Identification & Analysis of popular trends around the world

CSC 591 Data Intensive Computing
Team 8 (DireWolves)

Aditya Bhardwaj abhardw2
Gautam Verma gverma
Samir Jha sjha4

December 6, 2017

Abstract

Modern technology and social media have made it possible for the voice of a single person to be heard by millions of people worldwide. They can share their opinions, collaborate on ideas, organize events and activities with other people, amongst others. This has affected our personality in diverse ways as people treat different users: strangers and friends in the same way. In this study, our aim is to analyze the real-time events and conversations of people active on three social websites: Twitter, Reddit, and Meetup to find where events related to popular topics are currently happening. To achieve this, we used AWS services viz. Kinesis, DynamoDB, lambda and built a custom lambda^2 architecture. This tool can be used to identify active social gatherings and help organize targeted activities with high audience participation. Additionally, it offers insights of the data with respect to different time frames such as hour, day, or month.

1 INTRODUCTION

Social media is one of the many greatest achievements that modern technology has endowed us with [1]. Different users today use diverse types of applications like Twitter, Reddit, Facebook, Wikipedia, Meetups, Instagram, etc. for various kinds of activities [3]. More importantly, its been used by large companies to gain business value [6], teaching industry to promote higher education [7], building relationships [8], non-profit organizations [9], etc. The number of users using social media applications are still increasing rapidly by the day [10]. This in turn has influenced the behavior of people across various domains like making decisions ,organizing events, marketing, etc.

In this study, we use real-time data from Twitter and Reddit to find out what people are talking about in current time. We use this information and combine it with data from Meetup to find out where popular events are occurring in real time according to time and geography on a world-level scale. The end product of this project is a real-time data crunching SaaS

application built using JAVA distributed framework AKKA and Ruby on Rails. The web-service is built using Play and dashboard application are deployed on Heroku. Anyone can use this application to see the live active popular social events and buzz words on social media.

At the heart of our model, we employ distributed computing using Amazon services like Kinesis, DynamoDB, lambda functions and create a system that works like a lambda architecture, which we call lambda-square. We use Kinesis to stream the data from these social media websites, and virtual machines that read data from these streams, extract information, and load into the DynamoDB. We perform transform operation on this data using lambda functions. There are two types of lambdas for this whichh differ on when they run. One of them runs per minute and the other per hourly which store the processed data in mutable and immutable form respectively. We call the latter as historical data. Together, these two make the speed and batch layer respectively. Lambda functions are also used to handle user queries, which interact with the database to get results and thus acts as the service layer. In this way, we are using anonymous random machines on AWS to provide the features similar to lambda architecture and call it lambda^2.

In the following sections, we discuss our methodology in detail, justify our design decisions, show results, and conclude that our system can efficiently query real-time data on time-frame, geography, or both, and be used to organize targeted activities with high audience participation.

2 METHOD

2.1 SYSTEM ARCHITECTURE

The system contains various components working in conjunction as shown in Figure 1. Each component is explained in detail in the following subsections.

2.1.1 Producers

The system pipeline starts with the Python scripts that are continously running on VCL machines. They fetch the data from Twitter, Reddit and Meetups APIs, tag it with their respective identifier, and put it into the Kinesis stream in a round robin fashion in order to balance the load. Since they are continously producing new data, we decided to run them on a dedicated machine. The Kinesis stream is responsible to handle inflow of high velocity and volume data. We have reserved 6 shards in the Kinesis stream to manage the high inflow of data. The reading capacity of each shard is 2 MB/s and writing capacity is 1 MB/s as per the Kinesis documentation. Thus, the maximum rate of data inflow and outflow to and from the stream is 360 MB/minute and 720 MB/minute respectively.

2.1.2 Consumers

The consumers read the data from kinesis stream, clean it, extract valuable information [2.1.3], and load it into the DynamoDB. We call this data as raw since the keywords do not has a score associated with them at this time. Also, it is mutable in this state since the

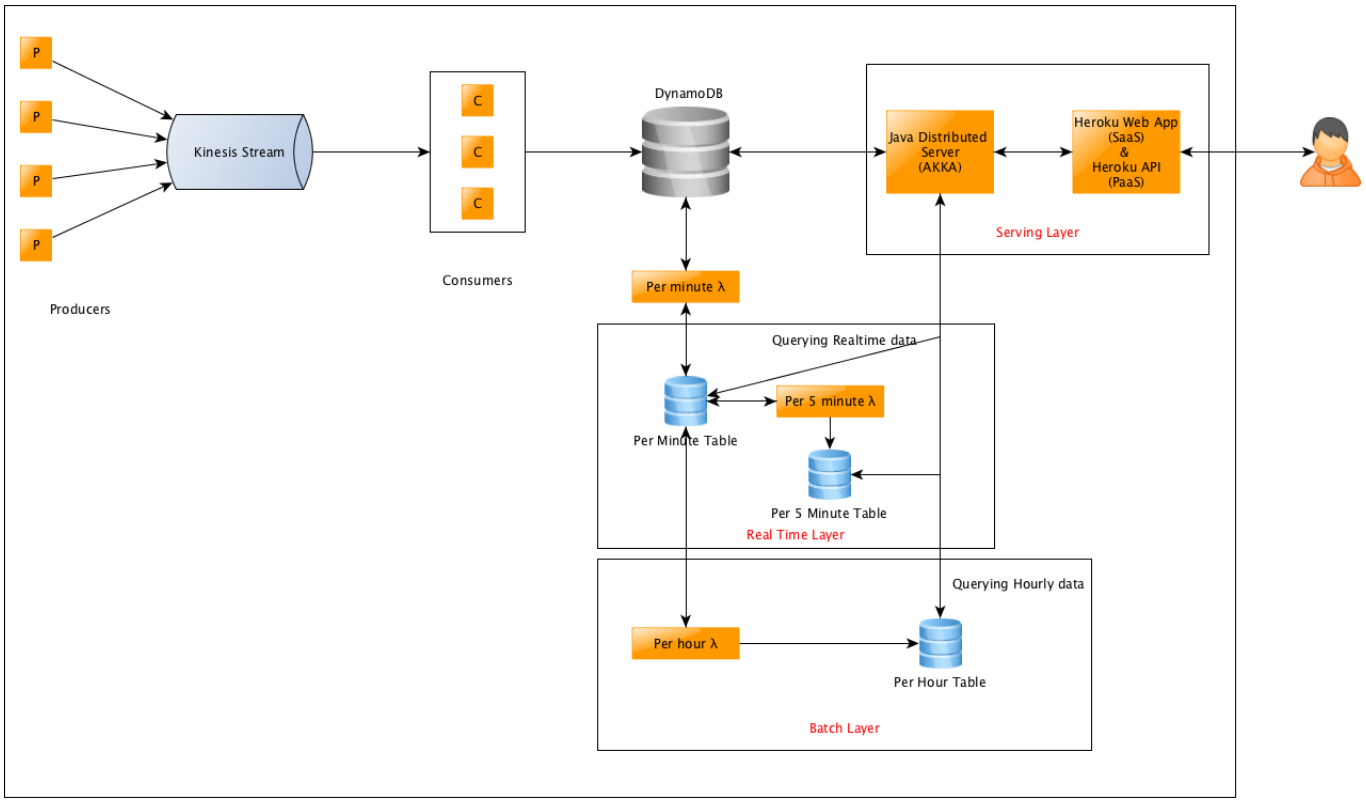


Figure 1: High Level System Architecture View of the system.

consumers updates the attrbutes if a key already exists (schema in tables 1 and 2). The consumers are also hosted on dedicated machines, VCL, at NCSU servers since they never stop consuming the data. There are 2 main consumers of the stream and each is responsible for 3 shards.

We extract the keywords from all the three source. We use 'Text' of Twitter data, 'Title' of Reddit data, and 'Name' and 'Description' of Meetup for this purpose. The consumers have a pre-installed python library - 'textblob' for filtering the noun phrases which make the keywords. They also record the time at which this operation occur. Keyword and timestamp together form the composite key in all the tables.

2.1.3 DynamoDb

The data from the consumers layer is fed into three tables in DynamoDB: liveReddit [1], liveTwitter [2], and meetups [3]. The Twitter and Reddit tables are live tables in that they store raw data loaded by the consumers. Also, they are mutable because if a key already exists, then the associated attributes are updated using addition (they are all integers. see schema in tables 1 and 2). The records in the Meetups table are only appended, thus it is not mutable in the same sense as live tables. The timestamp in live Reddit and Twitter is in the format "YYYY-mm-dd-HH-MM" to allow per minute updates whereas Meetups table use "YYYY-mm-dd-HH-MM-SS" to prevent overwrite on existing data. The tables

use keyword as the partition key for better scaling and timestamps as sort key to improve the query processing time. We have 3 more tables in addition to these, which are discussed in the next section.

Table 1: Reddit live Table Schema.

Attribute Name	Attribute Type
key_word	String (PK)
key_datetime	String (SK)
num_comment	Integer
score	Integer

*PK=Primary Key *SK = Sort Key

Table 2: Twitter live Table Schema.

Attribute Name	Attribute Type
key_word	String (PK)
key_datetime	String (SK)
favorites	Integer
replies	Integer
retweets	Integer

*PK=Primary Key *SK = Sort Key

Table 3: Meetups Table Schema.

Attribute Name	Attribute Type
key_word	String (PK)
key_datetime	String (SK)
key_category	String
key_location_city	String
key_location_country	String
key_location_state	String
name	String
venue	String

*PK=Primary Key *SK = Sort Key

2.1.4 Lambda Functions

The data from the live tables is processed per minute by a Lambda function hosted on AWS. It reads all the attributes of given data source per keyword and calculates a score for it. We use these scores to filter our top keywords. The formula used for this evaluation is given by equations 1 and 2. The constants in these equations were selected on the observation that features that users tend to less often should have more weight.

$$Score_{Reddit} = 2 * comments + score \tag{1}$$

$$Score_{Reddit} = 2 * replies + 4 * retweets + favorites \tag{2}$$

The advantage of using AWS lambda in this scenario is to minimize the cost of maintaining distributed system while providing better reliability, since we reduce the probability of

our application failure in case a system crashes on which the lambda was running. The second lambda functions that run per 5 minutes aggregates the data produced by the per minute lambda, extracts top K keywords for each category, and stores it into the DynamoDB. Thus, it provides the top results for current hour almost instantly. The hourly lambda function has a similar role in that it aggregates the records generated by the per minute lambda, extracts top K keywords, and creates a historical view. Since we almost always have pre-processed information in this sense, the time taken to resolve a user's query is significantly reduced. In essence, the lambdas running on minute basis form the speed layer layer while the hourly lambda forms the batch layer in our lambda^2 architecture. The lambdas put their output into three tables: minuteTable (Table 4), 5minuteTable (Table 5) and hourlyTable (Table 5) respectively.

Table 4: Per Minute Table Schema

Attribute Name	**Attribute Type**
key_word	String (PK)
key_datetime	String (SK)
score	Integer

*PK=Primary Key *SK = Sort Key

Table 5: Per Hour and 5 minute Table Schema.

Attribute Name	**Attribute Type**
key_word	String (PK)
key_datetime	String (SK)
rank	Integer
score	Integer

*PK=Primary Key *SK = Sort Key

2.1.5 Application Server

The application server is built on Play framework using Akka framework's Actor model [**?**] There are many advantages by using this approach in our system:

1. **Asynchronous:** The Actor model is completely asynchronous and works using message passing. In our system, we have different messages for the different API calls made to the DynamoDB.For aggregating results, we need to issue calls to different tables which are done asynchronously.

2. **Reliability** Akka's distributed framework provides up-time reliability. Actors can be started, stopped or restarted via simple message passing. A non-responsive Actor can hence be easily identified and reinitialized.

3. **Seamless integration with Heroku** Heroku provides the platform support for Play Framework. The dyno-worker model of Heroku fits well with the Akka Actor model. For our purpose, this helps us to concurrently query the different DynamoDB tables.

The application can be used as both PaaS and SaaS. As a PaaS, we provide a RESTful API endpoint through which a user can get popular events, and keywords in a particular timeframe or location. As a SaaS, we have built a web application that consumes the API endpoint services and provides a User Interface with custom built charts, plots and graphs to analyze our results.

2.2 Lambda^2 Architecture

The lambda^2 can be defined as an extension of the lambda architecture that runs in the cloud. The lambdas running on minute basis act as the speed layer whereas the per hour lambdas act as the batch layer. The former reads data from the live tables which is equivalent to 'All Data' in lambda architecture. It also purges this data after processing and thus reduces the storage required. A key difference in our lambda^2 architecture is that the batch layer also works on the data produced by speed layer. The hourly lambdas can crunch the data produced by the lambdas running per minute by removing unimportant data. The level of crunching is decided by the user and is inversely proportional to storage required. The downside of high amount of crunching is that it may reduce the accuracy of the system when large blocks of historical data are analyzed. But on the other hand, it decreases the latency of queries. Thus, there is a trade-off between time and space. Our Akka server is analogous to the serving layer of the lambda architecture. It receives queries from the user, fetches and merges the corresponding data from the batch and speed views. The lambda^2 provides the following added advantages:

1. **Better reliability**

   Lamda functions can be considered as small compute engines running anonymously in the cloud. They are automatically spawned on a machine at runtime, which need not be the same. Thus, the probability to find a healthy instance increases and risk of failure decreases.

2. **Lower cost**

   Since we only pay for the time we use resources on AWS lambda, we can cut the costs of reserving dedicated machines.

3. **Automated Scaling**

   AWS lambda provides the functionality to scale up the resources if needed by the system.

3 DISCUSSION

3.1 Replication of Incoming data

One of the major challenge in the project was to produce high amount of data being produced in order to test the load on the Kinesis stream. The additional limitations over producing huge data in real-time are that Twitter streaming API have limit per hour and per day. To overcome this limitation and produce more data for testing the streaming, the decision to replicate the data coming from all three sources: Twitter, Reddit, and Twitter by a factor of 1000. The maximum incoming stream fed to the Stream is recorded as 320 MB/min which is near the usage limit of 6 shards assigned in this application.

3.2 Design

3.2.1 How producers put data into Streams?

The system architecture mentions the use of Round-Robin approach to feed data into 6 shards which we find as the most efficient method. The following design choices were considered for implementing this:

1. **Linearly**

   In this approach, data would be loaded in a shard until it exhausts its maximum capacity. This is only feasible in the scenario when we do not know how many shards there are and there is no mechanism to manipulate them. However, this is not the case with AWS Kinesis. There are significant major disadvantages in this approach. Firstly, it is not parallel at all. Secondly, related data will form a big lump and flow through a single shard. Thus, the consumers will access older data for much longer period of time reducing concurrency.

2. **Randomly**

   Randomly choosing the shard to send the data can be a solution when large number of shards are being utilized. However, it is still unpredictable which shard may be consumed and it may suffer from the same disadvantages as Linear in worst case.

3. **Round Robin**

   In this approach, the data is uniformly distributed among the shards which results in their optimum utilization. Another significant advantage this design provides is that consider a scenario in which all the shards are being utilized at 40% of their capacity for a long time. In such case, we can reduce the total number of shards and reduce the overall cost.

3.2.2 Table Schema

The table schema for the project had a big impact on the overall working of the system and creating an optimal architecture. We formulated many such schemas and found the following two approaches to be most feasible:

1. **Multi-Tabular approach**

   Create new historical table for all the live tables per hour. There is no movement of data any time in the DynamoDB which reduces computational intensity of per hour lambda to some amount. However, it would increase the metadata and is very bad design strategy as it would linearly increases the number of tables in the DynamoDB.

2. **Cruncher approach**

   Create only two historical tables in addition to the live tables for the stream. These historical tables contains analysis per minute and per hour. Every hour, crunch the data into the hourly table from the minutes table. This is the approached adopted and already discussed.

4 RELATED WORK

4.1 Twitter Analytics for Business

Twitter provides analytics and insights dashboard for business [9], providing an interface to understand the dynamics of public information about the business. The product offered by Twitter focuses towards businesses which want to take decisions, add ad campaigns etc. on basis of real-time analytics. The dashboards included in the product are: Tweet Activity dashboard, Audience Insights dashboard and Campaign Dashboard. The application is similar to the project in a way that it provides real-time analytics and serves the specific purpose to support better decision making. Although, it is not known what technologies have been used for the tool, but Cassandra, Hadoop, Lucene, Pig are widely used by Twitter.

4.2 LinkedIn Job Listing

LinkedIn offers number of products such as people you may know, jobs you may be interested in, skills endorsement, and news feed updates. These products implements batch processing using Hadoop for providing recommendations to the users. The approach seems fit for such products as the website is targeted for specific user base, which means lesser real time traffic. Also, Linkedin pre-computes the data for People You May Know product [10] by recording close to 120 billion relationships per day in a Hadoop MapReduce pipeline, that runs 82 Hadoop jobs which require 16TB of intermediate data. The technologies used by LinkedIn are: Hadoop, Pig, Hive, Azkaban (Workflow), Avro Data, Zookeeper, Data In- Apache Kafka, Data Out- Apache Kafka and Voldemort.

5 RESULTS

We used 6 shards in the Kinesis to stream real time data. The producers (Twitter, Reddit, and Meetup) put the data into these streams in a round-robin fashion. This approach allowed the all the consumers to read related data concurrently increasing efficiency of the system. The producers write at a speed of around 300-320 Mb per minute as shown in Figure 2. The time of execution of per minute lambda function is between 0.15 to 2.6 seconds with an average of 0.9 second as shown in Figure 3a. It takes approximately 0.3-1.1 seconds for lambda function running per 5 minutes with an average of 0.8 second as shown in Figure 3b. For per hourly lambda function, the time of execution increase further and is about 20 seconds as shown in Figure 3c. Thus, all the lambdas easily finish executing before they are fired again. The response time for user queries is shown in Figure 3d with respect to different time intervals and finding Meetups corresponding to the resultant data from query.

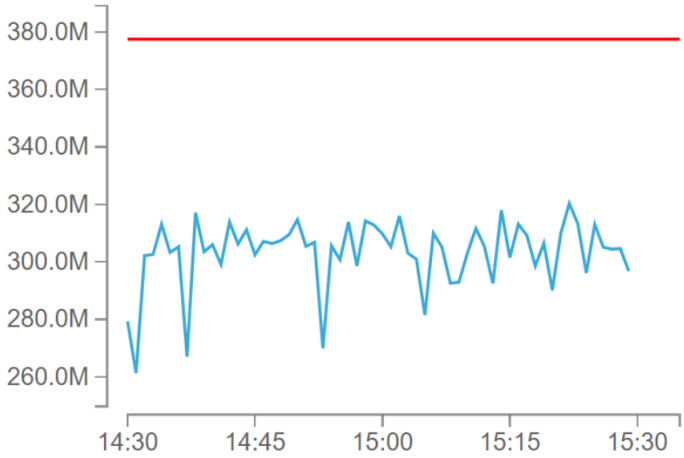
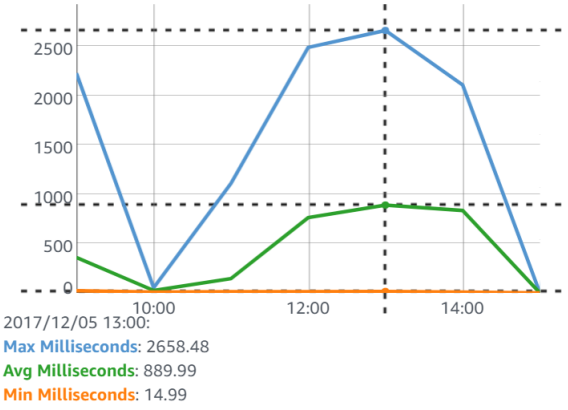
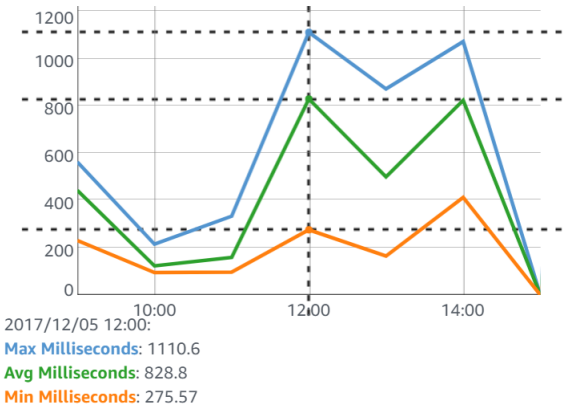


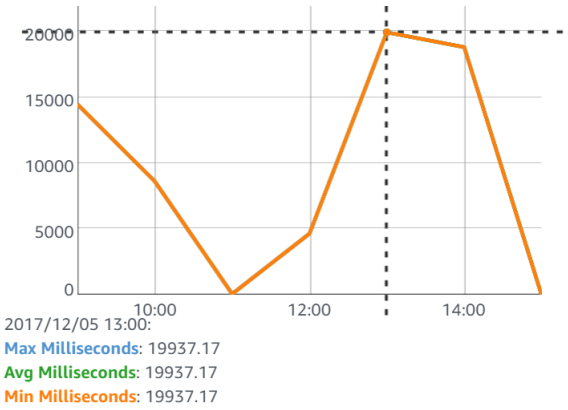
Figure 2: Shows the rate of incoming data in the Kinesis stream. Current time is represented on the x-axis whereas y-axis denotes the size of incoming data in Mega Bytes.



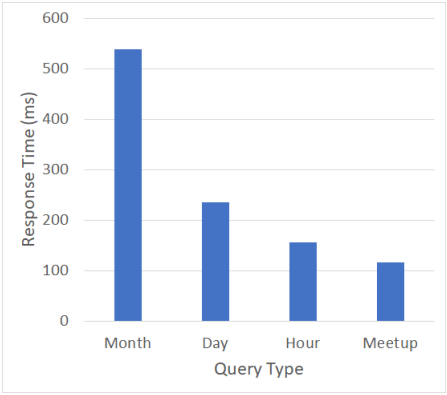
(a)



(b)



(c)



(d)

Figure 3: Figure (a), (b), and (c) show the time taken by the per minute, per 5 minute, and per hour lambda function to execute respectively. Current time is represented on the x-axis whereas y-axis denotes the time taken in milliseconds by the lambda function to execute. Figure (d) shows the response time for getting top keywords corresponding to queries associated with different time intervals and Meetups associated with these keywords.

6 CONCLUSIONS

We were successfully able to deploy Amazon Kinesis, DynamoDB, AWS lambda functions, and a custom-built lambda architecture to stream real-time data from Reddit, Twitter, and Meetup and perform analysis on it. Our analysis includes ranking data based on active communication of people on Reddit and Twitter, and finding where events related to these popular discussions are happening using Meetup. The system provides user the functionality to search in a particular time-frame, geography, or both. It can be used to organize targeted activities with high audience participation. Our data representation also allows to extend the project in multiple dimensions like finding popular venues according to the type of event, using machine learning to find patterns in the events and predict future gatherings, etc. Additionally, the architecture and design have the capacity to add more sources of data and use the same business logic to improve accuracy, and perform deeper analysis.

ACKNOWLEDGMENT

We would like to acknowledge the valuable inputs provided by Professor Vincent W. Freeh and TA Liang Dong in relation to the project.

References

[1] Mangold, W. G., & Faulds, D. J. (2009). Social media: The new hybrid element of the promotion mix. Business horizons,52(4), 357-365.

[2] Meyrowitz, J. (1986). No sense of place: The impact of electronic media on social behavior. Oxford University Press.

[3] Culnan, M. J., McHugh, P. J., & Zubillaga, J. I. (2010). How large US companies can use Twitter and other social media to gain business value. MIS Quarterly Executive, 9(4).

[4] Moran, M., Seaman, J., & Tinti-Kane, H. (2011). Teaching, Learning, and Sharing: How Today's Higher Education Faculty Use Social Media. Babson Survey Research Group.

[5] Briones, R. L., Kuch, B., Liu, B. F., & Jin, Y. (2011). Keeping up with the digital age: How the American Red Cross uses social media to build relationships. Public relations review, 37(1), 37-43.

[6] Lovejoy, K., & Saxton, G. D. (2012). Information, community, and action: How non-profit organizations use social media. Journal of ComputerMediated Communication, 17(3), 337-353.Gilbert, E., & Karahalios, K. (2009, April). Predicting tie strength with social media. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 211-220). ACM.

[7] Perrin, A. (2015). Social media usage. Pew Research Center.

[8] Junco, R., Heiberger, G., & Loken, E. (2011). The effect of Twitter on college student engagement and grades. Journal of computer assisted learning, 27(2), 119-132.

[9] https://business.twitter.com/en/analytics.html

[10] https://www.dezyre.com/article/how-linkedin-uses-hadoop-to-leverage-big-data-analytics/229