Impact of heat waves on electricity consumption and outages

Diana Rodenberger

Manuel Moreno

Olivier Zimmer

# 

[Previous comments](#_Previous_comments_and)

[Project Description](#h.k2x2l2ds4rdb)

[Purpose](#h.8v3tu6ixyy3l)

[Goals](#h.rmrxxb6dn1li)

[Limitations](#h.yk6pb2vmbz49)

[Architecture](#h.vusnqdve0090)

[Google BigQuery and Cloud storage](#h.zeytjctlguwm)

[Data Ingestion and Processing](#h.f1xdpbwxmm8g)

[Extraction & Transformation](#h.vw3jiau7zu0m)

[1. Electricity Data](#h.mp2peyvaw7kp)

[2. Weather data](#h.nde2j4jmbc1x)

[Data serving](#h.6ob0hyte5b)

[Future requirements](#h.hr6q2io7mw7v)

# 

# 

# Previous comments and Answers

\* The report is missing details about the high-level architecture of

the system, technologies to be employed, integration issues,

visualization aspects.

\* Please include all these in the next report taking the perspective

of a big data analytics project where injection, storage, processing,

serving of data is well-defined and justified, along with any scaling

issues (and limitations issues).

1. We chose to use Google BigQuery because it allowed us to collaborate and work on our project simultaneously and seamlessly. It also pairs very nicely with Google's Compute Cloud. It is something of a hybrid between a data store and a storage platform and, thus, cannot be pigeon-holed as easily as say, hadoop and hdfs. Furthermore, it is managed completely by Google which makes it hard for us to address some of the questions you posed previously (such as scalability, low-level architecture).

2. Because BigQuery interfaces quite nicely with Python, we have decided to perform most of our analysis using this rather than a solution such as Spark. While Spark can interface with BigQuery, this capability seems more like a way of allowing people to mix traditional data with BigQuery data rather than something that brings value to a BigQuery-only dataset.

3. Our predictor data (weather forecasts, power consumption data) is not is available in a streaming manner. Our plan is to retrieve weather forecasts on a daily basis and utilize our models to predict the outcomes, which would then be visualized in Tableau. We understand this is simply a fast batch model rather than a strictly-streaming method.

# 

# Project Description

## Purpose

The purpose of this project is to utilize historical weather, power consumption and power outage data to model the relationships between temperature and the following outcomes:

1. Cost of electricity across different sectors
2. Power outages occurring due to extreme weather

Using the aforementioned model, this project aims to leverage weather forecast data to provide semi-realtime predictions of the price and likelihood of power outages, thus allowing users to make better informed decisions.

## Goals

* Impact of heat waves on electricity consumption and on power outages.
* Characteristics above which a heat wave is likely to be harmful to the electrical grid.
* Relations between temperate and the price of power across different sectors.
* Probabilistic risk assessment of future outages and prices based on weather.

## Limitations

Because this project will be using publicly-available datasets, it suffers limitations inherent with the data sources. These include:

* Lack of spatial and temporal granularity for power data sets
* Inaccurate/Ambiguous geographical descriptions for outage data set
* Possible gaps in the outage data set

Future iterations of this project should strive to gather this type of data from the energy suppliers themselves as a way of mitigating these issues.

# Architecture

**Storage System**

Data Warehouse: BigQuery

**Data Definition**: csv source files

**Batch Processing Platforms**: Python & BigQuery

**Stream Processing Platforms**: ???

**Compute Platform**

Cluster High Perfomance: Google Cloud

**Data Serving**

BigQuery into

Tableau

**Query/Exploration**: SQL (BigQuery)

**Data Cleaning:** OpenRefine

**Data Ingest**

ETL using Python and R

Due to the collaboration needs of the project team, BigQuery was selected as the data platform for this project. This platform, paired with Google’s compute cluster fulfill the needs of this project while removing the need for management of a storage architecture and allowing fully-distributed access seamlessly.

## Google BigQuery and Cloud storage

Google BigQuery built on the Dremel infrastructure - [see the dremel paper](http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36632.pdf)

Data Lake hosted on Google Cloud storage

* Consistent, large scale storage (64 TB per disk)
* Low cost
* Archiving and disaster recovery
* Available in SSD -> optimize the speed of I/Os
* Seamless scalability

# Data Ingestion and Processing

## Extraction & Transformation

### 1. Electricity Data

* Consumption, Generation and Price data
  + Using the Energy Information Administration’s Open Data API. Due to the organization of the data, data collection was iterated as follows:
    - For each of the 51 regions (50 states plus DC)
      * Price and Demand data was collected for each of the 4 sectors at the state level
      * Generation data was collected at the state level
  + Period field was split into two fields:
    - Year (Period[:4])
    - Month (Period[4:])
* Outage data
  + Using Inside Energy’s grid disruption dataset is a compilation of annual power outage data from the Department of Energy.
    - This data encompasses disruptions from years 2000 to 2014.
    - The primary source of this data is OE-417 regulatory filings to the Department of Energy.
    - Different levels of geographical granularity due to unstandardized reporting requirements
    - Possible reporting gaps due to jurisdictional requirements and manual processes
  + The following fields will require cleaning:
    - EvenDescription - Identify advisories
    - DateEventBegan - Extract Month, validate year
    - GeographicAreas - Extract States affected
    - Tags - Identify weather outages

### 2. Weather data

* Reading the .dly file format specific to the Global Historical Climatology Network (GHCN) to be converted in a readable format .csv
* Compiling 1218 files from different weather stations
* Data wrangling to get relevant variables (TMAX and TMIN) and removing other variables
* Cleaning data (dealing with N/A values, here coded as -9,999)
* Other transformations: monthly average, monthly maximum…
* Match the right schema prior to loading on Google Big Query.

# Data serving

This project will utilize BigQuery’s built-in data serving capabilities in two manners:

* First, the data will be served to the processing server using the Python BigQuery Client Libraries. Where a the data models will be created. As a separate task, the processing server will gather weather forecast data using SOAP and will perform predictions using the models creating during the batch process.
* Secondly, the project team will leverage Tableau’s built-in BigQuery driver to gather the relevant prediction data to create a visualization indicating expected cost of power and possibility of outages for each of the regions in question.

# Future requirements

BigQuery offers seamless storage scalability and, thus, will only need to be managed from a budgeting perspective. Due to the relatively-modest size of the different data sets involved in this project, we do not expect processing requirements to increase to a significant degree in the near future (<10 years). That said, Google Cloud offers seamless compute capacity increases if such a need arises.

Updating the batch data sets will allow the models to be refitted to incorporate more up-to-date data and increase the accuracy of their predictions. There are two ways of updating the batch data:

1. **Drop and load:** This approach relies in the collection of full data sets on a recurring basis and involves the complete reconstruction of the data tables using the initializing procedures. This approach simplifies the loading process because no coherency checks need to be performed.
2. **Deltas:** The preferred method of updates, this approach will insert new data into the tables without updating previously-loaded data.