Rivulet: A Distributed Sketch for Voluminous Spatio-Temporal Observational Streams to Support High Throughput Query Evaluations

Name1, Name2, Name3, Name4
Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA
{name1,name2,name3,name4}@cs.colostate.edu

ABSTRACT

Abstract goes here.

1. INTRODUCTION

The proliferation of remote sensing equipment (such as radars and satellites); networked sensors and monitoring equipment; and commercial applications such as mapping services, location-based recommendation services and sales tracking have resulted in the exponential growth of spatiotemporal data. Spatiotemporal data comprise observations where in addition to the features of interest (such as humidity, air quality, disease prevalence, sales, etc) both the spatial and chronological components representing the time and location where measurements were made are available.

Spatiotemporal data includes a wealth of information that can be used to inform: decision making, scientific modeling and simulations, and resource allocations in several domains. These domains include atmospheric science, epidemiology, environmental science, geosciences, smart-city settings, and commercial applications. Querying these voluminous datasets and the expressiveness of these queries underpin efficiency in these settings. To be effective, regardless of the data volumes, the query evaluations must be in real-time and at high-throughput meaning a large number of queries must be evaluated per second.

Spatiotemporal datasets are naturally multidimensional with multiple features of interest being reported/recorded continually for a particular timestamp and geolocation. The values associated with these features are continually changing — in other words, the feature space represented by these datasets are continually evolving.

Queries specified over these datasets may have wide range of characteristics encompassing the frequency at which they are evaluated, and the spatiotemporal scope associated with these queries. The crux of this paper is to support query evaluations over continually arriving observational data. The queries may be continuous or discrete, involve sliding windows, and varying geospatial scopes.

1.1 Challenges

Support for real-time evaluation of queries discrete and continuous over a feature space that is continually evolving introduces unique challenges. These include:

- Data volumes: It is infeasible to store all the observational data. This is especially true if the arrival rates outpace the rate at which data can be written to disk.
- Data arrival rates: The data may arrive continually and at high rates. Furthermore, this rate of arrivals may change over time.
- Accuracy: Queries specified by the user continuous or one-time must be accurate.
- Continuous query evaluations: Users should be able to register query evaluations over the observational space that must be continually evaluated.
- Spatiotemporal data characteristics: Queries that are issued by a user may target spatiotemporal properties associated with these observations. For e.g., a user may be interested in feature characteristics for a geospatial location at a particular daily interval (say 2:00-4:00 pm) over a particular chronological range (say, 2-3 months)

1.2 Research Questions

1. How can we generate compact, memory-resident (i.e. the sketch) representations of the observational space? The sketch must be amenable to fast, continuous updates to ensure that the sketch is representative of the observational space.
2. How can we scale effectively in situations where the observations arrive at a rate faster than the rate at which the sketch can be updated? The density and arrival rates for observations vary based on the geospatial characteristics of the data. For example, in a smart-city setting NYC would like have a far higher rate of observations than a mid-sized city like Denver, which in turn would have a far higher rate of observations than a small city such as Fort Collins.

3. How can we effectively account for the spatiotemporal attributes associated with both the observations and the queries specified by the users? This spans the generation and updates of the sketch, scaling in response to high data arrival rates, and the evaluation of the queries.

4. How can we ensure high-throughputs in the assimilation of observations, updates to the sketch, and query evaluations? A particular challenge is preservation of accuracy in query evaluations despite high data arrivals, scaling in response to changes in system load, and concurrent query evaluations. This encompasses using horizontal scaling to proactively alleviate hotspots, maintaining a distributed representation of the sketch, and effective distribution, targeting, and combining of query evaluations.

1.3 Methodology

Our methodology targets: (1) creation of the sketch, (2) updating the sketch in response to data arrivals, (3) identifying performance hotspots via targeted alleviation of performance hotspots, (4) support for splitting and fusing a sketch, and (5) dynamic creation of distributed sketch. (6) using the distributed sketch to perform query evaluations, and (7) achieve high-throughput by minimizing synchronization between different portions of the distributed sketch.

Extract metadata from observations Organize this metadata in an in-memory sketch.

Our sketch supports 3 key features. Specifically, the sketch (1) maintains a compact representation of the observations seen so far, (2) is amenable to splitting and fusing, and (3) supports query evaluations including specification of sliding window sizes and geospatial scopes to support exploring properties of the observational space. The sketch is naturally amenable to support distributed representations; with each node within the cluster holding information about the observational space. The distributed representation of the sketch is that of a dynamic, continually-updated in-memory store.

There are several advantages to a distributed representation of the sketch. (1) It allows us to maintain a richer (finer-grained) representation of the feature space. (2) it improves accuracy during query evaluations, (3) supports multiple, concurrent query evaluations with each node evaluating queries independently.

Furthermore, at each node we dynamically adjust the representativeness of the sketch. There are two options here: bias towards the recent past or equal representations. In the former approach, the sketch targets finer-grained representations of observations in the recent past, and progressively coarser grained representations for the farther past. This scheme ensures that the apportioning of the available memory for the sketch is proportional to the time-intervals under consideration. For example, last 3 minutes, last 3 hours, last 3 days, last 3 months, last 3 years, and last 3 decades would have equal memory footprints. In the latter approach, the sketch maintains a coarser-grained representations at each node.

The sketch scales (up or down) dynamically to cope with data arrival rates. At each node, the expressiveness of the sketch can be tuned dynamically based on the available memory. At each node, the sketch continually updates the expressiveness of the portions of the feature space, dynamically reorients the graph-based structure, and minimizes traversals (using Bloom Filters) to support faster evaluations and preserve high-throughput.

1.4 Paper Contributions

In this paper we presented a framework for supporting real-time query evaluations over voluminous, time-series data streams. Our specific contributions include:

- Design of the Trident sketch that maintains compact, distributed representations of the observational space. At each node, the sketch supports dynamic reorientations to conserve memory and support faster query evaluations.

- Support for dynamic scaling of the sketch in response to the data arrival rates. Upscaling operations support targeted alleviation of hotspots. Only geographic locations that are observing high data arrival rates are scale. The framework manages the complexity of identifying these hotspots, splitting the particular portion of the sketch, and migrating relevant portions of the sketch. Most importantly, the framework achieves this while maintaining acceptable levels of accuracy in query evaluations.

- Support for one-time and continuous evaluations of the observational space. We incorporate support for CQL queries. Query evaluations can target arbitrarily sized (chronological) window sizes and geospatial scopes.

- Support for high-throughput, distributed evaluation of queries. To our knowledge, Trident is the first sketch designed specifically for geospatial observational data. We have validated the suitability of our approach through a comprehensive set of benchmarks with real observational data.

[1]

1.5 Paper Organization

The remainder of this paper is organized as follows...

2. SYSTEM OVERVIEW

3. METHODOLOGY

3.1 Construction of the Sketch

Metadata extraction and organization
Fine-grained/coarse-grained representations
Dynamic reorientations of the graph
Representativeness across time intervals

3.2 Stream Partitioning

We use the Geohash algorithm [2] to balance load and partition incoming data streams across processing resources. Geohash divides the earth into a hierarchy of bounding boxes identified by Base 32 strings; the longer the Geohash string, the more precise the bounding box. Figure 1 illustrates this hierarchy. Most of the eastern United States is contained within the bounding box described by Geohash string *D*, while *DJ* encompasses substantial parts of Florida, Georgia, and Alabama. The bounding box *DJKJ* (highlighted in

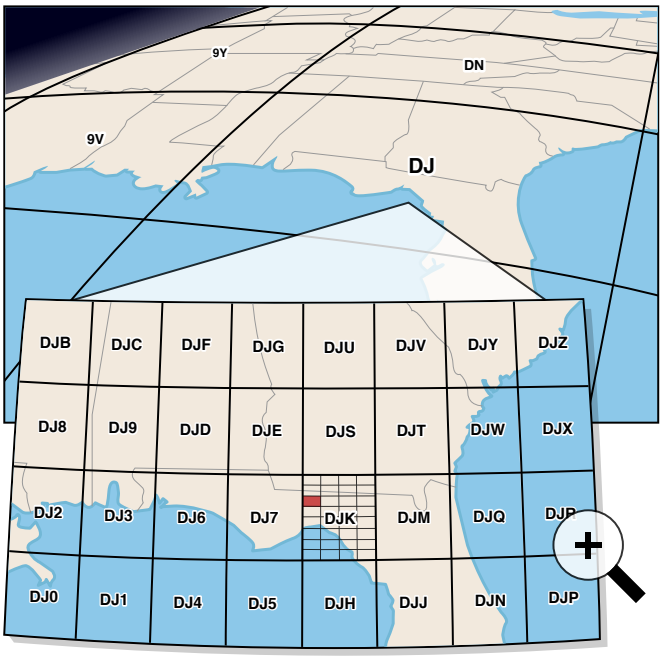

Figure 1: A demonstration of the Geohash algorithm. Each additional character in a Geohash string describes a finer-grained region; Geohash *DJ* contains substantial parts of Florida, Georgia, and Alabama, USA, while *DJKJ* (highlighted in red) encompasses Tallahassee, the capital of Florida.

red) contains Tallahassee, Florida. This hierarchical representation enables Rivulet to cope with both low- and high-density regions: several resources may be tasked with managing streams originating in and around large cities, while rural areas fall under the purview of a single node.

To achieve fine-grained control over our Geohash partitions, we operate at the bit level rather than Base 32 character level when routing streams. Each bit added to a Geohash string reduces its scope by half, with each character represented by five bits ($2^5 = 32$). In other words, a four-character Geohash string represents 20 spatial subdivisions. This property allows us to manage and allocate resources across a wide variety of observational densities.

3.3 Coping with High Data Rates: Scaling out

3.4 Downscaling

3.5 Query Evaluations

Accounting for data arrival rates, accuracy requirements, memory pressures, and the capacity of network links result in dynamic upscaling and downscaling of nodes that comprise Rivulet. The result of these scaling operations is a distributed sketch, with constituent nodes holding compact representations of observational data from different geographic locations.

Information about nodes that comprise the Rivulet sketch is maintained as a prefix tree at each node. Addition and removal of nodes comprising the distributed sketch due to scaling maneuvers are propagated through the system and the prefix trees updated. Information within the prefix trees

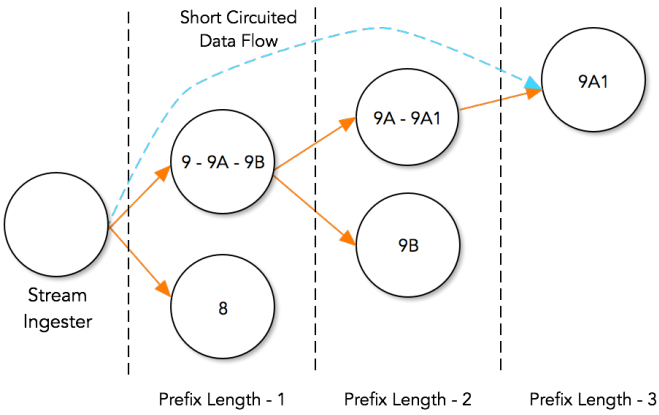

Figure 2: An example of the stream partitioning based on Geohashes of the stream packets

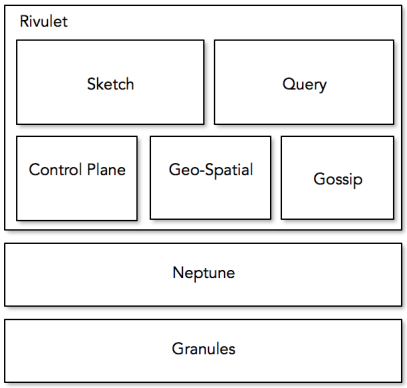

Figure 3: Rivulet is implemented as a specialized layer for geo-spatial data on top of Neptune stream processing system..

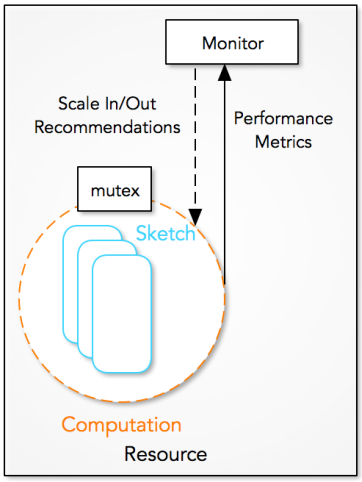

Figure 4: Dynamic scaling is triggered by monitoring the backlog of each computation and memory pressure incurred in each process.

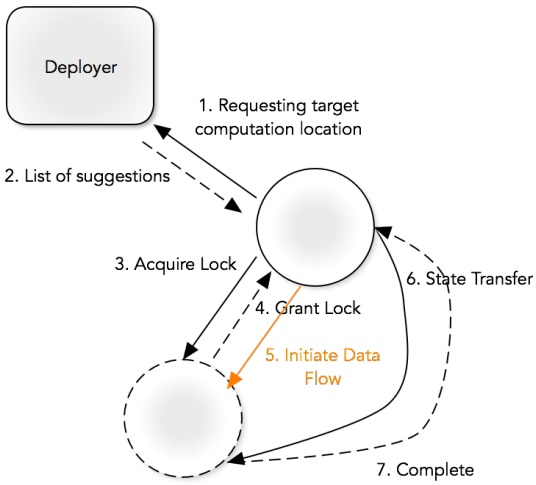

Figure 5: Scale out protocol

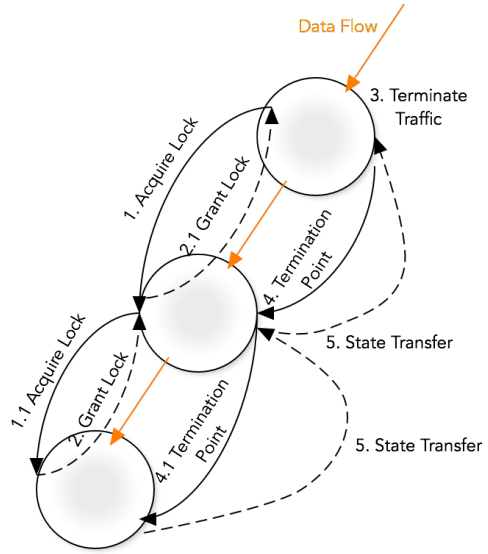

Figure 6: Scale in protocol

is designed to be eventually consistent. Alongside each vertex in this prefix tree, we maintain information about the spatiotemporal scope/range and the set of features available at the node.

Rivulet incorporates support for discrete and continuous queries specified by users. Queries specified by a user are evaluated over the distributed sketch. Depending on the spatial scopes that may be implicitly or explicitly specified within these queries, query evaluations will be performed on one or more nodes comprising the distributed sketch.

The entry point for these queries may be any of the nodes comprising the distributed sketch we refer to this node as the conduit for the particular query. During query evaluations, a first step is to identify the set of Rivulet nodes that must be targeted for query evaluations. The conduit node consults its prefix tree to identify nodes to contact based on the spatial, chronological, and particular features implied in the query. Conduit nodes are responsible for forwarding queries to the relevant nodes comprising the distributed sketch where the query must be evaluated. The conduit node also responds to the client with the list of nodes that will be responding to the query.

Results from query evaluations performed over the distributed sketch are streamed back to the client. Precisely what is streamed depends on the nature of the query. For example, if a client is interested in retrieving average temperatures in Colorado for July 2015 and if there are multiple nodes (within Rivulet) that hold relevant portions of this data; then the query must be forwarded to all these nodes. The results returned from each of these nodes would include ¡temperature, frequency¿ tuples that would then be combined at the client side to compute the final average temperature.

Alternatively, Rivulet nodes participating in the query evaluations will return vertices and edges (from their internal sketches) that satisfied the specified query constraints. The vertices and edges will be restricted to those pertinent to the query. For example, if a query targets a particular feature (say humidity) then vertices/edges relating to other features not targeted by the query will be pruned and not returned. Spatiotemporal features are exempt from this constraint. As vertices and edges are streamed back to the client from multiple nodes comprising Rivulet, these are organized into a graph that is then used to return the final results.

The prefix tree also allows query evaluations during both upscaling and downscaling operations. During query evaluations, queries are forwarded to child nodes (created during a recent upscaling operation). When the conduit forwards a query to a Rivulet node and the node is unavailable, it is either due to a failure at that node or a downscaling operation that merges the child node with its parent node. When a node is unreachable, the conduit forwards the query to the parent node.

3.6 Types of Queries

Queries within Rivulet can be discrete or continuous queries. Unlike discrete queries that are evaluated once, continuous queries are evaluated periodically or when observations become available. Though observations are never stored on stable storage, queries specified by users may target spatial and chronological scopes. In the case of continuous queries, the chronological range implicitly evolves as new observa-

tions trickle in; in the case of windowing operators, the specified chronological window continually slides over the observational streams. We classify queries discrete or continuous supported by Rivulet into 5 broad queries. These include:

1. Filter queries that specify inequality/relational queries along one or more dimensions.
2. Statistical queries allow users to explore statistical properties of the observational space. For example, users can retrieve and contrast correlations between any two features at different geographic locations at the same time. Alternatively, queries may contrast correlations between different features at different time ranges at the same geographic location. Statistical queries also support retrieval of the mean, standard deviation, and also of feature outliers based on Chebyshevs rule about the distribution of feature values.
3. Density queries support analysis over the distribution of values associated with a feature over a particular spatiotemporal scope. These include kernel density estimations, estimating the probability of observing a particular value for an observation, and determining the deciles and quartiles for the observed feature.
4. Set Queries target identification of whether a particular combination of feature values was observed, estimating the cardinality of the dataset, and identifying the frequencies of repeated observations.
5. Inferential queries are intended to support projections about how the feature is expected to evolve over a spatiotemporal scope. In this study, these projections are predicated on the registration of time-series models at the particular scope. Specifically, we rely on exponential smoothing methods to make forecasts about the estimated values of features at different time points. Besides compact representations and faster evaluations that are amenable to keeping pace with high data arrival rates, smoothing methods are able to handle trends and seasonality that often underpin observational data. We rely on the Holt-Winters model. Here, the trend and seasonality components for the forecast are computed separately and updated independently based on the error computed from the forecasted and observed values. The seasonality component in the Holt-Winters model may either be additive or multiplicative; we work with the multiplicative components that are are known to perform better with real-world applications.

3.6.1 Filter Queries

3.6.2 Correlation/statistical queries

3.6.3 Kernel Density estimation queries

3.6.4 Set Queries

- Membership (Bloom Filters)
- HyperLogLog: probabilistic estimator for approximating the cardinality of a dataset
- Count-Min: approximate frequencies of repeated elements within a dataset

3.6.5 Forecasting Queries

Queries that we have discussed so far report on what has happened; a forecasting query requests information about what will happen.

3.7 Support for High throughput Query Evaluations

3.8 Coping with Failures in Rivulet

4. PERFORMANCE EVALUATION

4.1 Datasets and Experimental Setup

4.2 Micro benchmarks

4.3 System Benchmarks

5. RELATED WORK

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a framework for constructing a distributed sketch over spatiotemporal observational streams. Rivulet maintains compact representation of the observational space, supports dynamic upscaling and downscaling to preserve responsiveness and avoid overprovisioning, and supports a rich set of queries to explore the observational space. Our methodology for achieving this is not restricted to the Neptune and is broadly applicable to other stream processing systems. Our empirical benchmarks, with real-world observational data, demonstrate the suitability of our approach.We now make a set of assertions pertaining to the research questions that we explored in this study.

RQ-1: Extracting metadata from observational data streams as they come in. We then check to see if the sketch needs to be updated. Each vertex represents a range of values; We achieve compactness because the number of vertices in the graph is influenced by the number of ranges and how they are organized both of which are dynamically managed in Rivulet. We also maintain summary statistics within these vertices to track the distribution/dispersion of feature values and the frequency with values fall within that range. Maintaining such summary statistics in an online fashion precludes the need to maintain fine-grained information that is memory intensive.

RQ-2: Data arrivals in observational settings are not uniform and the underlying infrastructure must be responsive to it. There will be variability in the rates and volumes of data arrivals from different geolocations due to differences and flux in the number of sensing sources. Rivulets scaling mechanism avoid overprovisioning via targeted scaling of portions of Rivulet. Targeted scaling allows us to alleviate situations where sketch updates cannot keep pace with data arrival rates. Memory pressure is also taken into account during upscaling, downscaling, and creation of spares. Accounting for memory pressure by tracking page faults, swap space, and available free memory allows us to cope with situations such as thrashing that may be induced by other collocated processes. Since Rivulet is memory-resident (to avoid disk I/O costs) tracking strains

induced either by nodes comprising the Rivulet distributed sketch or the collocated processes is critical.

Given the rates of data arrivals, reactive schemes for scaling would result in updates and query evaluations over portions of the feature space to be slow. Our proactive scaling scheme triggers dynamic upscaling and downscaling based on the expected data arrival rates and the accompanying memory footprints. Our distributed locking scheme avoid deadlocks and liveness issues during scaling operations.

RQ-3: Rivulet is spatially explicit, with nodes comprising distributed sketch responsible for managing particular geospatial scopes. Scaling decisions are localized to particular nodes and involve load shedding during upscaling and fusion during downscaling. Each Rivulet node also maintains a prefix tree that maintains compact information about the nodes managing geospatial scopes. At each node, the metadata graph maintains information about different chronological time ranges; the system maintains fine-grained information about recent data and coarser grained information about older, historical data. This structure supports incremental scaling, allows nodes to effectively assimilate observations at high rates, and redirect queries to nodes where they should be evaluated. Our graph based organization of the extracted metadata allows us to support a rich set of queries without compromising on timeliness; the accuracy of these evaluations is influenced by the granularity of the representations.

RQ-4: Since we support targeted upscaling and downscaling based on the data arrival rates and memory pressure, the assimilations and query evaluations can keep pace with the observations. During query evaluations, only those Rivulet nodes that hold portions of the observational space implicitly or explicitly targeted by the query are involved in the query evaluations, and that to independently and without synchronizing with each other: this allows us to support high throughput query evaluations.

Our future work will target support for Rivulet to be used as input for long-running computations that are expressed as MapReduce jobs. Such jobs that would execute periodically and on potentially varying number of machines could target the entire observational space or only the most recent ones.

Acknowledgments

This research has been supported by funding (HSHQDC-13-C-B0018 and D15PC00279) from the US Department of Homeland Security, the US National Science Foundation's Computer Systems Research Program (CNS-1253908), and the Environmental Defense Fund (0164-000000-10410-100).

7. REFERENCES

[1] T. Buddhika and S. Pallickara. Neptune: Real time stream processing for internet of things and sensing environments.

[2] G. Niemeyer. Geohash, 2008.