) \simeq 0;$$

that is, a misaligned pair in two different cubelets has time points sufficiently far apart to contribute nothing to the correlation computation. Then replace each cubelet mean and standard deviation in equation (2.1) by the combined mean and standard deviation; and replace each double sum in (2.1) by adding the two corresponding double sums (one for each cubelet); e.g., replace the first double sum in the numerator by

$$\sum_{k=1}^{2} \sum_{i=1}^{n_x^{(k)}} \sum_{j=1}^{n_y^{(k)}} \frac{x(s_i^{(k)})}{\widehat{\sigma}_x} \frac{y(t_j^{(k)})}{\widehat{\sigma}_y} K_h(s_i^{(k)} - t_j^{(k)}).$$

In addition to the information already required for updating the mean and standard deviation when combining cubelets, this correlation computation requires storing for each cubelet the four distinct double sums in (2.1).

2.5.3 Data Cubes - HashGrid for Updating Cubelets

In RUBIKS, the need for continuous cubelet updates to ensure query accuracy stems from the dynamic nature of the underlying data store. To effectively accommodate this evolving data landscape, concurrent data ingestion and cubelet updates within persistent memory are pivotal. This process is orchestrated through a hashgrid-driven approach, aimed at ensuring accuracy of constructed data cubes with the evolving dataset through the following core steps:

Binary Hierarchical Hashgrid: Rubik maintains a binary hierarchical hashgrid, wherein each element corresponds to a specific cube. This hashgrid serves as a reference to indicate whether a cube is up-to-date or requires updating due to changes in the underlying data.

Coordinator-Initiated Updates: During execution of the aggregation, the coordinator node also monitors and tracks cubes that have undergone modifications since the last update. The coordinator node updates the hashgrid based on the modifications detected. Each corresponding element in the

hashgrid is updated to reflect the current status of its respective cube – indicating whether it is up-to-date or not.

Hierarchical Update Propagation: The hierarchical structure of the hashgrid streamlines the propagation of updates. The coordinator node can efficiently update higher-level hashgrid elements based on changes in the lower levels. This hierarchical mechanism ensures a streamlined and efficient update process.

Cluster-Wide Synchronization: Once the hashgrid is updated by the coordinator, this updated hashgrid is disseminated to all the cluster nodes. This push informs each node about the cube that are currently out-of-sync and cannot be used for query evaluation due to outdated information.

By leveraging this hashgrid-driven approach, RUBIKS seamlessly incorporates continuous data updates into its cubes. This process ensures that the cubes remain relevant and accurate, enabling accurate and up-to-date query evaluations even over dynamic, continually-evolving datasets.

2.6 Data Cubes - Hierarchical Aggregation of Cubelets [RQ-3, RQ-4]

The computed cubelets, which represent fine-grained spatiotemporal aggregates, are systematically organized into a hierarchical structure. This hierarchical organization is achieved through the aggregation of lower-level cubelets that lie within the bounds of a given *parent* cubelet, ensuring efficient representation and management of the cubelets. Additionally, specific hierarchical structures such as quadtiles, Hydrologic Unit Codes (HUC) and Federal Information Processing Standards (FIPS) codes are employed to cater to diverse geospatial bounds. Temporally, we allow aggregation to be in units of days, weeks, months, or years.

To form coarser aggregates at higher levels of the hierarchy, cubelets are combined. This merging process allows the creation of larger aggregations, providing a multi-resolution perspective of the data. Moreover, higher-level spatiotemporal extents are applied to encompass multiple cubes of varying types, further enhancing the versatility of the hierarchical framework. By employing these

methods, the hierarchical organization enables more insightful analysis of spatiotemporal patterns and trends within the datacubes.

The cube hierarchy (with cubelets at the lowest level and dynamically, recursively constructed data cubes) is maintained in the form of a metadata graph. However, since we can deterministically and hierarchically aggregate based on spatiotemporal bounds, there is no need to maintain actual links between the cubes themselves. We maintain these cubes as a set of hashmaps, grouped by their spatial and temporal keys, allowing targeted, efficient $O(1)$ retrievals.

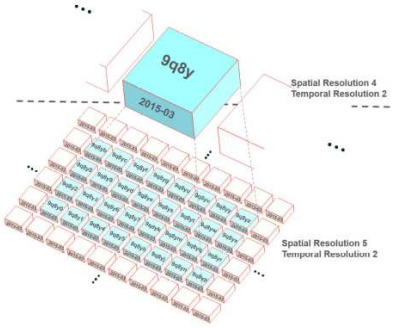

Figure 2.3: Cubelet Spatiotemporal Bounds

2.7 Data Cubes - Query Evaluation

The following is a sample spatiotemporal query that we support at client-side.

```
select pearson(iron, mercury), ...
from Aqua_Dataset
where coornidates in Polygon
and time_stamp in Query_Time
group by spatial_resolution, temporal_resolution
```

Fig. 2.2 provides an insight into the query evaluation process orchestrated by RUBIKS. The system handles analytical queries over spatiotemporal data through the utilization of datacubes. We elaborate on the overall query evaluation mechanism of RUBIKS in a distributed context. When a client query is initiated, it is initially directed towards the relevant cluster nodes (as explained

further in the subsequent section). At each node, a search is conducted within the in-memory cache for cubes that either precisely match or can be repurposed to meet the requirements of the current query. Subsequently, the query is enhanced to retrieve any unfulfilled spatiotemporal extents from the backend storage.

In the backend storage, RUBIKS engages in a search for ephemeral cubes that can be effectively employed or repurposed to furnish accurate responses for the ongoing query. For any cubes that are found missing, they are dynamically constructed from the collection of perennial cubelets and subsequently dispatched to the requesting node. In addition to addressing the query at hand, these newly formed data cubes are added to the roster of ephemeral cubes for future potential use. Data cubes created using this dynamic and targeted process are cached. By orchestrating this interplay of cache utilization, dynamic data cube construction, and query enhancements, RUBIKS realizes a robust and responsive query evaluation mechanism that ensures efficient utilization of available data and computational resources. Crucially, correctness is preserved while ensuring efficient analysis of spatiotemporal datasets.

2.8 Data Wrangling [RQ-1, RQ-2, and RQ-3]

The data leveraged in our analyses arrives from different sources including the EPA [19], USGS National Hydrology Database [20], and the US Census Bureau [21]. Our data wrangling schemes are designed to ensure (1) harmonization across diverse datasets, (2) reconciling measurements reported in different units, (3) spatiotemporally aligning observations, and (4) minimizing duplicate processing. A related goal is to ensure that visualizations are not impacted by errors or deficiencies in the data.

From the EPA's Water Quality Data dataset we ingest the Sites and Physical/Chemical collections. The Sites collection contains records represented by a geoJSON point (coordinate pair), metadata, and a unique identifier. The Physical/Chemical collection contains records represented by measurement name, unit of measurement, time measured, and an identifier which associates the data with a measurement site. At the time of this writing, this collection contains records starting

January 1st 1970 - September 2023 within CONUS (continental United States). In this study, term *measurement* refers to data points or groups of data points that have been measured: chemical compounds, water temperature, weather patterns, etc. We ingested two shapefile collections from the NHD, which has been federated by the USGS: Hydrography and Flowlines. The Hydrography collection contains records (encoded using geoJSON) representing bodies of water such as lakes, reservoirs, rivers, etc. that are represented as multi-polygons. The Flowlines collection contains records (also, encoded in geoJSON) representing water flow lines as MultiLineStrings. When we use the term water bodies, we refer to records from these two collections. Shapefiles can be rendered, colored, and extruded on a map. To support analyses based on administrative boundaries, we ingested the most recently updated Counties and States shapefile collections from the U.S. Census Bureau. We use the EPA's Water Quality Standards dataset, based on the EPA Clean Water act, for analysis of water quality standards.

2.8.1 Daily Data Ingestion

We ingest new data from the EPA's Water Quality Physical/Chemical and Sites collections daily. Our auto-ingestion script runs at midnight to ensure that the visualizations assimilate the most recent data. Our data retrievals are targeted and avoid retrievals of data that have been ingested within the system. We communicate with the EPA's servers to retrieve all new data. After performing unit coercion and duplicate/empty values handling the data are staged. To identify if new measurement sites have come online, the collected data is aggregated across Site Id and the resulting set (Set A) is compared to the set of Site Id's already in our database (Set B). We take the difference of these two sets (Set A - Set B) to find all new sites. We ingest the resulting site data and the staged measurement data into our production datasets.

2.8.2 Associating Measurements with Shapefiles

The association between measurements and shapefiles (water bodies or states/counties) is a foundational component of our data visualization. The spatial component of all our visualizations come from shapefiles and the data being visualized are measurements. Measurements have an

implicit association with the station they were taken at. Because our measurement and station data come from a different source than our shapefiles, we need to build associations between the two sources. Specifically, each station needs a reference to each shapefile that it resides within or is proximate to. This involves extensive use of inclusion, exclusion, intersections, and proximity calculations. All distance calculations in aQua are based on spherical coordinates and account for the earth's curvature.

We implemented spatial proximity queries at 50-meter resolution to aggregate stations near water bodies. In the case of shapefiles coming from the Hydrography collection, this was useful to find sites near shorelines, as the perimeter of these shapefiles represent the shoreline of a water body. In the case of the Flowlines collection this was imperative because that collection is only represented using lines, which precludes building associations based on spatial inclusion.

2.8.3 Coercing Measurement Units

In order to improve data availability, we perform unit coercions when possible. Our unit coercion typically involved converting measurement values to the smallest metric unit used in the dataset with some exceptions. In some corner cases, we converted units to a larger value, for example we converted pg/l to ng/l. To preserve accuracy, we never coerced a unit if that coercion would result in greater than a 3 order of magnitude conversion. We allow users a choice of unit coercion or not; this is possible because we do not alter the original measurement values or unit, rather we added new fields for coerced values and units.

Unit coercion improves data availability because when we aggregate measurement data for visualization we differentiate between the same pollutant/chemical being measured in different units. This is important because we cannot convert units on the fly during visualization especially when units may be nonstandard or underspecified; this is often the case for the EPA Water Quality Physical/Chemical dataset. For example, consider Phosphorus with the following units and associated number of measurements: [mg/l: 122,453, ug/l: 1,594,356, NONE: 532, stdUnits: 1,523, COLSTRAT 3: 3,285]. If we convert mg/l to ug/l we stand to gain 122,453 datapoints every time

Table 2.1: Top 10 coerced measurements in aQua

Original Unit	Converted To	Records
mg/l	ug/l	69,272,262
m	cm	3,745,295
ft	cm	2,958,144
ft3/s	m3/sec	1,604,301
mg/kg	ug/kg	1,333,580
deg F	deg C	916,404
ppm	ppt	739,802
mph	km/hr	697,310
mg/L	ug/l	548,199
cfs	m3/sec	511,192

the user wants to visualize Phosphorus. For obvious reasons we cannot do any conversions for NONE, stdUnits, or COLSTRAT 3 but we do not remove these observations because they could be meaningful to the organizations that collected them. Table 2.1 displays the top 10 units we coerced along with their respective numbers of coerced datapoints. We coerced 66 units totaling 84,994,949 datapoints.

2.9 Managing Perceptual Limits [RQ-1, RQ-2]

Our visualization and data analysis is structured as a web Mercator projection based visual analytics tool that executes within web browsers. As such, interactions with the tool are likely to trigger data movements that may adversely impact interactivity and responsiveness.

Front-end visualizations occur in the context of a map. This map may be zoomed in and out. As a user zooms out to visualize a much larger aggregate spatial extent, shapefiles below a certain surface area become impossible to discern. Trying to render more than a few thousand shapefiles at once may crash the user's web browser. We analyzed the surface area distribution in the Hydrography collection and found the following statistics for the distribution of the spatial area: Min=0.0 km^2 Max=82,002.288 km^2, Mean=9.031 km^2, and Standard Deviation: 549.182 km^2. Furthermore, we find that 30,021 records (81%) have a surface area less than 1 km^2. This

demonstrates a significant left-skew of these data with respect to surface area; see [Fig. 2.4, Fig. 2.5].

To address this, we define a set of perceptual limits represented by a zoom threshold and minimum visual surface area pair. If the map crosses either one of these thresholds we automatically filter out all shapefiles below the minimum surface area. All surface areas are measured in square kilometers. For example, consider the following {threshold: minimum surface area} mapping: {13 : 0.001}, {11 : 0.01}, {9 :0.1}, {7 : 1}. When the map is zoomed out and crosses zoom level 9, we filter out all shapefiles with a surface area less than 0.1. In other words, we only query and render shapefiles with a surface area $\geq 0.1\ km^2$. In this way we effectively constrain the search space to the subset of shapefiles which will be visible, and therefore useful, to the user.

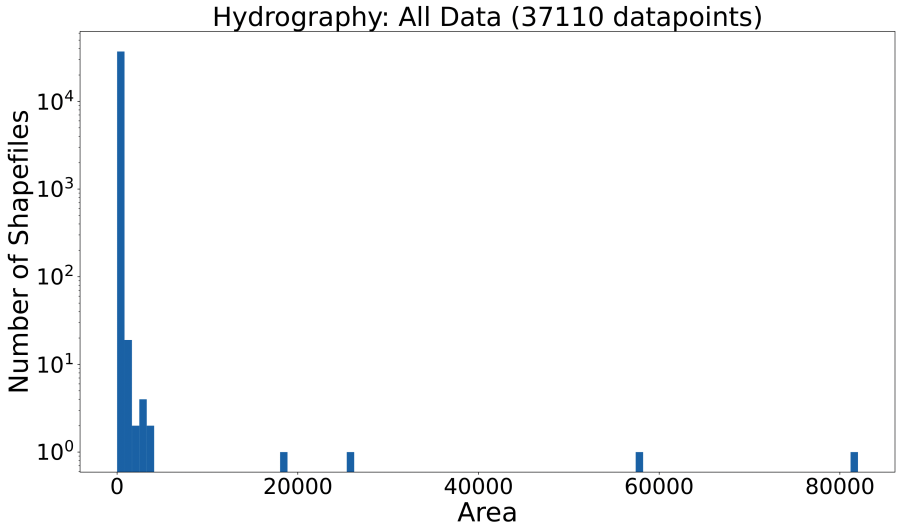

Figure 2.4: Surface Area Distribution for Hydrography Collection. The y-axis is plotted on a logarithmic scale and the x-axis is plotted on a linear scale.

2.10 Expressive Queries [RQ-2 and RQ-3]

Queries underpin our analyses. We include support for: (1) visual composition of queries, (2) longitudinal similarity, and (3) instance-based similarity. We also support indexing subsets of features to facilitate fast evaluation of queries.

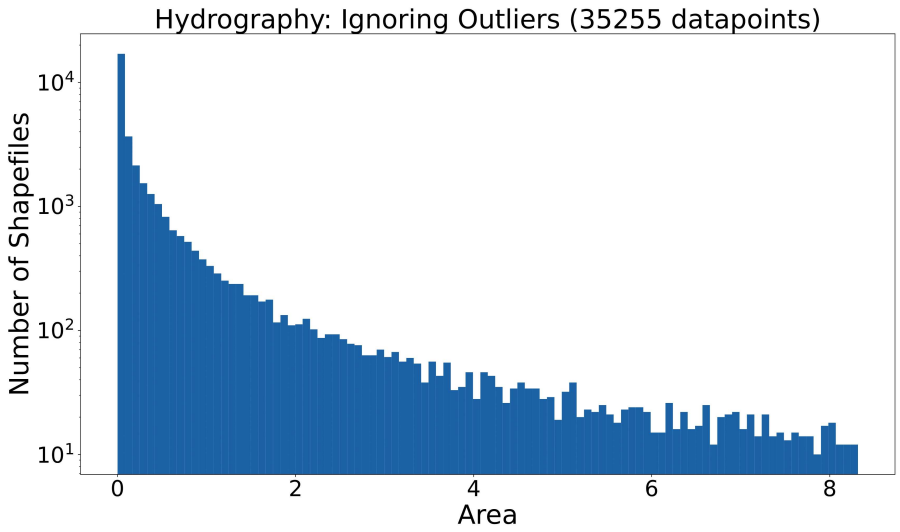

Figure 2.5: Surface Area Distribution for Hydrography (Ignoring Outliers).

Users can visually compose queries and render outcomes of those queries. The query composition includes support for dynamically refining the spatial scope of the queries (to the viewport), the temporal bounds of interest, and the ranges associated with the feature values based on summary statistics computed during the ingestion phase. Our methodology includes support for indexing a subset of features and generation of compound indices based on features most commonly used during analysis. We also include support for an Overview feature which allows users to visualize which organizations are measuring data, what the most frequently measured data are, and what the general data availability trend is like. We define data availability as the frequency of measurement collection in a given temporal range. Users will see the top 10 organizations and top 10 measurements, with an 11th category for "other". The two distributions are displayed in color-coded pie charts with data tables beneath.

We support two distinct query mechanisms to detect similarity: instance similarity and longitudinal similarity. Both instance and longitudinal similarity can specify temporal bookends over which the similarity may hold true. The instance similarity feature allows users to choose a water body, then define a similarity index in order to locate water bodies that are similar in specific ways. There are four components to the similarity index measure: (1) Surface area similarity index allows users to choose between 50 and 100 percent similarity. Let α be the user-defined percentage

as a decimal and β be the selected water body's surface area. We define a range of surface areas that satisfy our similarity index: $[\beta - ((1 - \alpha) \times \beta), \beta + ((1 - \alpha) \times \beta)]$. For example, if the selected water body's surface area is 10 and the user specifies a similarity index of 75% for surface area, our range for surface area would be [7.5, 12.5]. (2) Data availability similarity index works in a very similar way to surface area, except that it considers the data availability of water bodies instead of surface area. (3) Organization similarity index allow users to select organizations from a data table of all organizations measuring data at the selected water body. If any organizations are selected (i.e. the organization similarity index $> 0\%$) this feature will only search for water bodies that are measured by at least one of the selected organizations. (4) Measurements similarity index works similar to the organizations similarity index except that it considers measurements involving specific chemicals rather than organizations.

A user must select one or more of these four categories for an instance-based similarity query. Once the user submits their similarity index, the results streamed from the server are organized into a data table of shapes matching the query built from their similarity index. This feature is useful because it allows users to group water bodies while considering categorical information like organizations or choose a water body to use as a basis for comparison.

Longitudinal similarity identifies water bodies that are similar to each other over much longer time scales. We leverage an unsupervised learning technique to inform our longitudinal similarity: clustering. Once the temporal bounds of interest are identified, we represent each water body with a feature vector. We leverage longitudinal similarity to cluster water data across a feature vector of the user's choosing. This is useful for identifying areas with unusually high or low values for certain features. For example, a user could choose [Sodium, Chloride, Calcium] as the feature vector and identify water bodies with outlying averages for these compounds. A user could also use this feature to find counties with spiking averages for lead, arsenic, and other toxic chemicals that can leach into water sources. After selecting a shapefile collection and temporal range, the user can build a feature vector of between one and seven measurements associated with the spatial collection within the time range. This feature vector is used to aggregate shapefiles and determine

averages for each feature in the vector. This aggregation is used as input to a longitudinal similarity algorithm that uses either Euclidean or Manhattan distance measures. The clustering operation outputs a list of clusters each of which has a centroid and a list of shapefiles with associated averages for each feature in the vector.

Users can visualize clusters on the map by selecting a cluster and loading the associated shapefiles into the map. Users can then visualize time series charts for the feature vector, and compare averages across the cluster. We also support a scatter plot so that users can plot two features against each other and see the clusters in different colors. This is useful for determining which features do and do not separate the clusters.

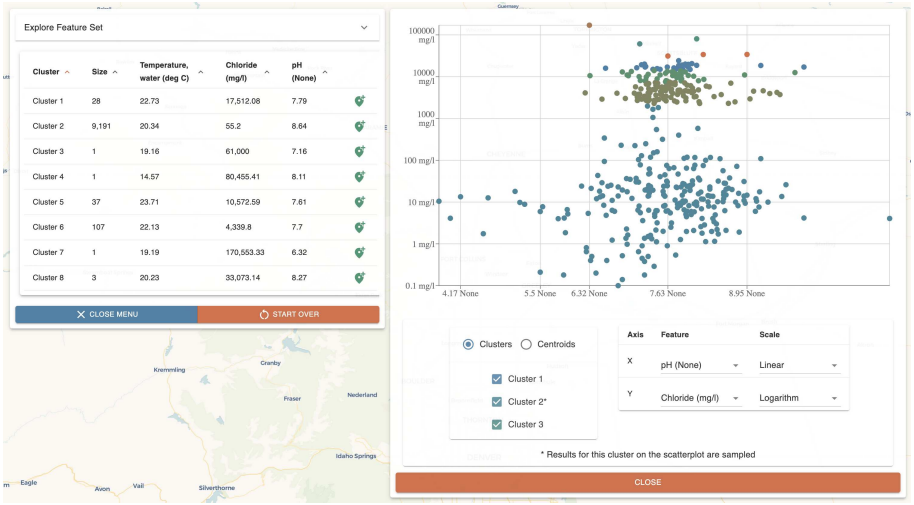

Figure 2.6: Clustering Based on Water Temperature (deg C), Chloride (mg/l), and pH (none). The cluster data table is on left, and the scatter plot is on right. The asterisk next to cluster 2 indicates that the results for that cluster are sampled on the scatter plot.

We can change the scale for either axis as well as which feature from the vector is plotted along that axis. The data table on the left [Fig. 2.6] allows users to load any of the clusters into the map at which point they may perform any of the other visualization features across that cluster.

2.11 Graphing at Micro and Macro Scales [RQ-2 and RQ-3]

After loading water bodies into the map, users can visualize time series line charts of all Physical/Chemical measurement data associated with the water bodies. This visualization can be aggregated across multiple selected water bodies or specific to a single water body. The user interface (UI) is integrated such that users can also easily view comparisons between the selected water bodies.

Due to the voluminous nature of the EPA's Water Quality data, we use discretization and binning during data processing on the server-side. Consider a water body and 30-year temporal window of measurements collected regularly by several different organizations. There may well be tens of thousands of measurements associated with that water body in the time range. Displaying these measurements individually on the front-end is infeasible. To maintain a high degree of accuracy, we define the maximum number of bins (or buckets) to be 250 where each bin represents $1/250^{th}$ of the time range and associate each measurement with the appropriate bucket. If the data are continuous, we calculate the mean of each bin. If the data are categorical, we perform a count operation on each bucket and maintain a map of category: count for each category. We chose 250 as the maximum bin number because this is the maximum number of discrete bins that our client-side charting framework can display within the horizontal pixel space allocated to it on a 13-inch laptop.

The time series analysis feature also displays data availability along the same time series. Users can visually compare observed measurement values and data availability simultaneously. This is important because data availability informs confidence in the averages for a given temporal bucket. If the current temporal window is wide, a single bucket may comprise several months of data. Data availability matters because the average of a bucket with 3,000 data points provides a higher degree of confidence than the average of a bucket with 2 data points.

2.12 Water Quality Standard Analysis [RQ-1]

This feature allows users to easily visualize which counties do or do not conform to their state's water quality standards. Users select a state, then choose a pollutant to visualize water quality standards for. The list of available pollutants comes from the EPA's water quality standards dataset for the selected state and is collated with measurement data from that state to filter out any pollutants from the water quality standards dataset which are not actually measured in the chosen state during the time frame selected. Once a user selects a pollutant, we search the Physical/Chemical measurement data for all occurrences of that pollutant within the chosen state, grouping by county. We analyze the pollutant data for each county by determining the mean pollutant value. The user sees each county in the chosen state which has measurement data for the selected pollutant. The county is colored orange if its average is above the water quality standard, and green if its average is below the water quality standard. The counties are extruded from the map; the higher the county appears on the map the farther from the standard it is [Fig. 2.7]. The heights are calculated using the normalized distance from the standard. Further, a data table is displayed which contains the average and the data availability in each county.

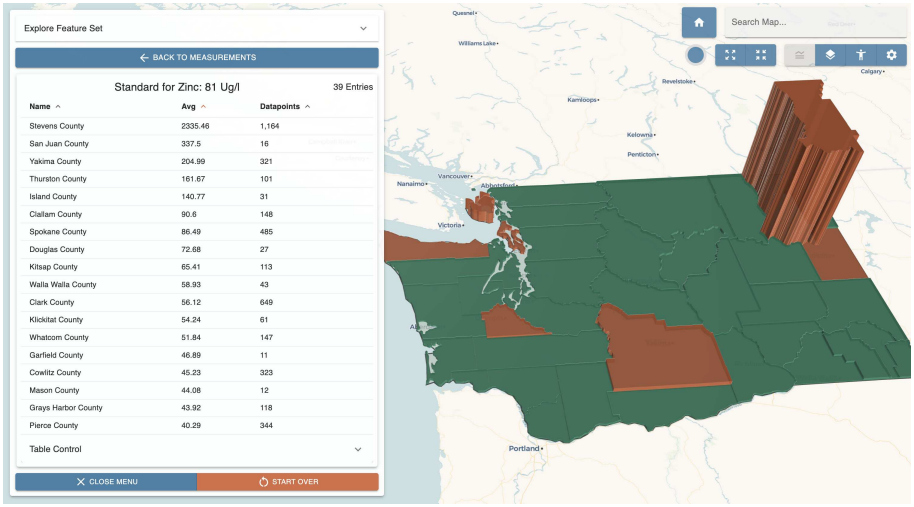

Figure 2.7: Water Quality Standard Analysis in Washington State for Zinc (ug/l). We can see that Stevens County has a significantly higher average measurement value for Zinc than the state's water quality standard. The counties of San Juan, Yakima, Thurston, and Island are also in violation of the state's standard, though less dramatically.

Chapter 3

System Evaluation: Data Cubes

3.1 Experimental Setup

To evaluate compute-intensive satiotemporal queries and ingestion rates over our system, we profiled RUBIKS over a cluster of 20 nodes. The data ingestion and query evaluation occurs in a spatiotemporally partitioned manner over the same cluster of 20 nodes. Each node in our distributed cluster is an Intel Xeon E5-2620v3, with 64 GB RAM, each with a Quadro P2200 GPU (5GB of memory) with 1280 cores and several local 7200RPM SATA hard disks. The data ingestion operations and spatiotemporal queries over the cluster get partitioned throughout the cluster uniformly based on the first 6 characters of their Quadtile key [22].

3.2 Dataset and Spatiotemporal Extent

In total, the dataset comprises of measurements from ∼991K recording stations with a total of $\sim$ 226 million data points till date with new data getting ingested daily.

3.3 Cubelet Construction Time

We evaluate the time taken to construct the *perennial cubelets* over RUBIKS in a cold-start scenario, where we have to ingest $\sim 226M$ entries into our distrubuted storage. Table 3.1 compares the time taken to construct cubelets constructed over varying non-overlapping geospatial bounds. We can see compared to the total number of records being ingested, the overall time to construct cubelets is quite low, in the order of a few minutes. Additionally, we note that the overall time taken to construct the cubelets is directly proportional to the total number of cubelets being constructed. For instance, for the total geospatial extent of the CONUS, the number of unique counties is 3163, the total number of quadtiles within the bounds is $\sim$ 15,000, whereas the total number of HUC-12

Table 3.1: Cubelet Generation: Comparison between time (seconds) taken to create Cubelets in a cold-start scenario vs daily updates

	County	**Quadtiles**	**HUC-12**
Cold-Start	187.53	219.90	363.77
Daily Updates	4.91	5.84	17.74

Table 3.2: Spatiotemporal Query: Comparison between latency (seconds) for varying sizes

	1 Month	**10 Months**	**10 Years**
Rubiks	3.3	3.32	6.51
Brute Force	12556.6	12606.2	12686.9

regions is $\sim$87,000, which directly impacts the total number of cubelets required to be created and saved into our framework. As expected, the overall cubelet construction times is proof of that.

We can also see that the update time for cubelets for incoming daily measurements is significantly low compared to a cold-start scenario, as expected. The difference in times taken for various geospatial cubelet bounds is also reflected here, as in the case of the cold-start scenario.

3.4 Fetching Data Cubes vs Raw Data

We profile the improvement in latency through our Rubiks framework, compared to that of a spatiotemporal query over raw data. Here, we profile the latency over queries of varying size. We evaluate the time taken to compute county-wise aggregate statistics per month. By keeping the spatial bounds of the query fixed to the entire CONUS, we vary the temporal extent of the query to a month, a year and a decade. Table 3.2 profiles the average time taken for each of these 3 types of queries with and without the use of Rubiks cubelets. We can see significant improvement in query times compared to fetching of raw data, with improvement ranging from 3800-2000x.

Chapter 4

System Benchmarks

In our implementation, we leverage gRPC to facilitate client/server communication. We use a Flask server and our client is written in JavaScript with React Hooks, ReCharts, and DeckGL to support GPU accelerated map rendering. We leverage a distributed data store based on MongoDB and MapReduce via aggregation pipelines for data processing. Our systems benchmarks profile ingestion times, server-side evaluation of queries, interactivity assessments and shape rendering time using the Google Lighthouse tool. We executed all benchmarks on a set of 50 Hewlett-Packard DL160-G6 machines equipped with a 6-core 2.4 GHz Intel Xeon CPU E5-2620 v3 processor and 64 GB RAM. Each machine currently runs Alma Linux with the 9.2 kernel.

4.1 Ingestion Benchmarks

We profiled the times involved in targeted retrievals of new data from the EPA servers, while accounting for new monitoring stations (data sources) and types of chemicals being tracked. We tracked ingestion times when the time gaps from the most recent ingestion was a day, a week, and a month. The observed ingestion times were 562 ms (for a day), 842 ms (for a week), and 1214 ms (for a month). The ingestion times were dominated by communication latencies.

4.2 Query Latencies

We profiled query evaluation latencies for several of our queries. We benchmarked each query made to the server-side 30 times, and report the mean and standard deviations for each query type. Several of our queries are generalized to handle different versions of the same query. For example, the measurements query can find measurements associated with a set of water bodies, measurements to build a feature vector, or measurements with water quality standards in a particular state. Both mean and standard deviation are reported in milliseconds. Because our queries can be performed across such a wide variety of spatial extents we provide two benchmarks for many

Table 4.1: Blocking times (milliseconds) reported by Lighthouse. WB: Water Bodies, M: Measurements

Task	Blocking Time	Spatial Data Complexity
Render Shapes	230	873 WB
Overview Feature	0	873 WB
Load Measurements	53.33	873 WB, 79,955 M
Time Series Chart	423.33	873 WB, 79,955 M
Shape Comparisons	236.67	873 WB, 79,955 M
More Like This	33.33	15,314 M
Water Quality Standards	0	151,557 M

of our queries at different spatial extents: one for the state of Colorado and one for the continental United Sates (CONUS). The temporal range for all of the reported benchmarks was 52 years. These benchmarks can be seen in Table 4.1.

4.3 Google Lighthouse Benchmarks

We used Google Chrome's Lighthouse tool [23] to benchmark client interactivity and to compare rendering times using a GPU accelerated map framework (DeckGL) with a standard CPU based map framework (leaflet). We report average blocking time for various rendering tasks in Table 2 and average blocking time for GPU vs CPU rendering in Table 3 and Fig 6. All blocking times are reported in milliseconds. Our benchmarks illustrated in Table 2 demonstrate that aQua is interactive despite vast spatiotemporal scope of the data that underpins the visualization and analysis and our benchmarks illustrated in Table 3 demonstrate that GPU acceleration becomes increasingly crucial with broader spatial extents.

Table 4.2: Blocking times for rendering tasks using GPU vs CPU. We chose 3,000 as the maximum because there are just over 3,000 counties within the United States, thus this is around the maximum number of shapes that we could reasonably expect a user to try and visualize at once (there are areas in which users could visualize over 3,000 water bodies at once, but they are few and far between). We chose 300 and 30 because they represent orders of magnitude backward from 3,000. Google Lighthouse provided *main thread work time* benchmarks for both GPU and CPU

Table 4.2: Blocking times (milliseconds) reported by Lighthouse. WB: Water Bodies, M: Measurements

# of Shapes Rendered	CPU Blocking Time	GPU Blocking Time
30	110	0
300	170	90
3,000	3,500	970

rendering 3,000 shapes, though not for 300 or 30. The *main thread work time* was around 300% lower for the GPU implementation.

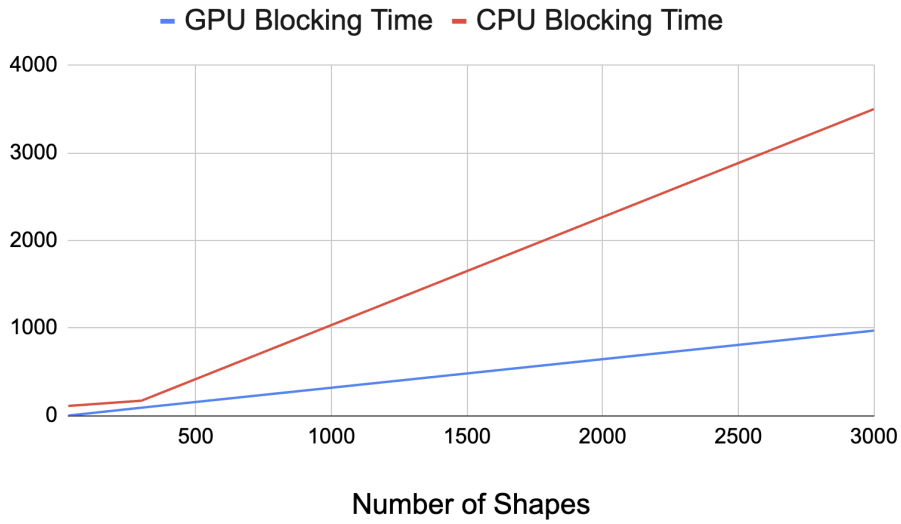

Figure 4.1: Due to the spatial density of water bodies in many regions throughout CONUS, it is common to render thousands of water bodies on the map at once. Interactivity in these spatial regions necessitates the visualization speed-ups available using a GPU accelerated map framework.

4.4 Server Benchmarks

We briefly describe these queries: (1) the shapes query is responsible for retrieving all water bodies within the map view port bounds, (2) the states query retrieves all states, (3) the "More Like This" query identifies water bodies that match a similarity index computed based on user-specified preferences, (4) the chart query is responsible for constructing chartable data by aggregating measurement values, bucketing, and averaging. (5) the SiteOrg query is responsible for identifying organizations that perform data collection activities at a set of water bodies, (6) the comparison query

Table 4.3: Query evaluation latencies in milliseconds

Query	**Mean**	**Std Dev**	**Spatial Scope**
Shapes	996.71	688.27	813 Water Bd.
States	126.67	88.57	CONUS
MoreLikeThis	914.63	2,604.58	CONUS
Chart	560.65	305.24	813 Water Bd.
Chart	947.14	461.88	Colorado
SiteOrg	619.49	623.60	813 Water Bd.
SiteOrg	2,853.65	749.58	Colorado
Comparison	749.73	235.31	813 Water Bd.
Comparison	2,187.26	569.72	Colorado
Longitudinal Similarity	73,785.14	9,820.79	CONUS
Measurements	824.42	327.55	813 Water Bd.
Measurements	12,834.84	1,524.74	Colorado

is responsible for averaging and normalizing measurement values across a collection of shapefiles, (7) the longitudinal similarity query performs the unsupervised k-means clustering [24] operation based on the user-specified set of features, and (8) the measurements query can be parametrized to retrieve measurements associated with water bodies, counties, states, or the entire United States.

Our server-side queries (Table 4.3) indicate that data exploration at the water body level usually returns results with sub-second latencies to ensure interactivity. Collate this with the client benchmarks (Table 4.1) and we find that data rendering time is under half a second. Data exploration at the state level can produce longer query times because, depending on the state, the server may be performing computations across hundreds of millions of data points. The longitudinal similarity query takes as long as it does because it is considering every data point associated with every body of water in the entire CONUS; further, the k-means clustering underpinning this query is also performed in the critical path.

Chapter 5

Related Work

Rose and Hildebrand [25] leverage WebGL to facilitate GPU Accelerated rendering of 3D molecular structures. By calling WebGL's API from the browser the authors are able to send complex rendering tasks directly to the GPU. Perrot et al. [26] describe browser-based visualizations optimized for GPU acceleration using WebGL with a focus on generating heatmaps at scale. Xie et al. [27] discuss the importance of GPU acceleration in the context of large-scale data visualization. Their research does not visualize data in the context of a web browsers and thus does not involve WebGL, though it does necessitate interactivity. The authors chose to use a GPU accelerated approach to maintain interactivity which is often lost in distributed computing settings. Li et al. [28] suggest that visualization can be defined as a function mapping data to visual primitives. Their framework ensures the availability of visualizations using a threshold which defines the minimum size of visual primitives. Lewark et. al. [29] use DeckGL to facilitate rapid visualization of vast spatiotemporal data across the entire United States.

We chose to use a WebGL based mapping framework for similar reasons that [25] and [26] chose to integrate WebGL into their research. We leveraged a WebGL-based framework (DeckGL) for GPU acceleration while harnessing mapping layer types we need for our visualizations. Crucially, the use of DeckGL frees up the CPU to maintain user interaction while rendering is taking place. While [28] defined a threshold for the minimum size of visual primitives we defined a threshold for the minimum surface area of water bodies to ensure the user's browser wouldn't crash and that we were not rendering information that is not visible to the user.

Mayorga and Gleicher [30] describe their approach to visualizing voluminous spatial data in scatter plots by defining perceptual limits which define the maximum data density visible in a given area of the screen. Healey and Sawant [31] discuss their work on perceptual level-of-detail limits for visualizing data. To avoid performing wasteful computations the authors sought to define a process for determining when there are not enough pixels on the screen to render a visual element.

Liu et al. [32] present methods for maintaining interactivity in systems designed to visualize big data. Their methods follow the principle that perceptual limits determine the scalability of visual information. Their experiments were conducted in the context of a web browser-based application which used WebGL to facilitate GPU accelerated rendering. Harrison et al. [33] attempt to determine the effectiveness of perceptual limits in defining visualization design using Weber's Law to quantify the perceived change in a given stimulus and to compare perceptual precision.

Our work is complimentary to the work described in [30] and [31] because we also implemented perceptual limits in order to deal with problems relating to interactivity in the context of vast spatial data rendering. Our work differs in the exact implementation of perceptual limits. Where [30] defined maximum data density and [31] identified minimum numbers of pixels we mapped zoom thresholds to minimum water body surface areas. This implementation of perceptual limits allows us to cut out computation relating to water bodies which would be imperceptible to the user. Further, the impact on interactivity in [33] validates our use of perceptual limits.

Stream processing frameworks such as Spark [34], Storm [35], and Samza [36] have been used to support efficient streaming in several applications. Our choice of using a Flask server was driven primarily by its lightweight properties (lower memory footprints and reduced computational overheads) that though not quite as full-featured, meets all our functional and, more importantly, performance requirements.

The concept of data cubes has gained traction as a structured means of aggregating and analyzing multi-dimensional data. These are data structures constructed dynamically or through a prefetching scheme [2, 37] in anticipation of data tiles being queried in the future to improve latency. Often, multivariate data cubes are maintained at various resolutions to enable comprehensive and flexible analysis of data across varying resolutions, providing a multi-dimensional analytical framework [13].

Techniques involving spatial hashing and distributed systems have emerged to manage large-scale spatial data. These approaches enhance scalability and enable efficient query processing in distributed environments. Cache-based storage of data in memory in a distributed fashion has also

proven to improve interactivity [38–42]. In dynamic datasets, the evolving nature of data requires adaptive analytical approaches. Existing solutions struggle to efficiently update and maintain analytical structures while accommodating continuous data changes.

The challenges of managing data and metadata [43] in the context of voluminous scientific datasets [44, 45] have been examined in various contexts, including web services, large-scale peer-to-peer grids [46–48], and collaborative environments [49, 50]. The use of distributed clusters has become instrumental in handling large-scale data processing. Techniques such as DHT-based distribution and cluster synchronization have evolved to cater to the complex requirements of spatiotemporal data [51]. Several existing frameworks address spatiotemporal data analysis, including GeoSpark [7], GeoMesa [52], STARK [53], Galileo [54], Trident [55], and Atlas [**?**] alongside support for queries that are *ad hoc* [56], geometry constrained [57], and expressive [58]. Such efforts have been also been supplemented to incorporate support for data sketches such as Synopsis [59], Pebbles [60] and Gossamer [61].

While these frameworks offer valuable insights, RUBIKS distinguishes itself through its cubelets-based approach and its specialized methods for handling evolving datasets. Additionally, algorithms that can update aggregations incrementally, like Welford's algorithm, have demonstrated their efficacy in handling real-time data changes while minimizing computational overhead [18, 62]. RUBIKS implements online update of statistics in its cubelets leading to improved speed of update.

Chapter 6

Conclusions & Future Work

In this study we described our methodology to rapid summarization and explorations of high-dimensional, voluminous spatiotemporal datasets and facilitate visualization-driven exploratory analyses over voluminous spatiotemporal data collections.

6.1 Conclusions

RQ-1: Preservation of interactivity is predicated on precomputing cublets (atomic units) that can be leveraged in the computation of data cubes. Space efficiency of the cubelets alongside storage of additional metadata allows the same data cube to be leveraged in the computation of diverse data cubes. This results in substantive reduction of latencies (up to three orders of magnitude) during analytic operations.

Dynamically pruning the rendering complexity of the visual elements based on shape simplifications and perceptual limits during drill-down and roll-up operations reduces computational requirements while preserving interactivity. Prefetching data to account for boundary conditions around the viewport allows us to ensure responsiveness during panning operations. Streaming and rendering visual elements allows incremental rendering, amortizes rendering overheads, and preserves responsiveness.

RQ-2: Relying on hierarchical spatial aggregations allows the operations to be targeted while also limiting the number of I/O operations that need to be performed. Our distributed caching schemes reduce duplicate processing and by prioritizing the residency of cubelets over ephemeral cubes reduce I/O requirements.

Our preprocessing tasks are expressed such that they have data locality reducing network I/O and contention. Accounting for perceptual limits allows us to avoid streaming and rendering visual elements. More importantly, this reduces the amount of I/O that needs to be performed. To support data retrievals at scale, we preferentially index a subset of features based on their usage and include

support for compound indices for features that are often used in tandem with each other. This allows faster query evaluations while conserving memory.

RQ-3: Declarative queries and dynamic query generation simplify the complexity of analyses tasks. Our similarity analysis allows users to specify features of interest and leverage unsupervised learning to cluster spatial extents in a multidimensional space. Users also have the ability to configure and override several aspects. Finally, our discretization and binning capabilities allow users to analyze feature value changes compactly and at scale. Leveraging GPU accelerated visualizations on the client-side allows our framework to be responsive and scale with a large number of users. This is especially critical when visualizing voluminous datasets.

RQ-4: Hierarchical aggregations alongside the online Welford's algorithm lay the groundwork for effective summarizations across diverse spatiotemporal scopes. We supplement this with a kernel based weighing of misaligned measurements to cope with the complexity of covariance computations when the measurements across variables are not synchronized in time and when the number of discrete measurements across variables are different. Deterministic identification of spatial scopes for aggregation (based on hierarchical prefix matching) alongside identification of temporal bounds allow us to be very targeted in the cubelets that are involved the calculation of data cube. We support diverse spatial extents: schemes that we currently support include administrative boundaries, watershed boundaries, and quad tiles.

6.2 Future Work

As part of future work, we propose to explore imputation schemes based on spatiotemporal variation in the pollutants. Our current work lays the groundwork for identifying data sources that are malfunctioning based on the rates, times, and gaps in data availability. We propose to fit lightweight timeseries models over data at different spatial aggregations. Because these timeseries models can capture trends and seasonality in the data, they can be used to both forecast expected values for variables of interest and to impute values where gaps exist.

Bibliography

[1] Mazzetti P. Santoro M. Papeschi F. Craglia M. Nativi, S. and O. Ochiai. Big data challenges in building the global earth observation system of systems. *Environmental Modelling & Software*, 2015.

[2] Pallickara S Pallickara SL. Mitra S, Khandelwal P. Stash: Fast hierarchical aggregation queries for effective visual spatiotemporal explorations. *In2019 IEEE International Conference on Cluster Computing (CLUSTER)*, 2019.

[3] Malensek M. Koontz, J. and S. Pallickara. Geolens: Enabling interactive visual analytics over large-scale, multidimensional geospatial datasets. *IEEE/ACM International Symposium on Big Data Computing*, 2014.

[4] Daniel Rammer, Sangmi Lee Pallickara, and Shrideep Pallickara. Atlas: A distributed file system for spatiotemporal data. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 11–20, 2019.

[5] Wang F. Vo H. Lee R. Liu Q. Zhang X. Aji, A. and J. Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 2013.

[6] A. Eldawy and M.F. Mokbel. Spatialhadoop: A mapreduce framework for spatial data. *IEEE 31st international conference on Data Engineering*, 2015.

[7] Wu J. Yu, J. and M. Sarwat. eospark: A cluster computing framework for processing large-scale spatial data. *In Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015.

[8] P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. *In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012.

[9] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Communications of the ACM*, 2012.

[10] K. Bruhwiler and S. Pallickara. Aperture: Fast visualizations over spatiotemporal datasets. *In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019.

[11] J.J. Van Wijk. The value of visualization. *IEEE Visualization*, 2005.

[12] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1:29–53, 1997.

[13] Lauro Lins, James T Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[14] Guoren Wang, Yue Zeng, Rong-Hua Li, Hongchao Qin, Xuanhua Shi, Yubin Xia, Xuequn Shang, and Liang Hong. Temporal graph cube. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[15] Saptashwa Mitra, Matt Young, Jay Breidt, Sangmi Pallickara, and Shrideep Pallickara. Rubiks: Rapid explorations and summarization over high dimensional spatiotemporal datasets. In *Proceedings of the IEEE/ACM 10th International Conference on Big Data Computing, Applications and Technologies*, BDCAT '23, New York, NY, USA, 2024. Association for Computing Machinery.

[16] Jakob Nielsen. Powers of 10: Time scales in user experience. Technical report, NNGroup, 2009. Available: https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/.

[17] Project Sustain: aQua. Available: https://urban-sustain.org/aqua-client/.

[18] Kira Rehfeld, Norbert Marwan, Jobst Heitzig, and Jürgen Kurths. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404, 2011.

[19] United States Environmental Protection Agency, 'Water Quality Physical/Chemical', US EPA [online]. Available: https://www.epa.gov/waterdata/water-quality-data-download.

[20] United States Geological Survey, 'National Hydrology Database', USGS [online]. Available: https://www.usgs.gov/national-hydrography/access-national-hydrography-products.

[21] United States Census Bureau, 'United States Counties', US Census [online]. Available: https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html.

[22] Quadtiles, November 2018.

[23] Šumak B. Heričko, T. Towards representative web performance measurements with google lighthouse. *In Proceedings of the 2021 7th Student Computer Science Research Conference*, 2021.

[24] K.P. Sinaga and M.S. Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 2020.

[25] Peter W. Hildebrand Alexander S. Rose. Ngl viewer: a web application for molecular visualization. *Nucleic Acids Research, Volume 43, Issue W1*, 2015.

[26] N. Hanusse F. Lalanne A. Perrot, R. Bourqui and D. Auber. Large interactive visualization of density functions on big data infrastructure. *IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, 2015.

[27] F. Sauer . Xie and K. L. Ma. Fast uncertainty-driven large-scale volume feature extraction on desktop pcs. *IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, 2015.

[28] H. Matsuzaki X. Li, A. Kuroda and N. Nakajima. Advanced aggregate computation for large data visualization. *IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, 2015.

[29] Everett Lewark, Matthew Young, Paahuni Khandelwal, Sangmi Lee Pallickara, and Shrideep Pallickara. Periscope: A Framework for Visualizations of Multiresolution Spatiotemporal Data at Scale . In *2024 IEEE International Conference on Big Data (BigData)*, pages 1373–1380, Los Alamitos, CA, USA, December 2024. IEEE Computer Society.

[30] A. Mayorga and M. Gleicher. Splatterplots: Overcoming overdraw in scatter plots. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[31] Christopher G. Healey and Amit P Sawant. On the limits of resolution and visual angle in visualization. *Association for Computing Machinery*, 2012.

[32] Jiang B. Liu, Z. and J Heer. immens: Real-time visual querying of big data. *Computer Graphics Forum*, 2013.

[33] S. Franconeri L. Harrison, F. Yang and R. Chang. Ranking visualizations of correlation using weber's law. *IEEE Transactions on Visualization and Computer Graphics*, 2014.

[34] Das T. Li H. Hunter T. Shenker S. Stoica I Zaharia, M. Discretized streams: Fault-tolerant streaming computation at scale. *In Proceedings of the twenty-fourth ACM symposium on operating systems principles*, 2013.

[35] Bhagat N. Fu M. Kedigehalli V. Kellogg C. Mittal S. ... Taneja S Kulkarni, S. Twitter heron: Stream processing at scale. *In Proceedings of the 2015 ACM SIGMOD international conference on Management of data*, 2015.

[36] Paramasivam K. Pan Y. Ramesh N. Bringhurst J. Gupta I. Campbell R. H. Noghabi, S. A. Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 2017.

[37] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

[38] Rui Li, Jiapei Fan, Xinxing Wang, Zhen Zhou, and Huayi Wu. Distributed cache replacement method for geospatial data using spatiotemporal locality-based sequence. *Geo-spatial Information Science*, 18(4):171–182, 2015.

[39] Rui Li, Wei Feng, Huayi Wu, and Qunying Huang. A replication strategy for a distributed high-speed caching system based on spatiotemporal access patterns of geospatial data. *Computers, Environment and Urban Systems*, 61:163–171, 2017.

[40] Rui Li, Yinfeng Zhang, Zhengquan Xu, and Huayi Wu. A load-balancing method for network giss in a heterogeneous cluster-based system using access density. *Future Generation Computer Systems*, 29(2):528–535, 2013.

[41] Shaoming Pan, Lian Xiong, Zhengquan Xu, Yanwen Chong, and Qingxiang Meng. A dynamic replication management strategy in distributed gis. *Computers & geosciences*, 112:1–8, 2018.

[42] Sanjoy Paul and Zongming Fei. Distributed caching with centralized control. *Computer Communications*, 24(2):256–268, 2001.

[43] Sangmi Lee Pallickara, Shrideep Pallickara, Milija Zupanski, and Stephen 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*, pages 573–580. IEEE, 2010.

[44] Sangmi Lee Pallickara, Marlon Pierce, Qunfeng Dong, and ChinHua Kong. Enabling large scale scientific computations for expressed sequence tag sequencing over grid and cloud computing clusters. In *PPAM 2009 Eight International Conference on Parallel Processing and Applied Mathematics Wroclaw, Poland*, 2009.

[45] Sangmi Lee Pallickara, Shrideep Pallickara, and Marlon Pierce. Scientific data management in the cloud: A survey of technologies, approaches and challenges. *Handbook of Cloud Computing*, pages 517–533, 2010.

[46] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, and Harshawardhan Gadgil. Building messaging substrates for web and grid applications. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1757–1773, 2005.

[47] GC Fox, Hasan Bulut, Kangseok Kim, Sung-Hoon Ko, Sangmi Lee, Sangyoon Oh, Shrideep Pallickara, Xiaohong Qiu, Ahmet Uyar, Minjun Wang, et al. Collaborative web services and peer-to-peer grids. *SIMULATION SERIES*, 35(1):3–12, 2003.

[48] Yogesh L Simmhan, Sangmi Lee Pallickara, Nithya N Vijayakumar, and Beth 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*, pages 317–333. Springer, 2007.

[49] Sangmi Lee, Sung Hoon Ko, and Geoffrey C Fox. Adapting content for mobile devices in heterogeneous collaboration environments. In *International Conference on Wireless Networks*, pages 211–217, 2003.

[50] Geoffrey C Fox, Sung Hong Ko, Kang-Seok Kim, Sangyoon Oh, and Sangmi Lee. Integration of hand-held devices into collaborative environments. In *International Conference on Internet Computing*, pages 231–250, 2002.

[51] Randall T Whitman, Michael B Park, Sarah M Ambrose, and Erik G Hoel. Spatial indexing and analytics on hadoop. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 73–82, 2014.

[52] James N Hughes, Andrew Annex, Christopher N Eichelberger, Anthony Fox, Andrew Hulbert, and Michael Ronquest. Geomesa: a distributed architecture for spatio-temporal fusion. In *Geospatial informatics, fusion, and motion video analytics V*, volume 9473, pages 128–140. SPIE, 2015.

[53] Stefan Hagedorn, Philipp Gotze, and Kai-Uwe Sattler. The stark framework for spatio-temporal data analytics on spark. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, 2017.

[54] Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. Galileo: A framework for distributed storage of high-throughput data streams. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pages 17–24. IEEE, 2011.

[55] Matthew Malensek, Walid Budgaga, Ryan Stern, Shrideep Pallickara, and Sangmi Lee Pallickara. Trident: Distributed storage, analysis, and exploration of multidimensional phenomena. *IEEE Transactions on Big Data*, 5(2):252–265, 2018.

[56] Matthew Malensek, Sangmi Pallickara, and Shrideep Pallickara. Fast, ad hoc query evaluations over multidimensional geospatial datasets. *IEEE Transactions on Cloud Computing*, 5(1):28–42, 2015.

[57] Matthew Malensek, Sangmi Pallickara, and Shrideep Pallickara. Evaluating geospatial geometry and proximity queries using distributed hash tables. *Computing in Science & Engineering*, 16(4):53–61, 2014.

[58] Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. Expressive query support for multidimensional data in distributed hash tables. In *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, pages 31–38. IEEE, 2012.

[59] Thilina Buddhika, Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. Synopsis: A distributed sketch over voluminous spatiotemporal observational streams. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2552–2566, 2017.

[60] Thilina Buddhika, Sangmi Lee Pallickara, and Shrideep Pallickara. Pebbles: Leveraging sketches for processing voluminous, high velocity data streams. *IEEE Transactions on Parallel and Distributed Systems*, 32(8):2005–2020, 2021.

[61] Thilina Buddhika, Matthew Malensek, Shrideep Pallickara, and Sangmi Lee Pallickara. Living on the edge: Data transmission, storage, and analytics in continuous sensing environments. *ACM Transactions on Internet of Things*, 2(3):1–31, 2021.

[62] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.