

# Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings

---

Volume 17 *Boston, USA*

Article 12

---


2017

## Optimizing Spatiotemporal Analysis Using Multidimensional Indexing with GeoWave

Richard Fecher  
*Digital Globe*

Michael A. Whitby  
*Digital Globe*

Follow this and additional works at: <https://scholarworks.umass.edu/foss4g>

 Part of the [Databases and Information Systems Commons](#), [Geographic Information Sciences Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

---

### Recommended Citation

Fecher, Richard and Whitby, Michael A. (2017) "Optimizing Spatiotemporal Analysis Using Multidimensional Indexing with GeoWave," *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*: Vol. 17 , Article 12.

DOI: <https://doi.org/10.7275/R5639MXD>

Available at: <https://scholarworks.umass.edu/foss4g/vol17/iss1/12>

This Paper is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings by an authorized editor of ScholarWorks@UMass Amherst. For more information, please contact [scholarworks@library.umass.edu](mailto:scholarworks@library.umass.edu).

# Optimizing Spatiotemporal Analysis using Multidimensional Indexing with GeoWave

Rich Fecher<sup>a,\*</sup>, Michael A. Whitby<sup>a</sup>

<sup>a</sup>*Digital Globe*

---

---

**Abstract:** The open source software GeoWave bridges the gap between geographic information systems and distributed computing. This is done by preserving locality of multidimensional data when indexing it into a single-dimensional key-value store, using Space Filling Curves (SFCs). This means that like values in each dimension are stored physically close together in the datastore. We demonstrate the efficiencies and benefits of the GeoWave indexing algorithm to store and query billions of spatiotemporal data points. We show how applying its multidimensional indexing strategies can reduce query and processing times by multiple orders of magnitude using publicly available taxi trip data published by the New York City Taxi Limousine Commission (NYCTLC). Furthermore, we demonstrate how this efficiency lends itself to analysis that would otherwise be unfeasible.

---

\*Corresponding author

Email address: [richard.fecher@digitalglobe.com](mailto:richard.fecher@digitalglobe.com) (Rich Fecher)

## 1. Introduction

With the constantly growing amount of data available for use in geospatial applications, the ability to scale those applications is becoming more and more important. Distributed computing systems provide a highly scalable solution, but they also store data in single dimensional key-value stores that are not natively able to preserve multidimensional locality. To preserve multidimensional locality in a distributed key-value store is to store similar values in each dimension using similar keys, resulting in values physically close together to the great benefit of range-based retrieval. The open-source software GeoWave (GeoWave 2017) uses SFCs to efficiently index data into a key-value store in a way that preserves locality. The configurable options and expandable capabilities of GeoWave allow a user to adapt it to best fit his or her specific use case. In section three, we will show the benefits of GeoWave's indexing approach in the context of a real-world multidimensional use case. We measure a reduction in query and analytic times by multiple orders of magnitude over a technique that does not use GeoWave's locality preservation. In section four, we will discuss an application that utilizes billions of historical taxi trips indexed into the GeoWave framework to provide interactive trip estimation to a user.

## 2. Design Description

One of the greatest advantages of GeoWave over other current spatial and temporal indexing methods is the ability to work with any number of dimensions. The primary method to accomplish this is a tiered, hierarchical, SFC storage system. GeoWave uses Compact Hilbert SFCs (Hamilton and Rau-Chaplin 2008) to represent n-dimensional grids at multiple tiers that it then uses to provide an index in which each elbow (discrete point) of an SFC maps to a cell of the grid (Whitby et al. 2017). To achieve varying levels of precision, each dimension is progressively decomposed in a manner similar to a quadtree. However, there is a significant difference between the GeoWave process and a standard quadtree as there is no direct link between progressive levels (or tiers) in the GeoWave decomposition. Each entry in the datastore has its corresponding tier stored in its key as part of its Row ID (Whitby et al. 2017).

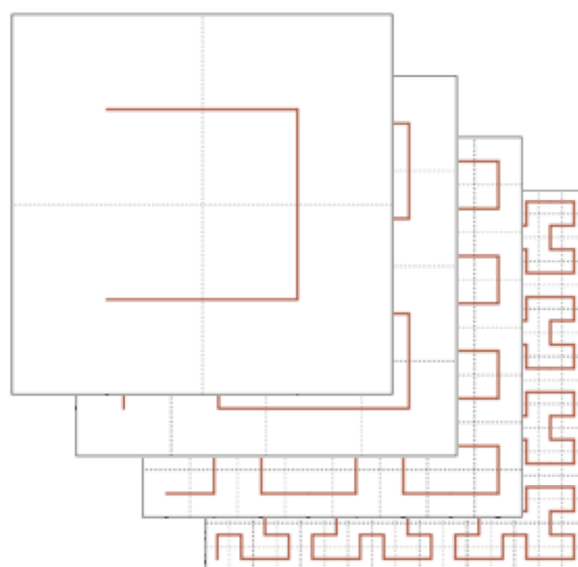


Figure 1: Visualization of the GeoWave tiered SFC storage system.

GeoWave also provides the ability to index multidimensional data into a key-value store using the XZ-Ordering algorithm proposed by Bohm (Bohm et al. 1999). The XZ-Ordering algorithm has the benefit of being insensitive to grid resolution. This ability, combined with the avoidance of object decomposition, makes it beneficial for use in indexing regular shaped line/polygon data.

A further challenge is that of unbounded dimensions like time. Unlike latitude and longitude, time does not have a standardized beginning or end, which would normally make it impossible to normalize in a way that would fit into an SFC. GeoWave solves this issue by defining a configurable periodicity for that dimension and creating bins for each period. Each bin is a hyperrectangle representing ranges of data and labeled by points on a Hilbert SFC (Whitby et al. 2017). A three-dimensional representation of this process is shown in Figure 2.

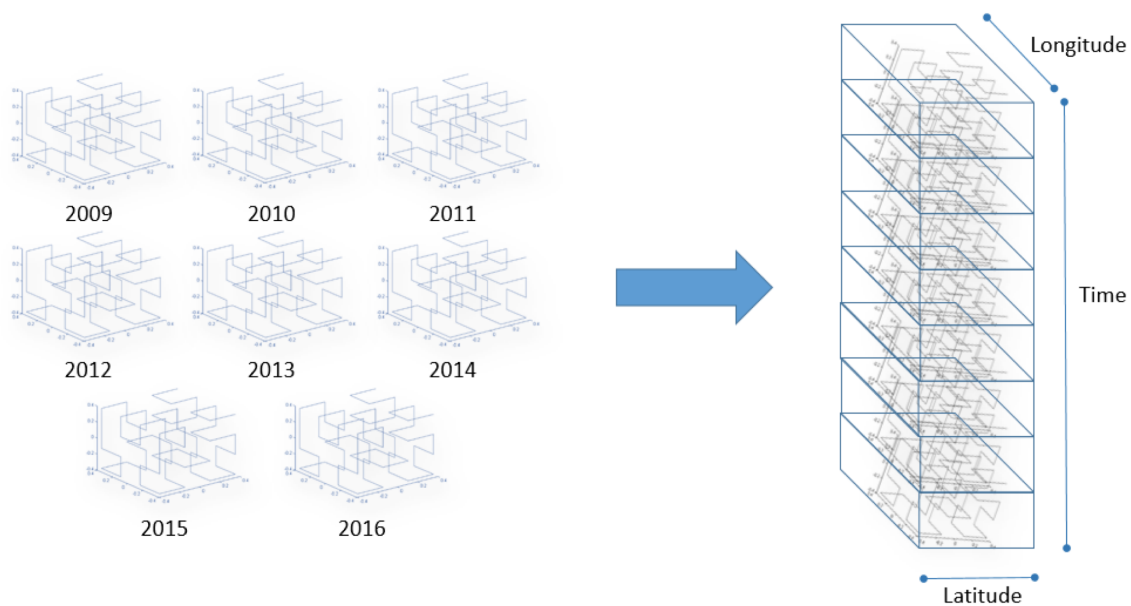


Figure 2: Composing the unbounded temporal dimension.

The GeoWave framework is implemented on top of a distributed key-value store like Apache HBase (Apache 2017b) or Apache Accumulo (Apache 2017a), however its user facing tools and developer facing APIs are agnostic to the underlying datastore (Whitby et al. 2017). This allows a user to take advantage of GeoWave’s indexing strategies and analytics on whichever key-value store he or she deems best suited for the job.

### 3. Evaluation

For the experiments done in this section, we are using data provided by the NYCTLC on over 1.3 billion taxi trips dating back to 2009 (TLC 2016). We used GeoWave to index this data into an Apache HBase (Apache 2017b) backend in multiple ways to demonstrate the advantages and disadvantages of the various methods, as well as the level of customization that GeoWave provides. We also measured the performance of using HBase in this scenario without leveraging GeoWave’s indexing approach. Table 1 provides the names of the indexing methods that will be referred to in this paper and shows which particular attributes, or dimensions, are indexed by each. The names of each method are intended to be partially descriptive in order to help differentiate between the methods in the results figures. The None method represents

a full table scan using a unique Feature ID we created for this dataset. This None method represents a naive way of storing multi-dimensional data within HBase without any efficiency gains provided by GeoWave. The table also designates whether the method used the traditional tiered technique or the XZ-Order technique. Time Range in this case is defined as the time range between the pickup time and the drop-off time. Table 1 shows indexing methods used for the NYCTLC data.

Name	Pickup Latitude	Pickup Longitude	Pickup Time	Drop-off Latitude	Drop-off Longitude	Drop-off Time	Time Range
6D	X	X	X	X	X	X	
5Dtiered	X	X		X	X		X
5Dxz	X	X		X	X		X
3Ddropoff				X	X	X	
3Dpickup	X	X	X				
2Ddropoff				X	X		
2Dpickup	X	X					
None							

Table 1: Indexing methods used for the NYCTLC data.

In this set of tests, we want to know how many taxi trips were taken to and from Yankee Stadium during the 2009 baseball season, the playoffs, the World Series (plus the final game), and the subsequent off-season. The speed at which these queries can be performed when dealing with a dataset of over 1.3 billion features is greatly dependent on how the data is indexed.

Running these queries on this amount of data would normally take a very large cluster. The raw data alone is over two terabytes in size due to our requirement for the experiment to index it in so many ways. We were able to utilize a feature of the Apache HBase ([Apache 2017b](#)) integration with Amazon Web Services' (AWS) Elastic Map Reduce (EMR) to store our GeoWave tables in the AWS S3 object storage system. Since we decoupled our compute requirements from our storage requirement, we were able to run all of the queries on an EMR cluster with one m3.xlarge master node and three m3.xlarge worker nodes. Our compute requirements are relatively light for the better index methods we run range scans with efficient, well-defined ranges from GeoWave. But to theoretically allocate enough HDFS disk space for all of these tables, we would have required more than an order of magnitude increase in cluster size.

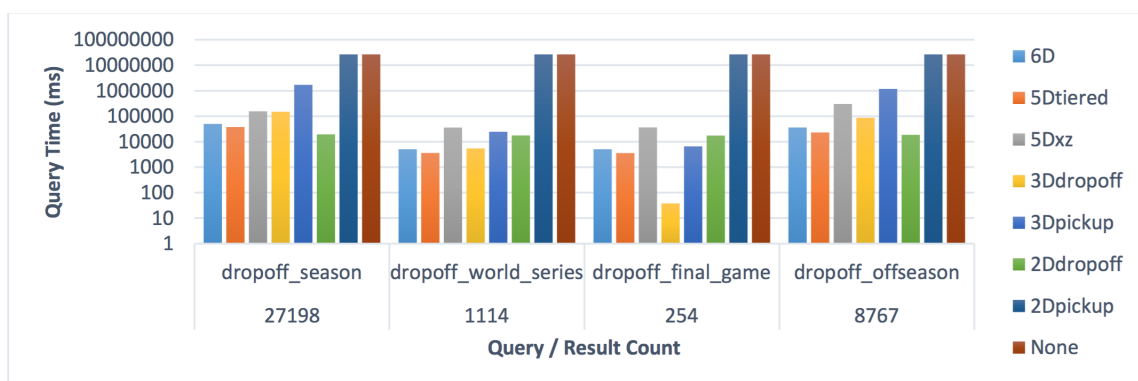


Figure 3: Results for taxi drop-off queries.

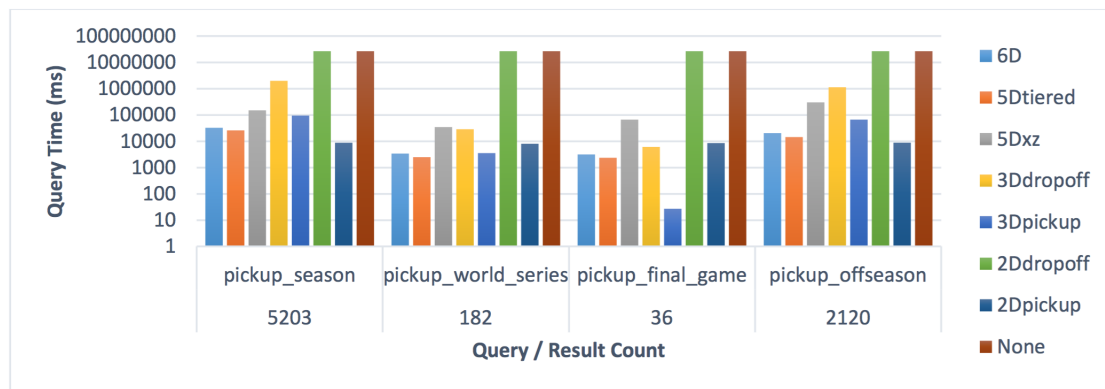


Figure 4: Results for taxi pickup queries.

As shown in Figures 3 and 4, a spatially and temporally constrained query has the best performance when the indexed attributes match the same latitude, longitude, and time as the query. This is why the 3Ddropoff index performs very well in the first query and why the 3Dpickup performs so well in the second query. However, the issue with both of these indexes is that their performance suffers dramatically when they are used for a query in which the spatial component of the index is not tightly constrained. In the case of the 3Ddropoff, the second query tightly constrains the pickup location but the drop-off location is completely unconstrained. This issue is even further exaggerated in the 2D indexes. The 3D indexes were indexed with respect to time, so even in the unconstrained spatial queries it still wasn't necessary to search through the entire data set. When using the 2Ddropoff indexed dataset for a query when the drop-off latitude and longitude are unconstrained, it forces the system to search through the entire dataset. This takes over seven and a half hours and defeats the purpose of indexing the data at all. This discrepancy between query times is visualized in Figure 6.

With a sufficiently sized dataset, it is neither efficient nor desirable to re-index the data for each individual query. The solution to this is to index the dataset with respect to a sufficient number of dimensions as to allow all of the expected query patterns to complete in a reasonable amount of time. We have done that with the 6D, 5Dtiered, and 5Dxz indexes. These indexes have similar performance between each of the eight queries (see Figure 5) with the 5Dtiered index performing best overall. The 5Dxz using the XZ-Ordering for its indexing method showed poorer performance as this is not the type of data for which it is optimal. We have found it performs best with more regular shaped extents that is data that has more similar lengths per dimension. In this case, the tiered indexing is the preferred method for point data such as this because we are able to decompose these queries to a very low level in the tiered structure, and only one of the dimensions, the time range, must be indexed as an extent. The 6D index was not optimal in this use case because our definition of the Time Range used in the 5Dtiered index was sufficient for the queries we ran here. If, within our queries, we needed to differentiate between taxi trips that were picked up within a certain time range and then dropped off within a separate time range the 6D index would be required.

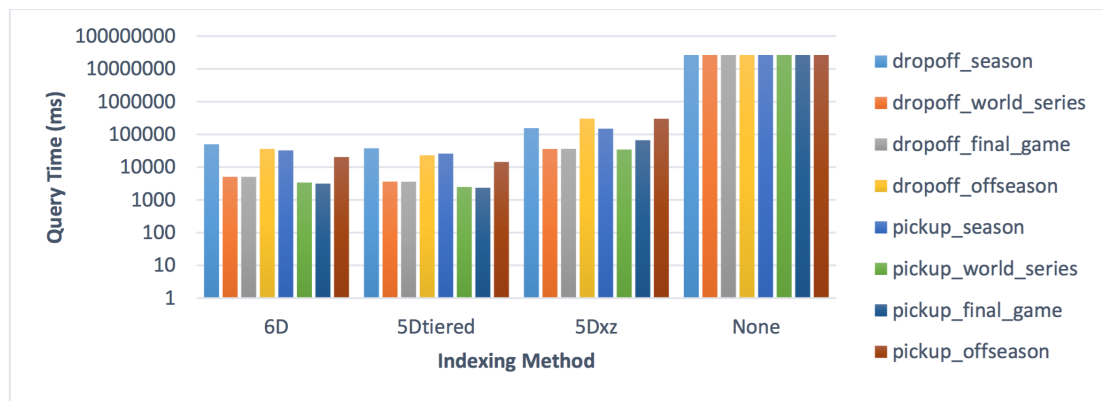


Figure 5: Results by indexing method.

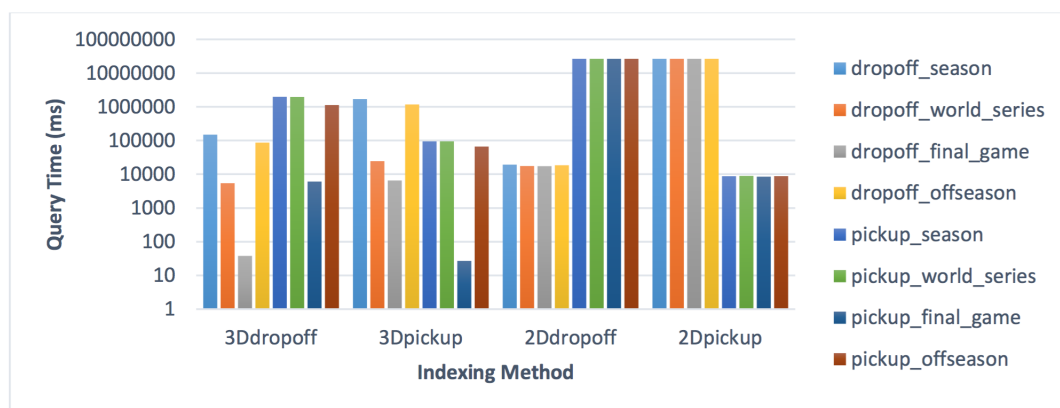


Figure 6: Results by indexing method (cont.).

In these basic data retrieval exercises, we show how a variety of questions can be asked of massive datasets at interactive rates given the appropriate indexing. Now we will extend this exercise of merely counting records to perform more advanced data analysis. We utilize the nature of distributed storage techniques lending themselves to large scale processing to run a parallelized, distributed Kernel Density Estimation (KDE) algorithm and ultimately produce heatmaps. We aggregate the data during each baseball season between 2009 and 2016 for any taxi trips that either started or finished at Yankee Stadium within this KDE process, producing weights for areas where passengers are either going to or coming from Yankee Stadium. We then run the KDE for the likewise trips during the offseason between the same years. These weights are scaled by a ratio of the number of total days during a baseball season to the total days of offseason and subtracted from the first set of weights. This step, including the offseason, normalized by the ratio of time periods, is intended to offset the theoretical regular taxi traffic during the season to and from Yankee Stadium that is completely unrelated to baseball games. This is based on an assumption that the traffic unrelated to games is fairly constant year-round. It is worth noting that most types of queries measured above are required to perform this analysis. Moreover, there are 34 distinct queries to account for both pick-ups and drop-offs at Yankee Stadium for each individual season and offseason time period. Therefore the 5Dtiered index was chosen as the data source to minimize worst case performance. We certainly wanted to avoid waiting several hours for any one query, and this index had proven flexible enough to retrieve the data for all of these queries within 30 seconds or much less. The final aggregate heatmap was added as an interactive layer on a webmap with screenshots shown in Figure 7.

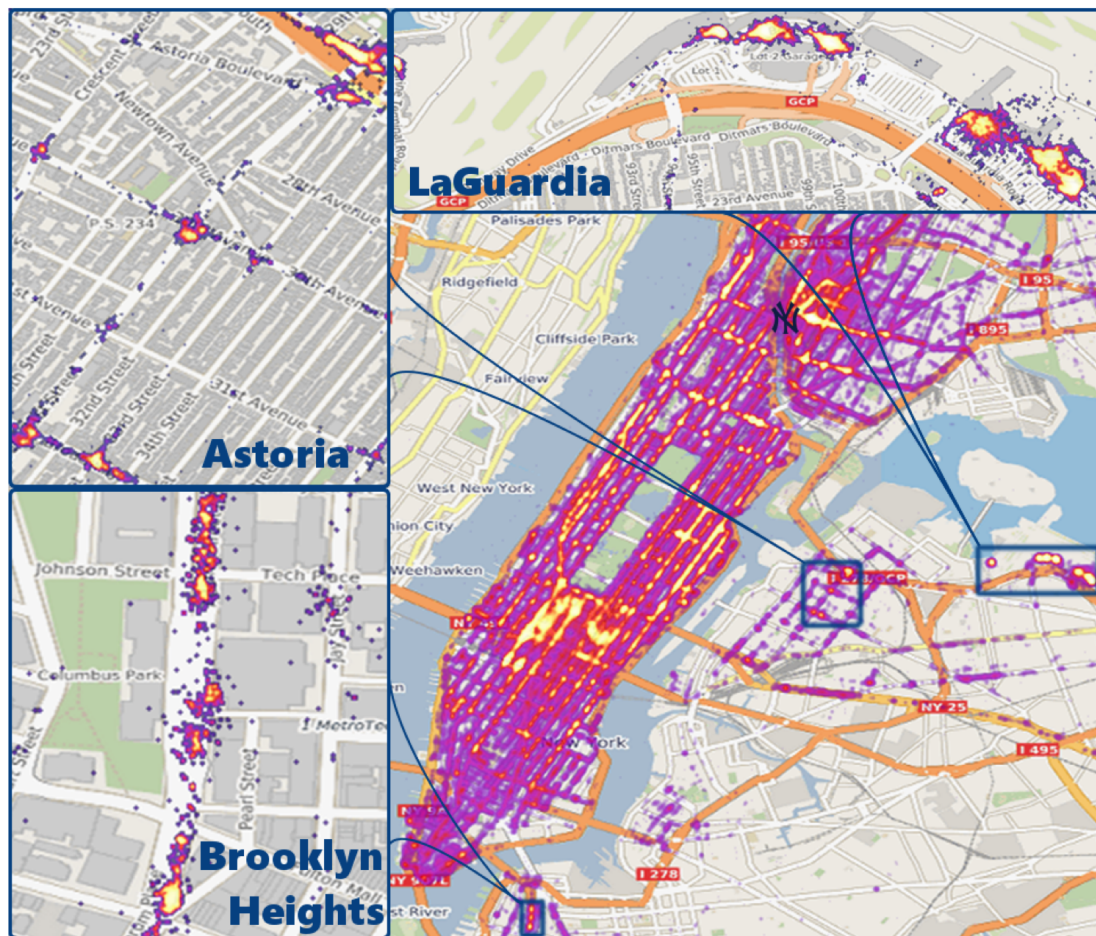


Figure 7: Overview heatmap and detailed area callouts Graphic background MapBox and OpenStreetMap.

Many of the trends seem intuitive, such as high amounts of taxi trips between Yankee Stadium and Manhattan, but some trends were not so apparent without taking a closer look.

There are hotspots of many taxi trips within Brooklyn Heights and Astoria, particularly near subway entrances. Without knowing New York City very well, this didn't make particular sense until we took a close look at the subway network itself. It is not possible to access the Yankee Stadium subway stop from these locations, particularly Astoria, without incurring transfers and a lengthy trip. The taxi ride is not far though, particularly between Astoria and Yankee Stadium. Clearly, baseball fans making a trip from the extremities of the subway system often find it easiest to get off of the subway without going into Manhattan, followed by leveraging taxi service for the final leg of the trip. Another result that initially caused confusion was significant activity at LaGuardia Airport. We did not realize baseball fans would go directly from the airport to the ballpark and back, but it is actually common. Many baseball fans fly in for a game, and use taxi services directly to and from LaGuardia Airport.

#### 4. Demonstration

We extended the GeoWave framework into a project called nyc-taxi with an interface that allows a user to select a start and end point on a map of New York City for a theoretical taxi trip. We then created a 20-node EMR cluster. The nyc-taxi project was installed onto



the cluster, and a GeoWave datastore with an Apache Accumulo (Apache 2017a) backend was created. The publicly available dataset with over 1.3 billion historical data points was ingested into the GeoWave datastore with locality preservation in the five following dimensions - time of day, pick-up latitude, pick-up longitude, drop-off latitude, and drop-off longitude (TLC 2016).

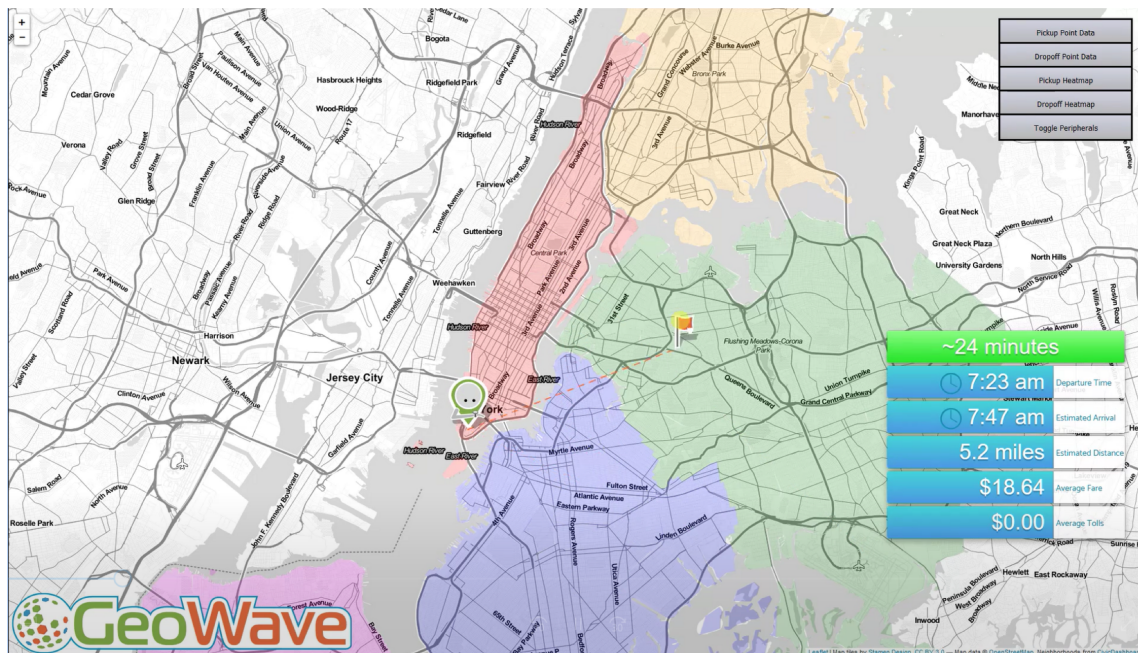


Figure 8: Screenshot of Taxi estimation demo.

The GeoWave indexing strategy allows the application to efficiently query the massive historical dataset for similar taxi trips based on user-selected start and end points, as well as the current time of day. The returned data can then be used to calculate useful information and display it for the user. The aggregation of these statistics is distributed within each data node so that it appears real-time to the user. Once the markers are placed on the interface map, the application displays the approximate trip length, average fair, estimated distance, etc. to the user interactively. As shown in this paper’s experimental results, without the multidimensional locality preservation provided by the GeoWave framework, this application would have to perform much larger scans on each request, causing a significant reduction in its performance.

## 5. Conclusion

Using the GeoWave framework, we were able to show the massive benefits of using multi-dimensional indexing to preserve locality in a single dimensional key-value store. Specifically, we showed the benefits of n-dimensional indexing in regards to spatiotemporal applications and discussed the methods that GeoWave uses to accomplish it. We also demonstrated an application that extends the GeoWave framework to interactively return an estimated trip length and cost to a user based on over 1.3 billion historical taxi trips. GeoWave is an open source project and can be found on GitHub at <https://github.com/locationtech/geowave>.

## References

- Apache, 2017a. Apache accumulo. Online, online; accessed July 21, 2017.  
URL <https://accumulo.apache.org>
- Apache, 2017b. Apache hbase. Online, online; accessed July 21, 2017.  
URL <https://hbase.apache.org>
- Bohm, C., Klump, G., Kriegel, H.-P., 1999. Xz-ordering: A space-filling curve for objects with spatial extension. Proc. 6th SSD, LNCS 75-90.
- GeoWave, 2017. Geowave. Online, online; accessed July 21, 2017.  
URL <https://github.com/locationtech/geowave>
- Hamilton, C., Rau-Chaplin, A., 2008. Compact hilbert indices: Space-filling curves for domains with unequal side lengths. Information Processing Letters 155163.
- TLC, 2016. Tlc trip record data. nyc taxi limousine commission. Online, online; accessed July 21, 2017.  
URL [http://www.nyc.gov/html/tlc/html/about/trip\\_record\\_data.shtml](http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml)
- Whitby, M. A., Fecher, R., Bennight, C., 2017. Geowave: Utilizing distributed key-value stores for multidimensional data. 15th International symposium on Spatial Temporal Databases.