CMPT-732:

PROJECT DATASTORM

By Crystal Dias, Karishma Damania, Jeanne Helm

Semester:	Fall 2022
Date of Submission:	2022/12/09
Instructors:	Greg Baker, Kai Yee (George) Chow
Teaching Assistants:	Shivek Chhabra, Shrey Grover, Sambhav Rakhe

PROBLEM DEFINITION

Canada has always been a safe haven for refugees during war and crisis. Considering recent events, more than 32000 Ukrainians have found support and a safe place to live here. While Canada is a large country, most of it is not very suitable for living, especially for those seeking refuge. Our aim for this project is to provide a data backend study of different locations in Canada where the government could expand cities and build suitable shelters. The factors to be considered would be the region's weather, including temperature, rainfall, snowfall, and snow depth. With this information, the government could spend resources efficiently on setting up transportation, crops and livestock, infrastructure and housing, and sustainable energy sources.

For this project, we will use the GHCN (Global Historical Climatology Network) -daily dataset compiled for 180 countries and more than 100,000 stations by NCEI. The records of about 175 years have been compiled under this dataset. A descriptive analysis of this nature might need to be performed regularly due to the changing climatic conditions and demographic of Canada. The GHCNd dataset is a comprehensive set of observations recorded over several locations, but to remove meaningful conclusions to solve our problem statement, we may need a more scalable alternative. With the help of the big data tools we have dabbled in during this course, 'CMPT 732 - Programming for Big Data 1,' and more, we have designed a pipeline adhering to the 4-Vs volume, velocity, variety, and veracity.

This proposed methodology could be further modified to understand any geographical region's weather conditions better. From this dataset, we have selected three elements- Wind, Snow and Precipitation and analyzed them for this project.

METHODOLOGY

APACHE SPARK

All ETL for this project has been performed using Apache Spark. The raw GHCNd data, including - the information on different station IDs, station names, observation types, the duration for which the record was monitored, and the value of these observations - are in .dly and .txt files. With the help of Spark and data frames, we first implemented the extraction and transformation of helpful information into a predefined schema. For further transformation, we used the similar cleaned data available on the cluster provided for this course. As per our requirement, we further transformed the data into parquet files partitioned on observation values and stored it in our S3 bucket. This entire transformation was done with the help of Spark+DataFrames on the EMR cluster.

AMAZON ELASTIC MAPREDUCE (EMR)

In our architecture, we create a cluster for the purpose of running spark jobs with 1 master and 2 worker nodes/instances of type - c7g.xlarge. Our EMR Cluster is used to perform the ETL Phase of the project wherein we run the Python scripts for the Spark application on the cluster, which read the data from the S3 Bucket optimized with S3Select filter to efficiently reduce the amount of data that Amazon S3 transfers thus reducing the cost and latency to retrieve this data. The data after ETL is partitioned by the observation values and stored back into an S3 bucket as parquet files. All the cluster configurations, as well as partitioning details, are provided here.

AMAZON SIMPLE STORAGE SERVICE (S3)

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance and can be used to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. For our methodology, we use different S3 buckets to store the raw data, the scripts required by the Spark Application on the EMR Cluster, the processed data after ETL, as well as to hold the outputs of the queries performed subsequently in Athena. We also use S3 as a data lake to use in conjunction with AWS Glue Crawler and Catalog to obtain the final tables used for the analysis and visualization phase of the project.

AMAZON ATHENA

Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL. Amazon Athena also makes it easy to interactively run data analytics using Apache Spark without having to plan for, configure, or manage resources.

Athena SQL and Apache Spark on Amazon Athena are serverless, so there is no infrastructure to set up or manage, and we pay only for the queries we run. Athena scales automatically—running queries in parallel—so results are fast, even with large datasets and complex queries. This also expands the future scope of the project since these services can be easily scaled to deal with data that are many times larger than the current. Further, we used Athena in our architecture because it also easily integrates with the AWS Glue Data Catalog, which offers a persistent metadata store for our data which is in Amazon S3. This allowed us to create tables and query data in Athena based on a central metadata store available throughout our Amazon Web Services account and integrated with the ETL and data discovery features of AWS Glue and Crawlers. Further reasoning behind why we preferred to work with Amazon Athena over Redshift + Spectrum is detailed here.

AWS GLUE

Athena can only use the data stored in Amazon S3 after using the AWS Glue Data Catalog. AWS Glues provides a file crawler that we use to add metadata, like table and column names, to the data stored in S3. The mapping of our S3 filesystem to databases, tables, and views appears then in Athena's query editor.

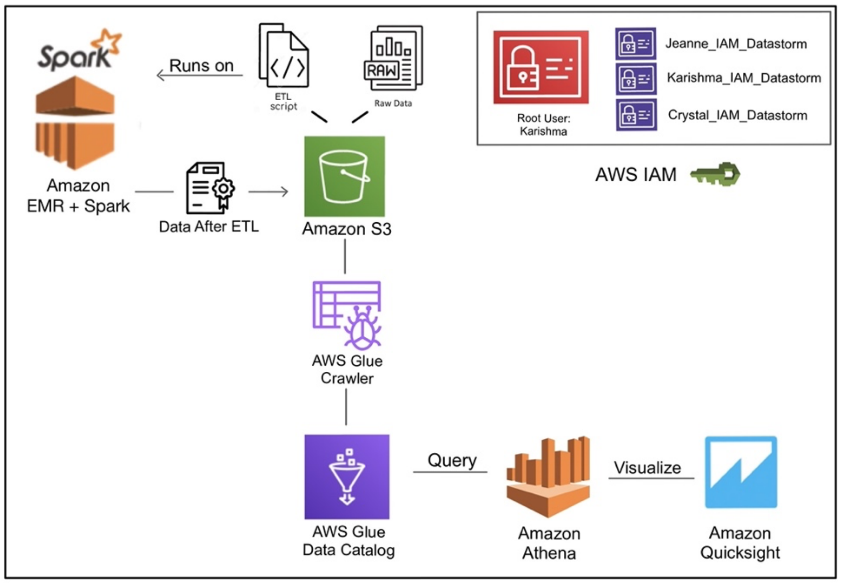
AMAZON QUICKSIGHT

Amazon QuickSight is a business analytics web service that makes it easy to build visualizations, perform ad-hoc analysis, and quickly get insights from data anytime on any device. It can be connected to Redshift, S3 or Athena. It is also possible to use Amazon's Lambda Connector to use a wider variety of services, e.g., Amazon DynamoDB. In this project, we mainly worked with Athena and QuickSight. Therefore, QuickSight submitted queries that were executed in Athena.

AMAZON IAM ACCESS

AWS Identity and Access Management is used to control who and what can access the different services and resources. It allows us to set and manage fine-grained access control for users and AWS Services. By specifying permissions for users and assigning roles to different AWS services, we can provide these with temporary security credentials for workloads that access your AWS resources while assigning them the least privileges required by them as governed by the best practices. Further, we can also manage multiple identities across a single AWS account by creating IAM Users under the root account. Following this, as depicted in the figure, we have implemented all our architecture in a single AWS account and created three IAM users for ourselves within it.

PROPOSED ARCHITECTURE



PROBLEMS:

ETL

Our dataset came from GHCNd. The organization decided to use the DLY format, where values must be read from fixed positions. We worked out these positions, as it was not documented for either values or metadata. We then encountered further difficulties splitting the data series (this comprised a whole month of data) into a more aggregable format. A better representation of this was found in our course's cluster. We decided to work with this data format first. However, we also provided a script that shows how the DLY dataset can be transformed into the SFU cluster's CSV format.

Our analyses and comparisons of both datasets helped us derive a more convenient structure for our further research on Canadian weather data. We decided to denormalize the data by pre-joining it with station metadata. Additionally, we chose parquet as a more expressive data format that is also column-oriented and should be queried much faster than CSV or DLY. Our complete findings and decisions can be found under the project's chapter on ETL.

TECHNOLOGY - ARCHITECTURE

We experimented with different services and technologies in this project before creating the final architecture. To carry out the ETL process, we attempted batch-writing the transformed rows in DynamoDB and using a Lambda connector to query them using Athena. We also experimented with different numbers of partitions and observed the time required to query the data with them. We attempted to read the S3 buckets directly before realizing the several benefits of AWS Glue. We also explored the reasons for choosing Athena for querying as opposed to Redshift and Spectrum. All of the noted observations, experiments and reasons that lead us forward from each of these decision points are detailed here.

PARALLELIZATION

For parallelization, we partitioned our data into parquet files on the 12 observations we needed for our future analysis. The best results in terms of execution time and size of parquets - were found when we didn't partition any further. More partitioning was increasing the execution time and creating larger output data due to the added meta-data for the parquet files. Further, this type of partitioning led to optimized query execution. This handled the "bigness" of our dataset. The architecture consisted of c7g.xlarge instances (1-master node, 2-worker nodes) with 4 CPU cores. The original data was 197 MB. It took about 52s and created files with a total size of 14.5 MB here.

VISUALIZATION

To find an optimal location, we need to compare the measurement data based on their state, coordinates, and elevation. Our tool of choice should be able to visualize our results on a map like geographical maps, but also box plots and heat maps. Also, we want to compare measurements over time. The loading should be fast, and the querying of data should be easily understandable. We don't want to implement a visualization solution from scratch and configure too much.

By comparing all the pros and cons concerning our problem statement, we decided to use Amazon QuickSight. A connection to other web services takes less effort, QuickSight is well-documented, and we will rely on only one set of products: Amazon.

For better loading times, we denormalized our dataset. But still, we are convinced that a higher grade of denormalization can lead to even better times. Leading to fewer computations and lower costs. We have stated this in more detail here.

RESULTS:

WIND

For Canada, there is almost no recording of thunder (only within a time range in the 1900s for NL) and no recording of hurricanes and tornadoes at all. So, we had to reduce our analyses to the primary observations and the wind speed only.

Therefore, we researched…

- the distribution of wind data per location
- the number of values over time
- the observation's values over time from 2015-2021
- …and compared these values on a geographical heatmap of Canada

All our results can be found here.

SNOW

A couple of analyses and visualizations were performed to study snow data and draw meaningful conclusions.

- Analysis of snow-depth, TMIN, TAVG and TMAX
- Analysis of snow depth from 2015 to 2021
- Snow-Wind tradeoff and Snow-Precipitation tradeoff for years 2015-2021

All our results can be found here.

PRECIPITATION

All of the results can be seen here.

CUMULATIVE RESULT

PROJECT SUMMARY:

We structured the directories of our GitHub project according to the given project summary for easier grading: https://github.sfu.ca/jfh7/datastorm

- Getting the data: 1
- ETL: 3
- Problem: 2
- Algorithmic work: 2
- Bigness/parallelization: 4
- UI/Visualization: 3
- Technologies: 5