Weather Forecast Evaluator (WFE)

Presented on August 17, 2016 in fulfillment of the final requirements of W205.

Instructor: Arash Nourian
Team Members: XT Nguyen, Chris McPherson, Rama Thamman, Chuck Bolin
Repository: https://github.com/ChrisMcPherson/mids-205-weather-assistant

Table of Contents

Background

Each day around the world people seek out the current weather conditions. The information is generally used to plan out the next hours of the day. For many people the current conditions are enough. For others, more data is required. These consumers look into the future for weather forecast information. The information is predicted for much later in the day, tomorrow, or several days out. This data is useful for farmers planning when to sow or to harvest, fishermen for venturing far from shore, contractors for planning the pouring of concrete or scaling scaffolding on skyscrapers under construction. Families plan their weekends, agents in the tourist industry plan events, and nervous brides and their families wrestle with moving the outdoor wedding into covered facilities.

Forecasting information is very important to almost everyone. More specifically, reliable and consistent weather forecasting data is in high demand. Unfortunately, this product is difficult to produce. Previous studies have concluded that *the weather is an undeniably complex process—and like any process, it can exhibit a lot of variation*.[1]

This begs the question regarding weather forecasts from around the world. Do some forecasts have less variation than others and are some forecasts simply more accurate in some places than other locations? Are all weather prediction models equal in quality and predictive power? Is it possible to classify some models as 'best of the best' and others as 'worse of the worse'?

The Team

This most significant success factor beside commitment to a successful project is the assigning of roles and tasks to the four team members. The division of labor was as follows.

- Analysis and Visualization - XT used her knowledge acquired from the Visualization course this semester along with her keen analytical mind.
- 24/7 Data Collection - Rama (Chris supporting) used Java/DB skills to get the process started of downloading files 'as the world turned'.
- Production System Design and Development - Chris (Rama supporting) used his innate curiosity of technical system design and an ability to delve into minutia of technology to construct a final production system.
- Getting started and project management - Besides helping everyone with their tasks, Chuck played an indispensable role of a project manager. He pushed the project forward from the very beginning till the finish line and made sure that we completed our tasks and requirements on time.

[1] https://www.minitab.com/en-us/Published-Articles/Weather-Forecasts--Just-How-Reliable-Are-They-/

Searching for a Data Source

Weather data is one of the most ubiquitous types of information available through the internet. However, it was challenging locating a source of data that met the following criteria.

- Historical actual weather data that included corresponding historical forecast data.
- Free or low cost data source.
- Readily accessible data simple to query.
- Data from locations throughout the world.

The team made several observations while searching for a good weather data source.

It became obvious after considerable searching that historical forecast data was not being retained long-term. For the most part it was absent from all of known repositories of weather data.

Historical data was available for free but in limited sizes. Larger sources of data had query per minute limits and total limits for the day. Greater volumes were expensive. OpenWeatherMap offered one-year old historical data at cost of $950 per month with a limit of 50,000 API calls per day.

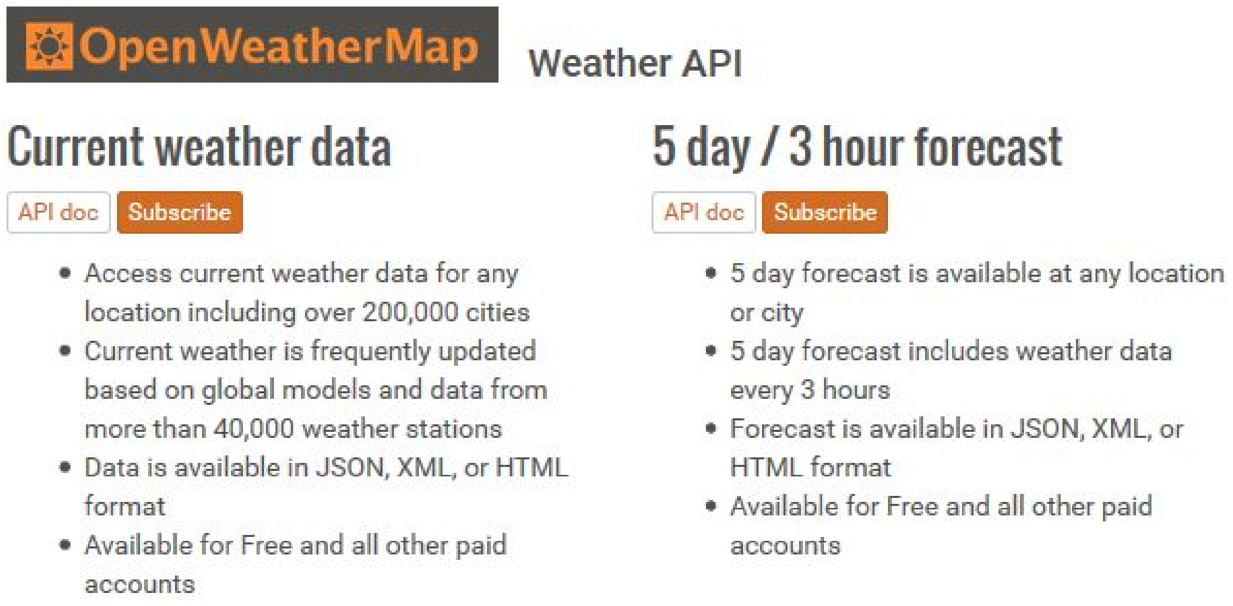
Data was provided in varying formats ranging from PDF, CSV, to JSON. There was a considerable amount of data worldwide embedded in web pages produced by local news organizations. Without a standard HTML/Javascript/CSS format for displaying this data it would be very difficult scrape so many pages. Essentially. a function would need to be created for each weather source. Worldwide coverage would requires tens of thousands such functions.

The team decided upon a better way.

Building a Repository of Quality Data

The team agreed that for $40 per month for two months, 600 API calls per minute could collect sufficient current weather conditions and five-day forecasts for 74,071 cities around the world in all twenty-four time zones. Given the constraint of the semester it was agreed upon that our team could collect over 35 days worth of data. In fact, when this paper is submitted for grading our team will have collected data for 53 days.

Thus the team would have at its disposal one of the only archives of both actual and forecast data in the world.



Current weather data

- Access current weather data for any location including over 200,000 cities
- Current weather is frequently updated based on global models and data from more than 40,000 weather stations
- Data is available in JSON, XML, or HTML format
- Available for Free and all other paid accounts

5 day / 3 hour forecast

- 5 day forecast is available at any location or city
- 5 day forecast includes weather data every 3 hours
- Forecast is available in JSON, XML, or HTML format
- Available for Free and all other paid accounts

Project Goals

The goal of this project is simple:

- Analyze historical weather data from around the world and compare the actual conditions with archived forecasted data.
- Use these comparisons to calculate a score by location that represents the quality of the forecast for each measure including a composite score.
- Utilize the scores to produce lists of the most accurate forecasts by location and time of year.
- Compile the results and place the data into the hands of consumers.
- In addition, analyze the compilation in search of patterns and evidence of superior and inferior predictive models.

About Weather Forecast Evaluator (WFE)

The WFE is a weather data collection system that uses the OpenWeatherMap API to collect actual and forecasted weather data, uses Amazon systems and software to store, transform, and present data to the consumer. This is applied to over 70,000 cities worldwide.

The target audience is anyone on the planet who has a strong interest in assessing the reliability of weather forecasts for one or more cities.

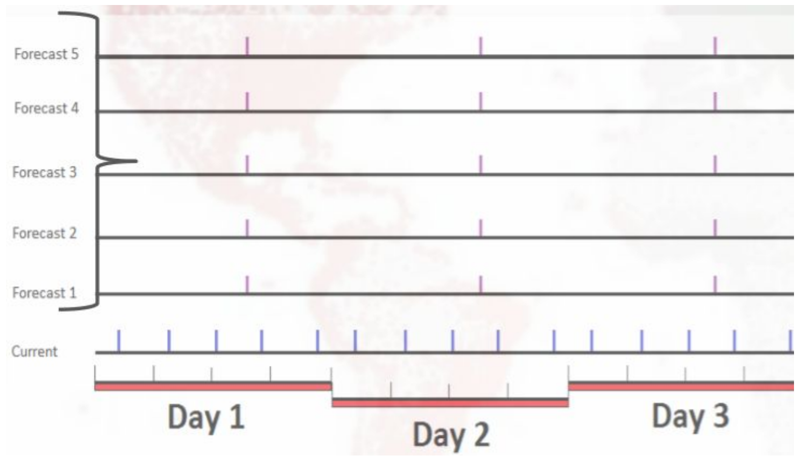
A Difficult and Challenging Task

Given any particular day in which a current weather conditions file exists, there are a corresponding five forecast files.

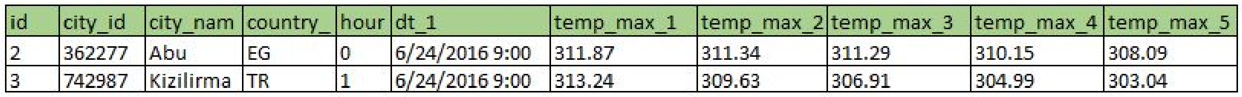
For example, imagine today is Friday and a current weather conditions file is downloaded. On this same day there exists a forecast file which we call ‘day 1’ forecast. This is a forecast created earlier in the morning regarding the remainder of the day.

On the previous day Thursday, a forecast is generated regarding Friday. This is referred to as ‘day 2’ forecast. Similarly on the prior Wednesday, a forecast is generated for this same Friday. This is called ‘day 3’ forecast. This pattern repeats for the remaining days.

Conceptually, a day 1 forecast should be the most accurate and a day 5 forecast should have the greatest variation.



The figure above illustrates a timeline representing time-based weather data from six files named ‘Current’, ‘Forecast 1’, etc. The challenge in this project is that these six files must be “stitched” together in such a way that a single database query includes the current metric along with each of the forecast days.The following graphic shows the forecasted metric ‘temperature max (temp_max)’ stitched together from five files. The current weather data exists in a separate table that is easily queried and joined with the forecast table.



id	city_id	city_nam	country_	hour	dt_1	temp_max_1	temp_max_2	temp_max_3	temp_max_4	temp_max_5
2	362277	Abu	EG	0	6/24/2016 9:00	311.87	311.34	311.29	310.15	308.09
3	742987	Kizilirma	TR	1	6/24/2016 9:00	313.24	309.63	306.91	304.99	303.04

Architecture

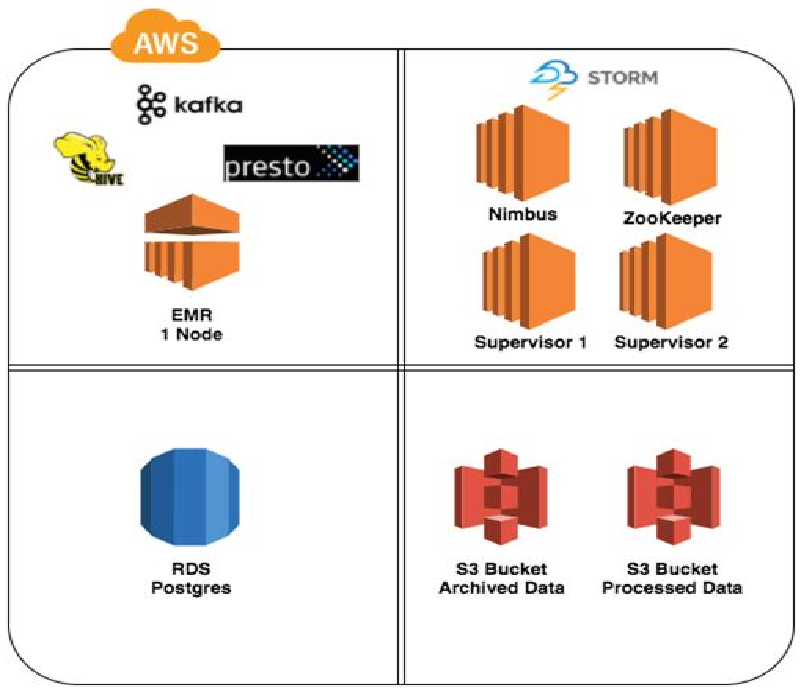
Overview

We used Amazon Web Services (AWS) for our computing and storage needs. AWS provided resources that were accessible by all members of the group and provided the capability to scale horizontally if needed. AWS also provided easy access to machine images with many of the tools we planned on implementing for our project (including Hive, Presto, and S3).

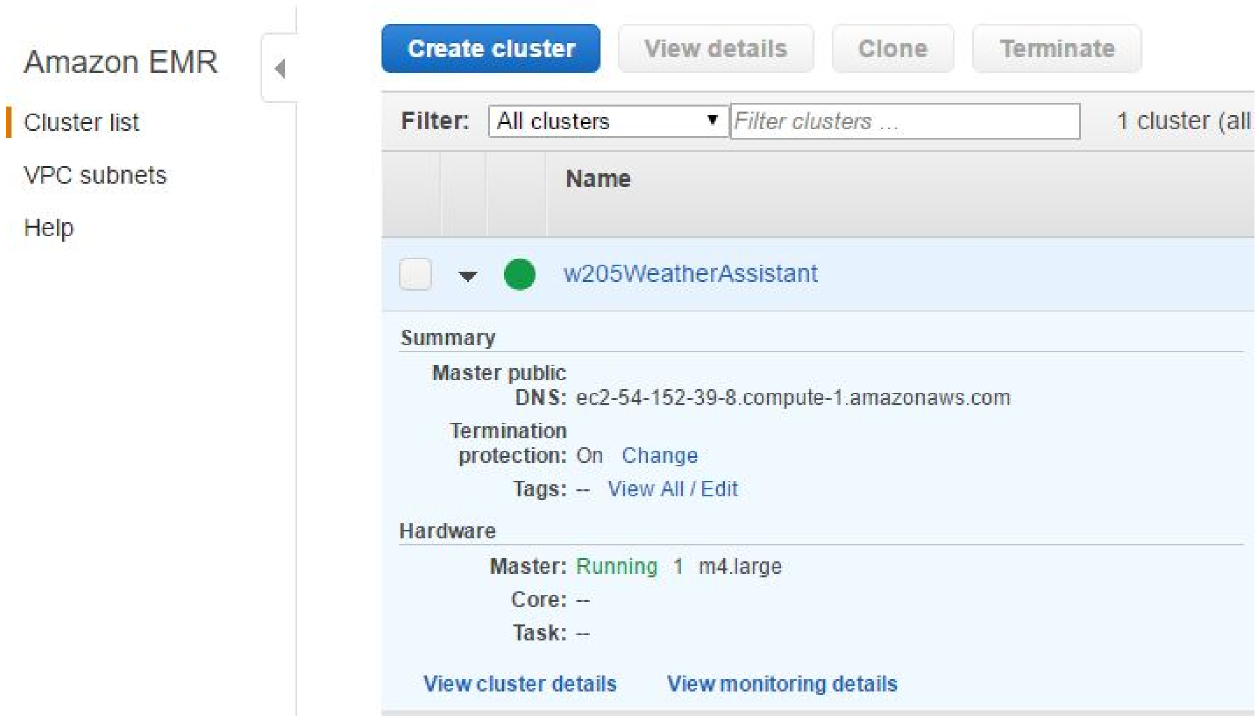
Tools

The Weather Forecast Evaluator required a suite of tools to successfully retrieve, process, and store the data required for the final analysis. For data retrieval we relied on a Cron scheduler. For data processing we relied on Apache Kafka as our messaging queue and Apache Storm for processing (we also evaluated Spark Streaming in the early planning phases and may be worth reevaluating). For data storage we used AWS S3 object storage for both the unprocessed data archive and the processed data storage. Apache Hive was used for it's metastore exclusively as Presto, our speed of thought query engine, relied on the Hive metastore to represent the tabular data stored in S3.

Infrastructure



Powering the Kafka messaging queues, the Presto “speed of thought” query engine, and the Hive metastore is a single AWS EMR node as visualized in the top right quadrant above.



Filter: All clusters | Filter clusters ... | 1 cluster (all

Name

w205WeatherAssistant

Summary

Master public DNS: ec2-54-152-39-8.compute-1.amazonaws.com

Termination protection: On Change

Tags: -- View All / Edit

Hardware

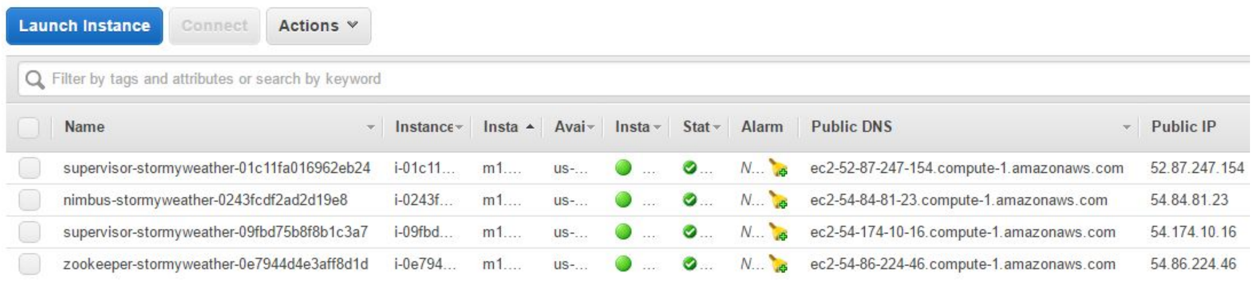
Master: Running 1 m4.large

Core: --

Task: --

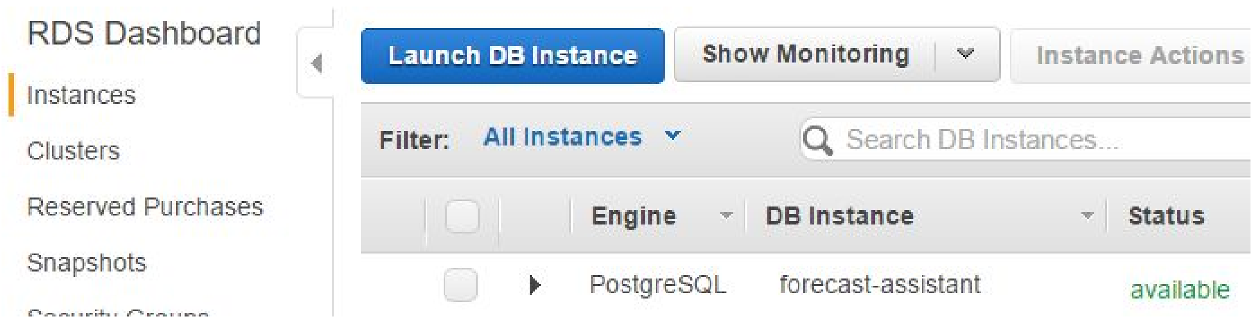
View cluster details View monitoring details

The Storm infrastructure consists of 4 AWS EC2 machines. One Nimbus node, one Zookeeper node, and two Supervisor nodes as visualized in the top left quadrant above. Distribution of the workload across multiple clusters allows the processing to keep up with the message throughput.



Name	Instance	Insta	Avai	Insta	Stat	Alarm	Public DNS	Public IP
supervisor-stormyweather-01c11fa016962eb24	i-01c11...	m1....	us-...	● ...	✓...	N...	ec2-52-87-247-154.compute-1.amazonaws.com	52.87.247.154
nimbus-stormyweather-0243fcdf2ad2d19e8	i-0243f...	m1....	us-...	● ...	✓...	N...	ec2-54-84-81-23.compute-1.amazonaws.com	54.84.81.23
supervisor-stormyweather-09fbd75b8f8b1c3a7	i-09fbd...	m1....	us-...	● ...	✓...	N...	ec2-54-174-10-16.compute-1.amazonaws.com	54.174.10.16
zookeeper-stormyweather-0e7944d4e3aff8d1d	i-0e794...	m1....	us-...	● ...	✓...	N...	ec2-54-86-224-46.compute-1.amazonaws.com	54.86.224.46

For application metadata storage there is one Postgres AWS RDS instance and for processed and unprocessed data storage there are two AWS S3 buckets as shown in the bottom two quadrants.



Engine	DB Instance	Status
PostgreSQL	forecast-assistant	available

Implementation

This seconds outlines implementation details. End-to-End data flow is outlined in this section.

Data Acquisition

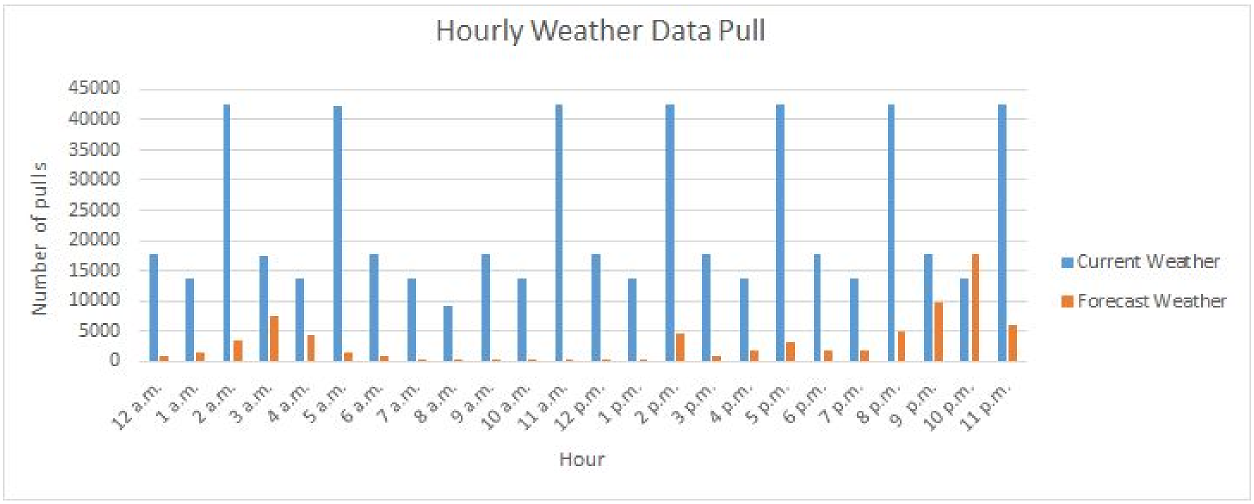
Primary weather data source is OpenWeatherMap.org. They provide a number of weather details for over 200,000 cities ranging from current weather to a wide array of forecast data. Right set of easy to use APIs are available for data retrieval and integration.

Two types of weather data were considered for the project:

Weather Data Type	API
Current Weather - data provides a snapshot of the current weather. Data ranging from rain, temperature, wind, cloud conditions, were available. Data source updates data frequently based on global models and data from more than 40,000 weather stations.	http://api.openweathermap.org/data/2.5/weather?id=
Forecast Weather - data provides seven day prediction for a given city.	http://api.openweathermap.org/data/2.5/forecast/daily?id=

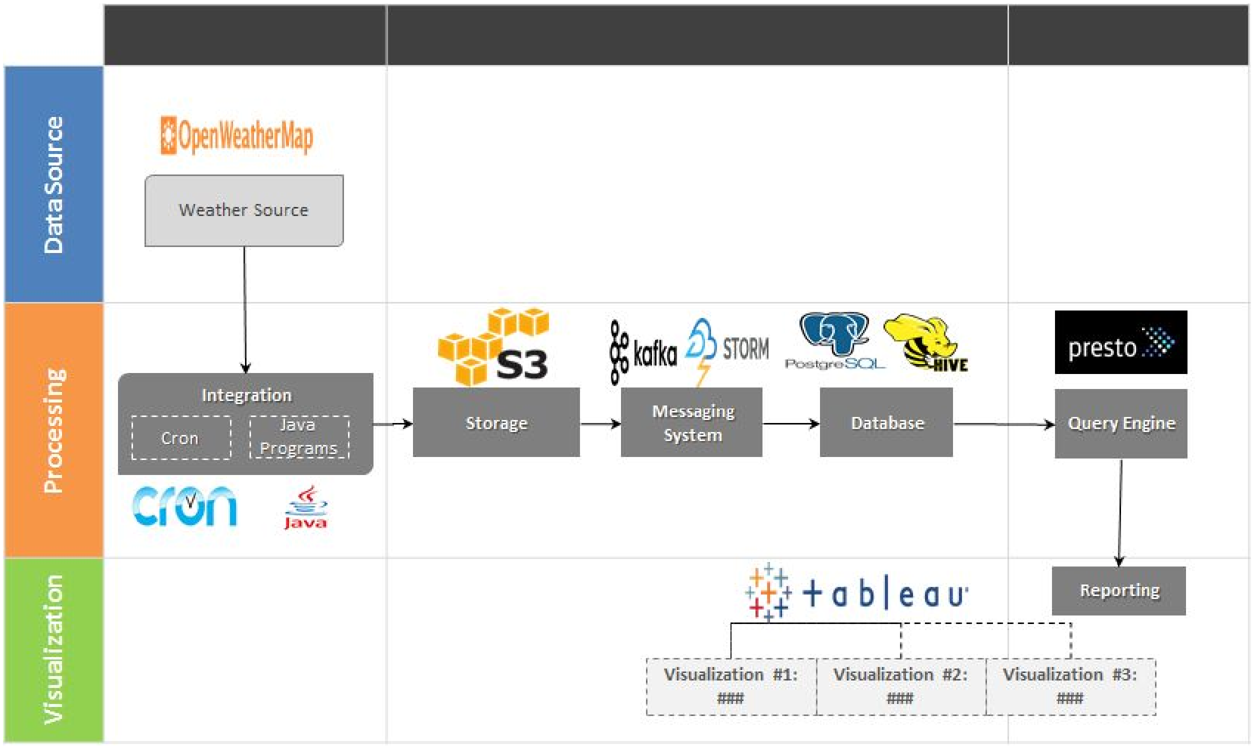
Data Source does not provide historic weather data, as such the integration layer required high resiliency in fetching current weather with very small/no room for failure. Retrieval code was built with robust error handling and retry mechanisms. Data retrieval process was spread out for each hour. Cities were grouped into different zones and each zone was processed each hour. We created 21 zones.

Current weather was retrieved eight times a day for a given city and Forecast weather was pulled once a day for each city. Chart below show number data pulls for each data type.



Data Flow

Diagram below outlines the data follow. The follow is grouped in three different swimlanes, Data Source, Processing and Visualization. Next couple of sections outline each of the groups.



Data Source:

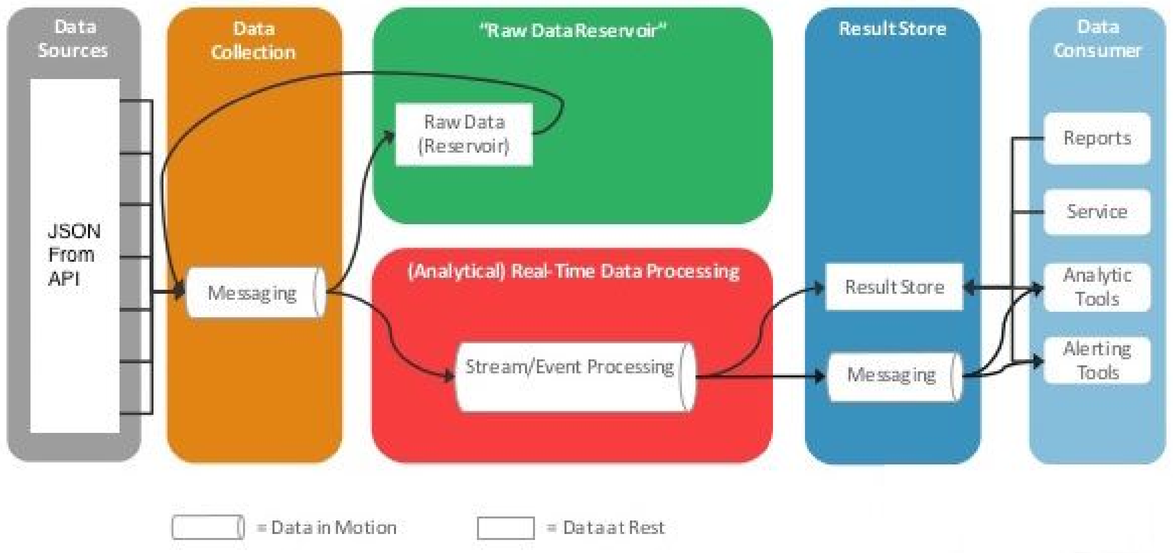
This layers hosts integration infrastructure that pulls weather data from the datasource. Integration logic was implemented using Java. CRON, a Linux utility triggers the Java programs to pull data for a given hour. The process was triggered at the top of every hour. Java programs use configuration in the CSV files to figure out which cities and weather data types to process for a given hour.

Data retrieved were transferred to S3 for data storage resilience. In addition, data was stored locally on the integration servers. This was done to enhance storage resilience. A clean-up job was configured to remove data older than three days in order to maintain local storage space.

Integration layer provides flexibility to write to a Kafka queue, in addition to storing in S3, should we need a real-time processing instead of a batch processing.

Processing:

The stream processing implementation was built using the Kappa architecture. This architecture treats everything flowing through it as a stream. This means that even when you need to reprocess the historic data you are simply replaying the data through the stream. A diagram representing this architecture is below.



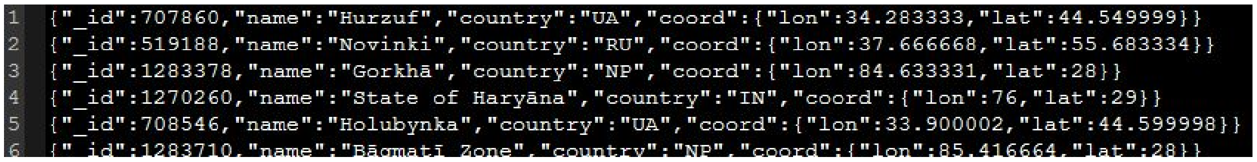
Kafka is used as our "Data Collection" system as visualized in the orange section of the graphic above. Kafka is sent data from S3 and persists the data for 5 days to facilitate any short term data replays. Kafka makes the data available for Storm and keeps track of Storms position in the queue. Storms primary job is to deserialize the json into individual elements to insert into our database tables. We also use Storm to transform the data and, in some cases, enhance the data. Examples of this are converting Unix timestamps to human readable timestamps and extracting data from the filename as additional elements. We had difficulty building a reliable Storm cluster, so we also built a batch processing system to run in parallel as a backup to the Storm system. In completing the processing steps, Storm, or the batch system, persists the results in external Hive tables. The external Hive tables use S3 as the storage layer and Hive as the metastore for the table metadata.

Visualization:

Tableau is used for data visualization. Flexibility exists to connect to Postgres or Hive through Presto query engine.

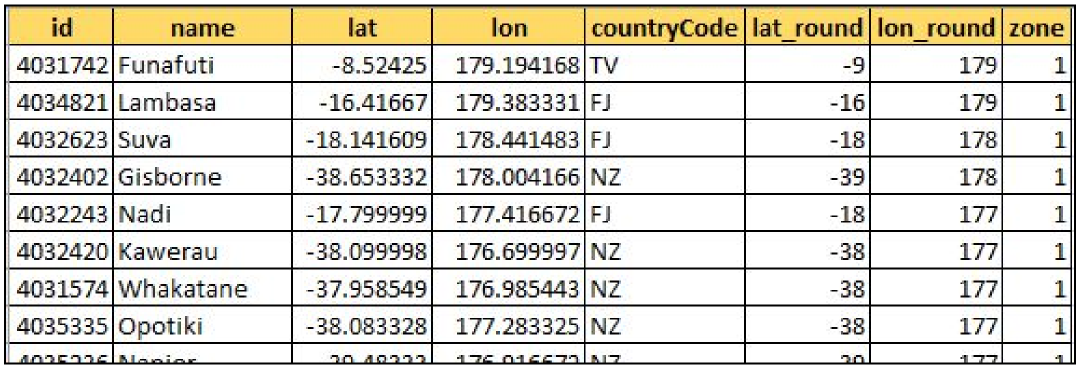
Ad Hoc Processing for Analysis:

The metadata consists of a unique city id, city name, country, latitude, and longitude. This data is obtained from http://openweathermap.org/help/city_list.txt and is provided in JSON format.



```
{"_id":707860,"name":"Hurzuf","country":"UA","coord":{"lon":34.283333,"lat":44.549999}}
{"_id":519188,"name":"Novinki","country":"RU","coord":{"lon":37.666668,"lat":55.683334}}
{"_id":1283378,"name":"Gorkhā","country":"NP","coord":{"lon":84.633331,"lat":28}}
{"_id":1270260,"name":"State of Haryāna","country":"IN","coord":{"lon":76,"lat":29}}
{"_id":708546,"name":"Holubynka","country":"UA","coord":{"lon":33.900002,"lat":44.599998}}
{"_id":1283710,"name":"Bāgmatī Zone","country":"NP","coord":{"lon":85.416664,"lat":28}}
```

The data is downloaded and converted into CSV using Python and further manipulated in Excel and then uploaded into the PostgreSQL database.



id	name	lat	lon	countryCode	lat_round	lon_round	zone
4031742	Funafuti	-8.52425	179.194168	TV	-9	179	1
4034821	Lambasa	-16.41667	179.383331	FJ	-16	179	1
4032623	Suva	-18.141609	178.441483	FJ	-18	178	1
4032402	Gisborne	-38.653332	178.004166	NZ	-39	178	1
4032243	Nadi	-17.799999	177.416672	FJ	-18	177	1
4032420	Kawerau	-38.099998	176.699997	NZ	-38	177	1
4031574	Whakatane	-37.958549	176.985443	NZ	-38	177	1
4035335	Opotiki	-38.083328	177.283325	NZ	-38	177	1
4035226	Napier	-39.48333	176.916672	NZ	-39	177	1

Data Characteristics

Velocity

Weather data was acquired round the clock. Every second data for seven cities were pulled on an average. Over half a million data pulls were done providing the system the flexibility to process files in real-time or in batch modes.

Daily	0.63 Million pulls
Hourly	440 pulls
Per Second	7 pulls

Volume

Volume is an important data characteristics for the project. Over the period of the project, about 12 weeks, approximately 60 gigabytes of data was collected and stored.

Per Day	0.7GB
Entire Duration	~60GB

Variety

Weather data accessed from OpenWeatherMap was in JSON format. Current weather and Forecast data had different fields but were in JSON format. Configuration information used for integrating with OpenWeatherMap were in CSV format. Configuration includes:

- Cities grouped into different zones
- Zones trigger list for current weather
- Zones trigger list for forecast weather

Weather Data	JSON (JavaScript Object Notation)
Configuration	CSV (Comma Separated Values)

Analysis

Analysis Goals

To align with the final goals of the project, we conducted analysis to ‘score’ accuracy of forecasts for each city and type of weather forecast. Specifically, we seek to answer questions such as the following:

- What is the forecast accuracy for each city for each of the weather type? (weather forecasts as provided by our dataset include temperature, humidity, pressure, rain, weather description, weather name and wind speed)

- What are the top 10 cities with the most accurate humidity forecast?
- What are the top 10 cities with the worst weather forecast for weather description?
- What is the variance of temperature forecast from actual temperature in Rio in the last 30 days?
- Is there any relationship between weather forecast accuracy among countries of different average elevations?

Note that all of our analysis results below are limited to our data sets that we have at the time of running. You might get very different results by running the same queries on a larger data set after we have streamed the data for a longer period of time.

Results and Visualization

In this sessions, we present a few common questions that our users might be interested in when interacting with our system:

What is the forecast accuracy for each city for each of the weather type?

1) Methodologies

To score weather forecast accuracy, we split forecast types to numeric-type forecasts and string-type forecast. We will also follow these methodologies for the rest of the analysis questions.

For numeric-type forecasts such as temperature, humidity, pressure and rain, we calculate the deviation of forecast from the actual data using the following formula:

$$D = \sqrt{\frac{\sum_{i=1}^{N}(forecast - actual)^2}{N}}$$

Where:

D: deviation of forecast from actual
forecast: value of a numeric-type forecast
actual: actual value
N: number of forecasts made

A low deviation indicates that the forecast value tend to be closer to actual value, while a high deviation indicates that the forecast value tend to be spread out over a wider range of values from the actual value. To

For string-type forecasts such as weather name and weather description, we calculate the percentage of match between forecast and actual data:

$$Score = \% \frac{accurate\,forecasts}{N}$$

Where:

accurate forecasts: forecasts that matches the actual data for a particular forecast made.
N: number of forecasts made

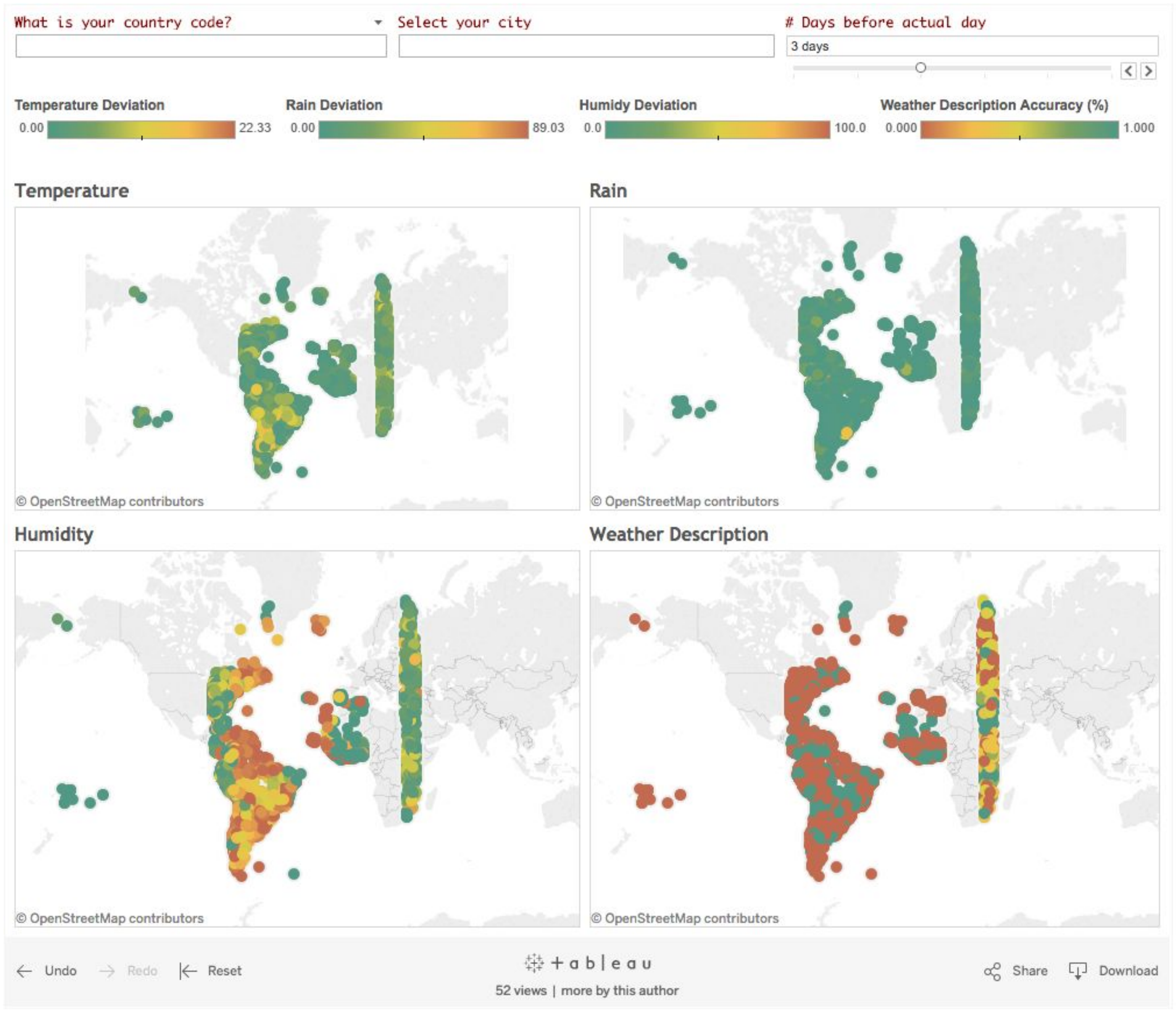
A high score indicates that forecast tend to be match the actual value, while a low score indicates that forecast rarely matches the actual values.

2) Results and Visualization

Following the methodology outlined above, we used Tableau to visualize weather forecast accuracy for all cities in the world. Users can interact with our visualization[2] by inputting the country codes and cities that they are interested in and learn more about weather forecast accuracy for those locations. For example, try inputting “RU” in “What is your country code?” or “Boston” in “Select your city” filters.

Below is a screenshot of the visualization on 08/17/2016. The longer we stream the data, the more cities will be covered. Eventually, we will have the entire world colored with weather forecast accuracy.

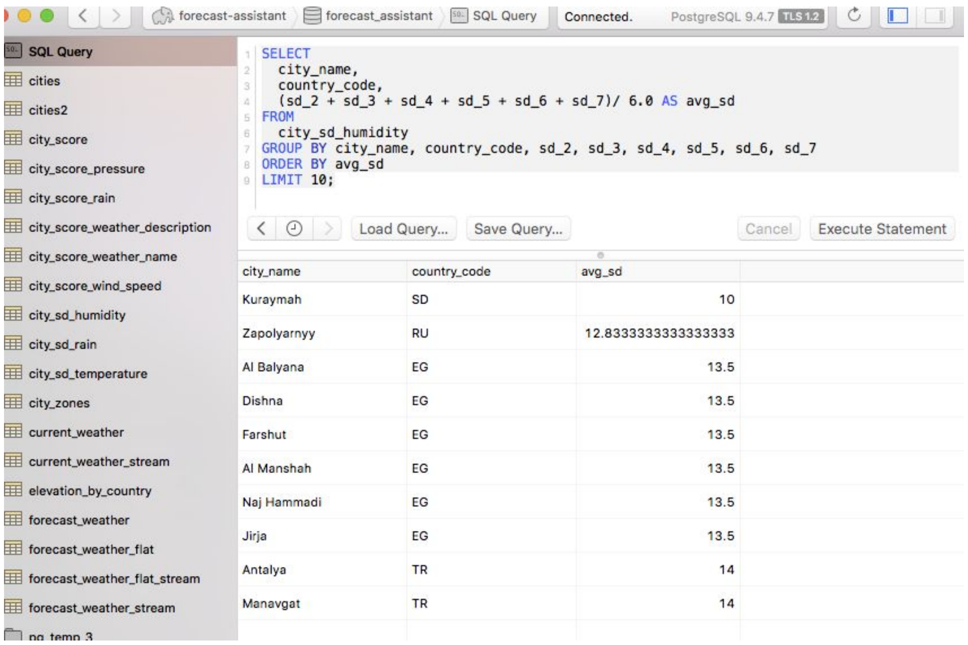
[2] https://public.tableau.com/views/w205_fp_viz/Dashboard?:embed=y&:display_count=yes



What is the deviation of temperature forecast from actual temperature in Rio during the Olympics?
Using our dataset, we can also zoom in a particular city for a particular period of time. For people who are interested in Rio Olympics 2016, this query[3] gives us the standard deviation of temperature forecast in Rio during the Olympics.

What are the top 10 cities with the most accurate humidity forecasts of all time?
To calculate top cities with the most accurate humidity forecasts of all time, we take the average of deviations of forecasts from actual values for all forecasts made from one day before to 6 days before. The cities with the most accurate humidity forecasts have the smallest deviations from their actual data. Below is the script[4] and result.

[3] https://github.com/xtnguyen-ucb/w205_fp_analysis/blob/master/rio_temp_deviation_during_olympic.sql
[4] https://github.com/xtnguyen-ucb/w205_fp_analysis/blob/master/top_10_humidity.sql

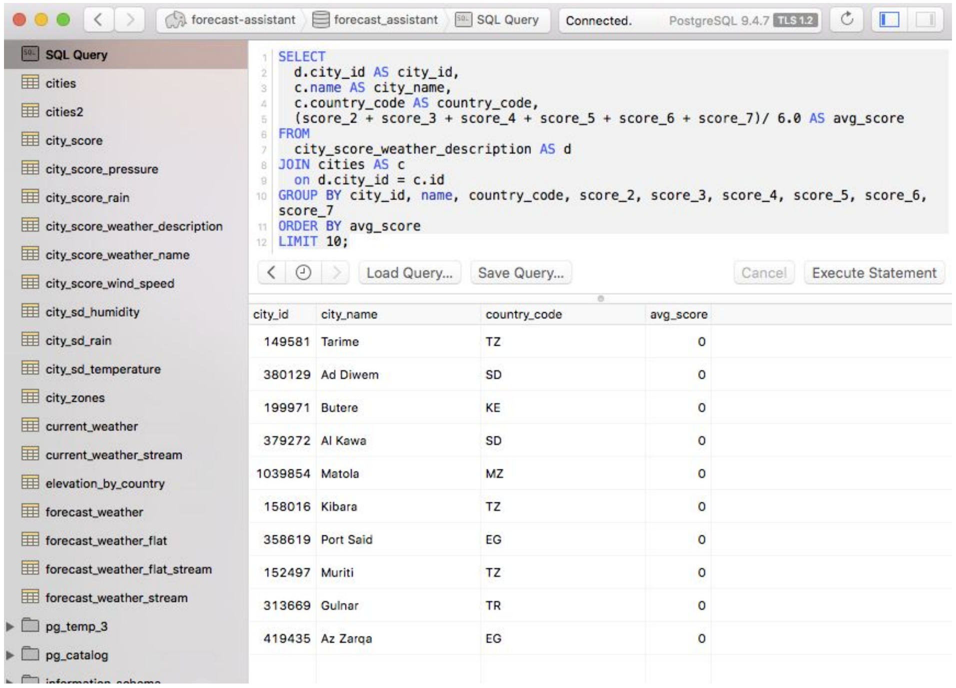


```sql
SELECT
  city_name,
  country_code,
  (sd_2 + sd_3 + sd_4 + sd_5 + sd_6 + sd_7)/ 6.0 AS avg_sd
FROM
  city_sd_humidity
GROUP BY city_name, country_code, sd_2, sd_3, sd_4, sd_5, sd_6, sd_7
ORDER BY avg_sd
LIMIT 10;
```

city_name	country_code	avg_sd
Kuraymah	SD	10
Zapolyarnyy	RU	12.8333333333333333
Al Balyana	EG	13.5
Dishna	EG	13.5
Farshut	EG	13.5
Al Manshah	EG	13.5
Naj Hammadi	EG	13.5
Jirja	EG	13.5
Antalya	TR	14
Manavgat	TR	14

What are the top 10 cities with the worst weather forecast for weather description of all time?

These are the top 10 cities with the worst weather description forecasts of all time. Average score of 0 means that weather description forecasts were never accurate for these locations during our data collection period. Below is the script[5] and result.



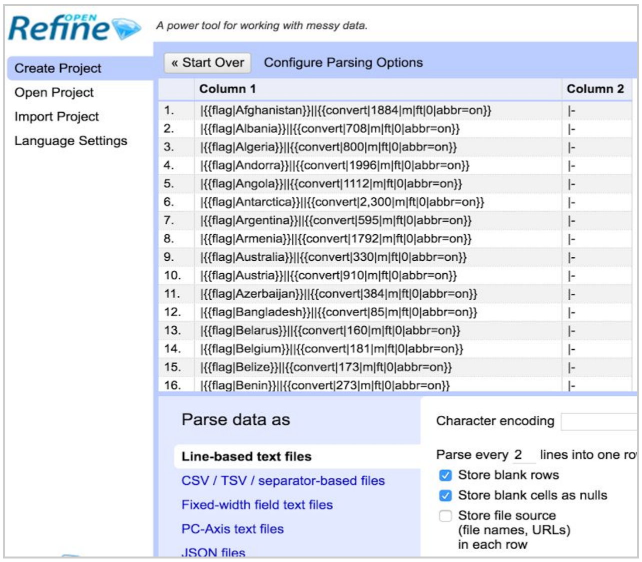
```sql
SELECT
  d.city_id AS city_id,
  c.name AS city_name,
  c.country_code AS country_code,
  (score_2 + score_3 + score_4 + score_5 + score_6 + score_7)/ 6.0 AS avg_score
FROM
  city_score_weather_description AS d
JOIN cities AS c
  on d.city_id = c.id
GROUP BY city_id, name, country_code, score_2, score_3, score_4, score_5, score_6,
score_7
ORDER BY avg_score
LIMIT 10;
```

city_id	city_name	country_code	avg_score
149581	Tarime	TZ	0
380129	Ad Diwem	SD	0
199971	Butere	KE	0
379272	Al Kawa	SD	0
1039854	Matola	MZ	0
158016	Kibara	TZ	0
358619	Port Said	EG	0
152497	Muriti	TZ	0
313669	Gulnar	TR	0
419435	Az Zarqa	EG	0

[5]https://github.com/xtnguyen-ucb/w205_fp_analysis/blob/master/worst_10_weather_description_forecasts.sql

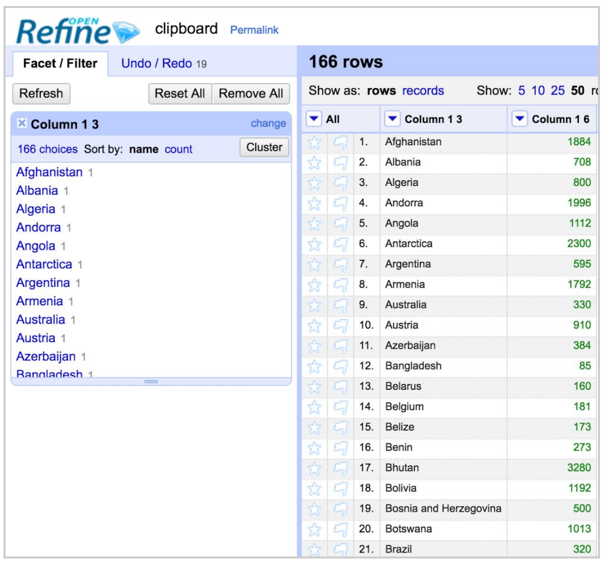
Is there any relationship between weather forecast accuracy among countries of different average elevations?

To answer this question, we found data on Wikipedia[6] on elevation by country. However, since the data had formatting issues, we had to download and clean it using OpenRefine[7]. The figure below shows the original form and the final table that was uploaded to our database.



	Column 1	Column 2
1.	|{{flag|Afghanistan}}||{{convert|1884|m|ft|0|abbr=on}}	|-
2.	|{{flag|Albania}}||{{convert|708|m|ft|0|abbr=on}}	|-
3.	|{{flag|Algeria}}||{{convert|800|m|ft|0|abbr=on}}	|-
4.	|{{flag|Andorra}}||{{convert|1996|m|ft|0|abbr=on}}	|-
5.	|{{flag|Angola}}||{{convert|1112|m|ft|0|abbr=on}}	|-
6.	|{{flag|Antarctica}}||{{convert|2,300|m|ft|0|abbr=on}}	|-
7.	|{{flag|Argentina}}||{{convert|595|m|ft|0|abbr=on}}	|-
8.	|{{flag|Armenia}}||{{convert|1792|m|ft|0|abbr=on}}	|-
9.	|{{flag|Australia}}||{{convert|330|m|ft|0|abbr=on}}	|-
10.	|{{flag|Austria}}||{{convert|910|m|ft|0|abbr=on}}	|-
11.	|{{flag|Azerbaijan}}||{{convert|384|m|ft|0|abbr=on}}	|-
12.	|{{flag|Bangladesh}}||{{convert|85|m|ft|0|abbr=on}}	|-
13.	|{{flag|Belarus}}||{{convert|160|m|ft|0|abbr=on}}	|-
14.	|{{flag|Belgium}}||{{convert|181|m|ft|0|abbr=on}}	|-
15.	|{{flag|Belize}}||{{convert|173|m|ft|0|abbr=on}}	|-
16.	|{{flag|Benin}}||{{convert|273|m|ft|0|abbr=on}}	|-

Parse every 2 lines into one ro
[✓] Store blank rows
[✓] Store blank cells as nulls
[ ] Store file source (file names, URLs) in each row



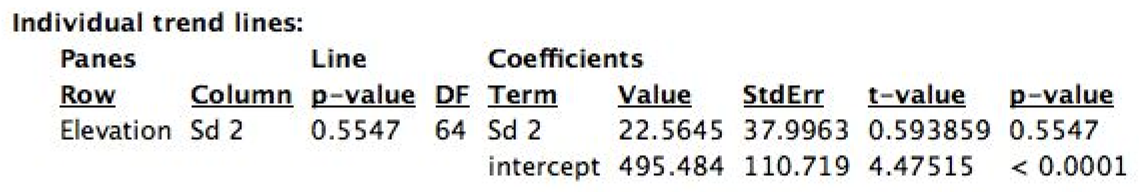
Afghanistan 1
Albania 1
Algeria 1
Andorra 1
Angola 1
Antarctica 1
Argentina 1
Armenia 1
Australia 1
Austria 1
Azerbaijan 1
Bangladesh 1

166 rows

Show as: rows records Show: 5 10 25 50

All	Column 1 3	Column 1 6
1.	Afghanistan	1884
2.	Albania	708
3.	Algeria	800
4.	Andorra	1996
5.	Angola	1112
6.	Antarctica	2300
7.	Argentina	595
8.	Armenia	1792
9.	Australia	330
10.	Austria	910
11.	Azerbaijan	384
12.	Bangladesh	85
13.	Belarus	160
14.	Belgium	181
15.	Belize	173
16.	Benin	273
17.	Bhutan	3280
18.	Bolivia	1192
19.	Bosnia and Herzegovina	500
20.	Botswana	1013
21.	Brazil	320

Below is a scatter plot for average elevation by country against temperature deviation by country, together with a linear trend model. We can observe that there is not a strong relationship between average elevation and temperature forecast accuracy (based on deviation) by country.

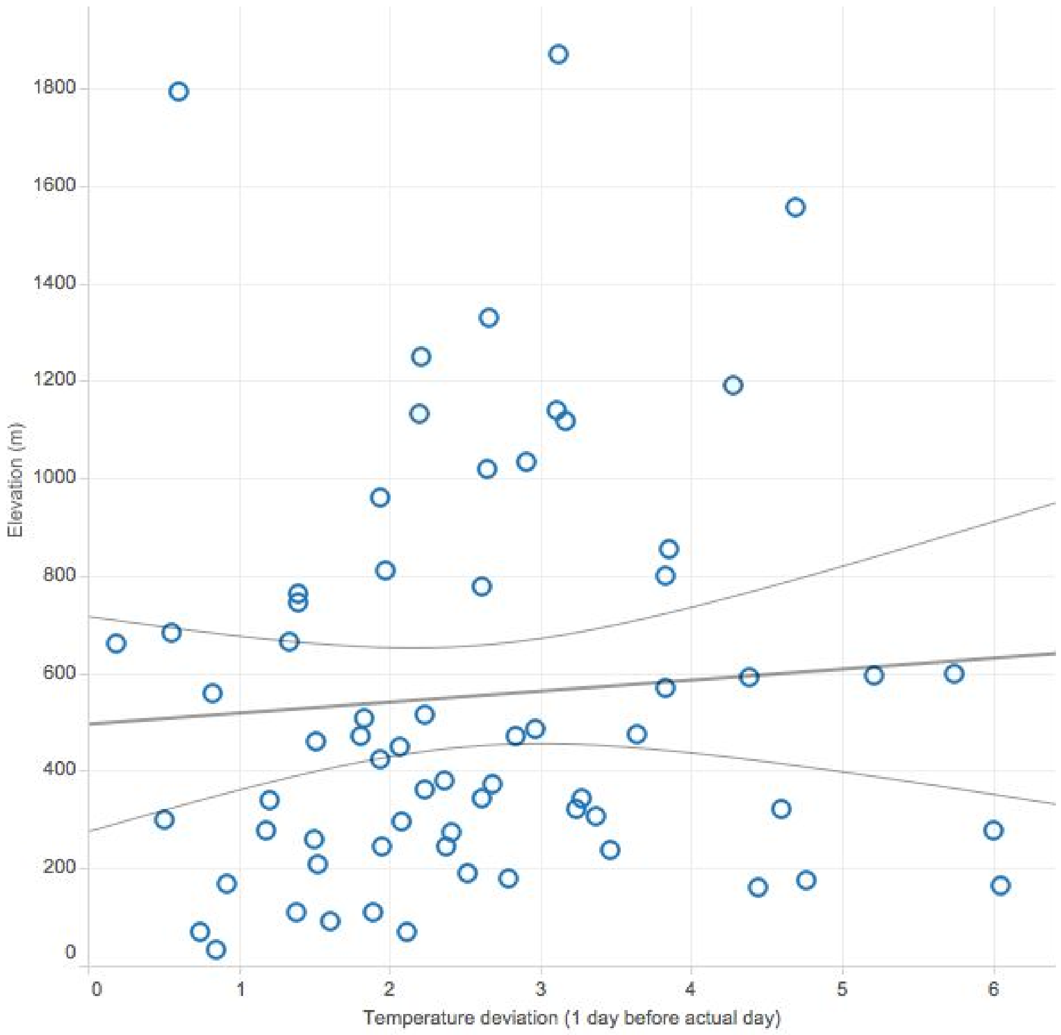


Individual trend lines:

Panes		Line		Coefficients				
Row	Column	p-value	DF	Term	Value	StdErr	t-value	p-value
Elevation	Sd 2	0.5547	64	Sd 2	22.5645	37.9963	0.593859	0.5547
				intercept	495.484	110.719	4.47515	< 0.0001

[6] https://en.wikipedia.org/wiki/List_of_countries_by_average_elevation

[7] http://openrefine.org/



Post Mortem

What went well?

There are two distinct observations that stand out about this project. The first observation is that our team worked very well together throughout the twelve-week project cycle. Specifically,

- The team agreed to meet weekly for an hour and during this window of time. This facilitated a lot of decision making upfront as well as providing weekly progress reports.
- The team recognized the need to share information with one another across the three time zones and to retain a history of these various conversations for subsequent reference. The team created a Slack.com account at https://mids205weather2016s.slack.com/messages/@slackbot/. This included six channels related to the project.
- As described earlier in the section named “The Team”, each member really seemed to dominate their respective area of contribution. It went very smoothly.
- The resultant system successfully stitched together actual and forecasted data.
- Early on in the project, the team received hourly emails notifying them of the number of files processed within that time period, along with any errors associated with downloading that were automatically retried.
- At no time did the team venture off tangent. Everyone was cognizant of the project outcome and acutely aware of the limited time that members had available due to competing interests of their jobs and course load.
- At no time did panic overcome team members. Issues were discussed and all ramifications were considered.

What could have gone better?

Naturally, time to work the final project was in constant competition with W205 labs, exercises, and asynchronous material such as lengthy readings and weekly videos. Specifically,

- More time to experiment with the various systems and software would have allowed for greater development of skills that may have resulted in an even better production system.
- Scoping of the streaming processing Kappa architecture effort was not optimal. The time invested in setting up a Storm cluster was much more than anticipated (and did not end up working). However, it provided an invaluable learning experience.
- Analysis was relegated literally to the final hours before the presentation by the team to the W205 class. Several weeks of additional time would have allowed the team to answer many of the questions specified in section titled “Goals”.
- There was no time for cross-training. For example, Chris delved into the intricacies of a viable production system that included many components described in the architecture. As a consequence, other team members did not have the opportunity to ‘get their hands dirty’ with a practical application.
- Additional data sources were discussed in the beginning but was not included. The sources consisted of data related to station elevation, solar radiation levels, pollution indices, climate of the region, education level of meteorologists, as well as information related to the various technologies used around the world to make forecasts.

Appendix

Important Connections:

Raw API data archive (S3):
Access key: AKIAJEELMKR3NAPXHURA
Secret key: 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
Bucket: weather-raw-json

Processed data storage (S3):
Public key: AKIAJPPB7B57OF64A6MQ
Private key: hugyxonaNYBU1jT9ckWoPUj6tOHk3qZMILBhnVGE
Bucket: mids205-weather-data

“Quick and Dirty” and metadata storage (Postgres):
Connection string (tested in SQLworkbench sql client):
jdbc:postgresql://forecast-assistant.c82uau1daq0n.us-east-1.rds.amazonaws.com:5432/forecast_assistant
Username: master
Password: 5elVPNit

Presto:
Connection string: jdbc:presto://ec2-54-152-39-8.compute-1.amazonaws.com:8889/hive
Username: hadoop
Password: [leave empty]

AWS EMR:
Public DNS: ec2-54-152-39-8.compute-1.amazonaws.com

Storm:
Nimbus Public DNS: ec2-54-84-81-23.compute-1.amazonaws.com
Zookeeper Public DNS: ec2-54-86-224-46.compute-1.amazonaws.com
Supervisor 1 Public DNS: ec2-52-87-247-154.compute-1.amazonaws.com
Supervisor 2 Public DNS: ec2-54-174-10-16.compute-1.amazonaws.com