Towards Interactive Analytics and Visualization on One Billion Tweets

Jianfeng Jia, Chen Li, Xi Zhang, Chen Li, Michael J. Carey
Department of Computer Science, University of California, Irvine
{jianfenj,chi24,xiz13,chenli,mjcarey}@ics.uci.edu

ABSTRACT

We present a system called "Cloudberry" that allows users to interactively query, analyze, and visualize large amounts of data with temporal, spatial, and textual dimensions. As a general-purpose full-stack solution, it has a friendly UI, intelligent middleware, and a powerful big data management backend running Apache AsterixDB to enable big data analytics and visualization. We will demonstrate the system using Twitter data on a computer cluster.

Keywords

Spatial-temporal-text data, data analytics, visualization, social networks, parallel database, Apache AsterixDB

1. INTRODUCTION

We are living in an era where a large amount of digital information is generated on a daily or even hourly basis through social networks, blogs, Web logs, online communities, news sources, and mobile applications. This massive amount of data holds valuable information that can be used for various analytical purposes, e.g., to track social opinions on different topics, trends of a certain event, etc. For instance, in this year's presidential election (2016), there are many ongoing efforts[1][2] on using social media such as Twitter to analyze the public political opinions on the candidates. These results can not only help us gain more insights about the election, but also help political candidates decide how to allocate campaign resources to maximize their winning chances. This type of analysis requires a significant amount of effort, and tends to be performed in an ad-hoc manner for certain requirements (e.g., comparing the public opinions on Hillary Clinton and Donald Trump), and thus is not general nor extendable to other applications.

We are developing a general-purpose system called "Cloudberry" to support interactive analytics and visualization on large amounts of data. We focus on data sets with temporal, spatial, and textual attributes, which are commonly available in domains such as social media and mobile phone application usage. The system has several unique capabilities: (1) *Scalability*: By using the Apache AsterixDB[3] big data management system, it can utilize a computer cluster to store, index, and query large amounts of information (e.g., billions of tweets). (2) *Interactivity*: By being able to answer an analytical query efficiently (e.g., in sub-seconds) using indexing and view-materialization techniques, it allows a user to analyze and explore data interactively. (3) *Visualization*: It provides a powerful and friendly interface for users to visualize information on a map. (4) *Currency*: By utilizing a unique data feed feature in Apache AsterixDB, it can ingest the current data "as of now" to allow users to query the latest information as well as historical data.

In this demo, we will show how Cloudberry can support interactive analysis and visualization on tweets. Figure 1 shows the interface. A user can first have an overview of the spatial, temporal, and hashtags distribution of all tweets. After identifying a hot area, time range, or hashtags of interest, the user can further drill down to the next level. The user can also type in keywords to focus on a certain event. Subsequently, the system lists all relevant tweets in a sidebar so that the user can review the finest details of the tweets. Our goal is to be able to enable this type of powerful interface on more than one billion tweets.

1.1 Related Work

The Cloudberry system follows the "overview first, zoom and filter, then details on demand" [5] design principle. Unlike a solely front-end solution such as the GAV Framework [4], or a desktop application like VIS-Stamp [3] that only allows users to explore the processed data loaded in a client's local machine, Cloudberry is connected to the big data management system Apache AsterixDB [1]. This allows the frontend to apply multiple predicates on different dimensions and to invoke multi-scale hierarchical aggregate functions on large data sets. Map-D[4] provides a similar analytical solution by relying on hardware GPU support. Compared to the centralized visualization solution Polaris [6], Cloudberry is a scalable solution running on multiple machines. In addition, our solution includes intelligent middleware to do view materialization and to answer queries using cached results in order to reduce query response times. There are also stream processing systems such as Tweet-

[1]http://electiontracker.us/
[2]http://react.brandwatch.com/uselection16/
[3]http://asterixdb.apache.org
[4]www.mapd.com/demos/tweetmap/

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

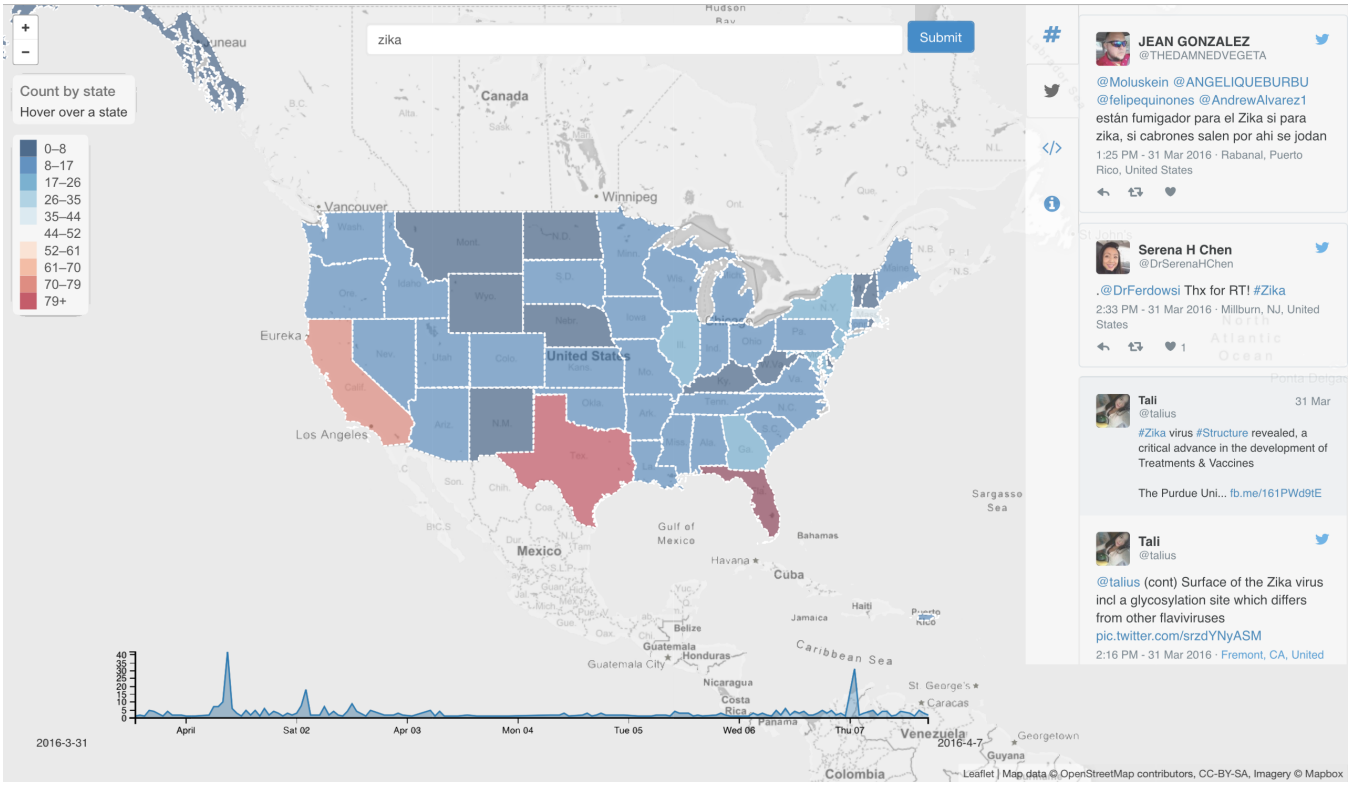

Figure 1: Cloudberry interface when searching "Zika" on tweets.

ping[5] that only watch the recent data. Cloudberry supports the retention of large-scale historical records as well, so it supports queries on both historical and real-time data.

2. SYSTEM ARCHITECTURE

Figure 2 shows the three-tier architecture of the Cloudberry system, a full-stack analytics and visualization solution. It includes a modularized web front-end interface, a back-end server to efficiently transform a front-end request to AsterixDB queries, and an AsterixDB cluster to continuously ingest, store, and query data in parallel. The three parts together enable an interactive, real-time visualization UI on a large-scale twitter dataset.

2.1 Front-end Interface

The front-end of Cloudberry is a single Web interface developed in HTML, Angular JS, and CSS. It displays the spatial and temporal distribution of tweets that contain user-specified keywords on both a map and a dashboard. The interface allows users to zoom into finer details on both spatial and temporal dimensions. Whenever the zoom-in level changes on either dimension, the front-end will issue a new HTTP request to the middleware server and render the results on the UI. The UI is continuously updated as new results are received from the back-end.

A front-end request to the middleware consists of filter conditions and aggregation conditions. A filter condition is a selection predicate that is a list of spatial areas (such as California, Texas, etc.), a time range (e.g., from 2016-05-01 to 2016-05-31), or keywords (e.g., "zika" or "virus"). An aggregation condition defines dimensions and measurements in the results. The dimension specifies which field and hierarchical level to group the results on. For example, we can group the tweets on the geolocation field by their city, county, or state. Also for the time dimension, we can group tweets per hour, per day, or per week. The measurement specifies the aggregation to apply in each group, which is an algebraic function, e.g., count, min, max, average, or a holistic top-k function. Optionally, a query can have an update interval to define a continuous query that runs periodically.

A front-end request is represented as a JSON record. An example JSON snippet is shown in Figure 3, in which a user wants to select all the tweets published from May 2016, located in California, Texas and Florida, that mentioned "zika", then aggregate those tweets per-day, per-state, and also return the top-50 hashtags. In addition, the user wants to get the latest results every minute.

2.2 Middleware Server

The Cloudberry middleware server is responsible for processing a front-end request and transforming it to efficient AsterixDB queries. In addition, it maintains materialized views for earlier queries to speed up query processing. Its main logic is implemented in the "Query Intelligence" (QI) and "View Manager" (VM) components of Figure 2. The QI first visits its cache to check if the predicates of the query match previously cached results. The cache is critical to optimizing a continuous query that is executed on the dataset many times periodically. In this way, Cloudberry can compute results on tweets incrementally without interacting with the AsterixDB server. For each front-end request (like the one shown in Figure 3), the middle-

[5]https://tweetping.net/

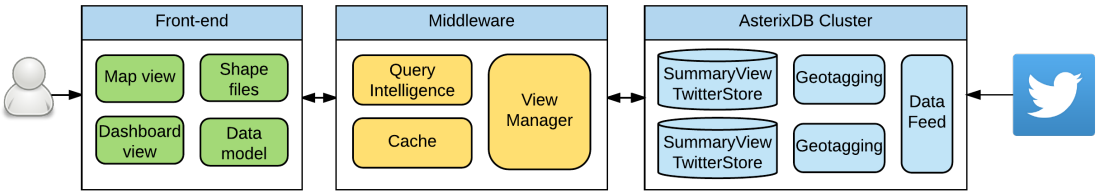

Figure 2: Cloudberry system architecture.

```
{
  "filters":{
    "time":{
      "start":"2016-05-01T00:00:00.000s"
    },
    "spatial":{
      "level":"state",
      "values":["CA", "TX", "FL"]
    },
    "text":["zika"]
  },
  "aggregate":{
    "time":{ "type":"count","level":"day"},
    "spatial":{ "type":"count","level":"state"},
    "hashtag":{ "type":"topK","k":50}
  }
  "period":{
    "value": 1, "unit": "minute"
  }
}
```

Figure 3: Front-end JSON request example

ware remembers the transformed AsterixDB Query Language (AQL) template, the last update time, and the latest cached results of the continuous query. For each time interval, the QI generates the corresponding database query by filling in the time predicate in the AQL template with the current time.

If the query cannot be answered by the cache, it will be forwarded to the View Manager, which is in charge of maintaining the information about the data sets and creating, updating, and deleting the materialized views on each dataset. A "view" is a dataset that stores the aggregated results of all predefined aggregation functions with certain spatial, temporal, and keyword predicates. Each view is identified by a single keyword and the spatial-temporal levels. For example, "view_zika" with summary level "county_day" stores the aggregation results of each (county, day) group of all the tweets mentioning "zika". The size of a view is significantly smaller than the original dataset. As a consequence, a query can be answered efficiently by using views, so that Cloudberry can provide an interactive user experience. The view manager maintains a meta-dataset to record the creation time, total visited times, latest accessed time, and the latest update time of each view. A new view is created if all the keyword predicates in a query cannot be found in the metadata. In that case, the first keyword in the query is selected as the identifier of the view. There is also one special empty-string-keyword view that preserves the by-county and hourly distribution of all the tweets. This view can help Cloudberry to answer the global overview aggregation request without scanning the entire dataset. Once a view has been created, the view manager will keep it updated automatically by appending aggregated results on newly ingested data. Inactive views will be removed periodically to save resources. For example, we can use a simple strategy to remove a view if it has not been used in the last 24 hours.

2.3 AsterixDB Cluster

Apache AsterixDB is used by Cloudberry to store a large number of records and many views and provide a high-speed query engine that can finish aggregation queries efficiently. It is a scalable big data management system that supports a variety of indexes such as B-tree, R-tree, and inverted index to support filtering operations without scanning entire datasets. In addition, it has a built-in parallel runtime query execution engine, Hyracks [2], to scale up to hundreds of machines so that an aggregation query can finish with a low latency. The view manager communicates with AsterixDB server via AQL queries. An example query is given in Figure 4, which shows how to get the per-state count by the given predicates on the tweet dataset.

```
use dataverse twitter
let $set := [14,23,42]  // id of CA,TX,FL
let $date_start := date("2016-05-01")
let $date_end := date("2016-06-01")
let $keyword := "zika"
for $t in dataset twitter.tweet
for $sid in $set
where $t.geotag.state_id = $sid
and $t."create_at">= $date_start
and $t."create_at" < $date_end
and similarity-jaccard(
    word-tokens($t."text"),
    word-tokens($keyword)) > 0.0
group by $c := $t.geotag.state_id with $t
return {
    "key": string($c) ,
    "count": count($t)
}
```

Figure 4: AsterixDB AQL example

In addition to query processing, Apache AsterixDB provides a "data feed" feature that accepts continuous data into the database. Inside a feed, one can apply various UDF functions to filter, project, and transform records on the fly and in parallel. The Cloudberry system uses AsterixDB's built-in socket feed adapter to get streaming tweets while applying a geo-tagging function to annotate the named locations of each tweet. In this way, we can query the newest tweets in real-time.

3. DEMONSTRATION SCENARIOS

In the demo, we will set up a cluster of six machines to demonstrate the usability of the full-stack Cloudberry system. Each machine has four cores, 16 GB memory, and a 1TB hard disk, and each runs an Apache AsterixDB node and a middleware server. The data set is collected using the Twitter Streaming API for the North American area. We have already collected data for about seven months, yielding about 720 million tweet records, and are still collecting new data at a rate of about 40 tweets per second. We will have about 1 billion tweet records to be used by the demo.

3.1 Initial Map View

Figure 1 shows the main Web interface, which consists of a U.S. geographic map, a time-serial chart, and a sidebar with rich context information. The user types keywords into the search box, and they are used as a filter condition in the front-end JSON request sent to the middleware. After the response comes back, the map displays the geographic tweet-count distribution for different states. The time-serial chart shows the number of tweets per day, and the sidebar shows the top-50 frequent hashtags in the tweets and also the details of the latest sample tweets. For instance, Figure 1 shows how "Zika" is discussed in different states at different hours. The color of each region illustrates the "hotness" of the topic in the region. In the figure, we can see that "zika" is discussed more frequently in Florida than in other states. The user can click on the sidebar to see the details of each tweet mentioning the keyword.

3.2 Zooming In

Cloudberry allows users to explore multi-granularity data from the highest level to a finer level. Figure 5 shows a user zooming into the Florida area. The map section automatically switches to the county-level distribution. We notice that there is a hotspot in Martin county. At the same time, the time-serial chart is updated to show the corresponding patterns of tweets published in the zoomed-in area. The user can further select an interesting time period to add another predicate on the time dimension to focus on tweets published within that period. By using this interactive exploration UI, the user can easily understand and explore the data.

Under the hood of the Web interface, every front-end action triggers an HTTP request to the middleware. Thus, the response time is a critical factor to achieving an interactive user experience. Even if the data does not fit into memory, with a carefully designed view manager and built-in indexes in Apache AsterixDB, we can avoid unnecessary disk scans and make the front-end UI responsive.

3.3 Current Data

Cloudberry allows the backend AsterixDB cluster to ingest tweets continuously in real-time, so it is important to keep the frontend results updated for the newly ingested data. To this end, Cloudberry has a customizable timer in the middleware which is used to retrieve the latest results from the AsterixDB server and continuously send it to the front-end UI. The middleware remembers the previous response and submits the corresponding AQL query periodically to the AsterixDB server with a new time range predicate so that AsterixDB only needs to access the most recent data. On the front-end side, by implementing the

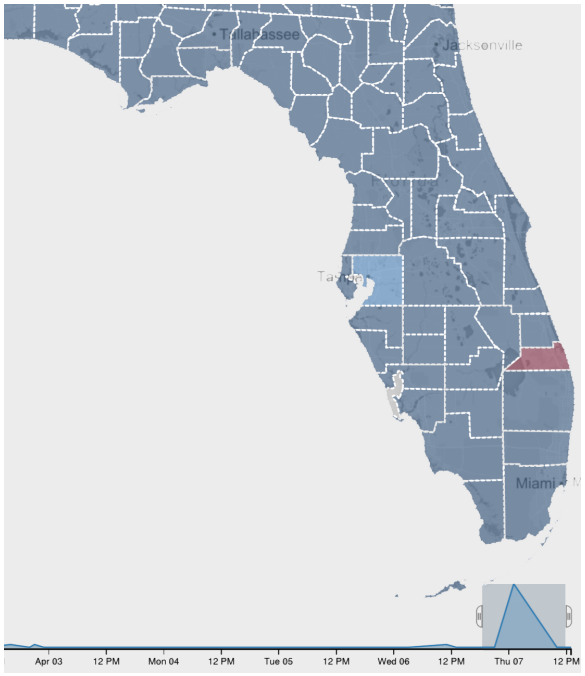

Figure 5: Zoom into county-level details in Florida.

"watch" interface in the Angular JS framework, Cloudberry can simply update the UI using newly received results without resending the request. This simplifies the front-end design.

4. REFERENCES

[1] S. Alsubaiee, Y. Altowim, H. Altwaijry, A. Behm, V. R. Borkar, Y. Bu, M. J. Carey, I. Cetindil, M. Cheelangi, K. Faraaz, E. Gabrielova, R. Grover, Z. Heilbron, Y. Kim, C. Li, G. Li, J. M. Ok, N. Onose, P. Pirzadeh, V. J. Tsotras, R. Vernica, J. Wen, and T. Westmann. AsterixDB: A scalable, open source BDMS. *PVLDB*, 7(14):1905–1916, 2014.

[2] V. R. Borkar, M. J. Carey, R. Grover, N. Onose, and R. Vernica. Hyracks: A flexible and extensible foundation for data-intensive computing. In *International Conference on Data Engineering*, pages 1151–1162, 2011.

[3] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, 2006.

[4] Q. Ho, P. Lundblad, and M. Jern. Geovisual analytics framework integrated with storytelling applied to HTML5. In *Proceedings of 16th AGILE Conference on Geographic Information Science*, 2013.

[5] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages, Boulder*, pages 336–343, 1996.

[6] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.