BIG DATA QUERY PROJECT

With Network Innovations

Schulich School of Engineering, University of Calgary

PROJECT MANAGER:
Madeline Mazurek

FRONT-END DEVELOPERS:
Ethan Card
Mushtaba Al Yasseen

BACK-END DEVELOPERS:
Parbir Lehal
Jacob Lansang
Dylan Mah

PROJECT SPONSOR:
Alex Shaharudin

ACADEMIC ADVISOR:
Leanne Wu

TEACHING ASSISTANT:
Manuel Lopez

ABSTRACT

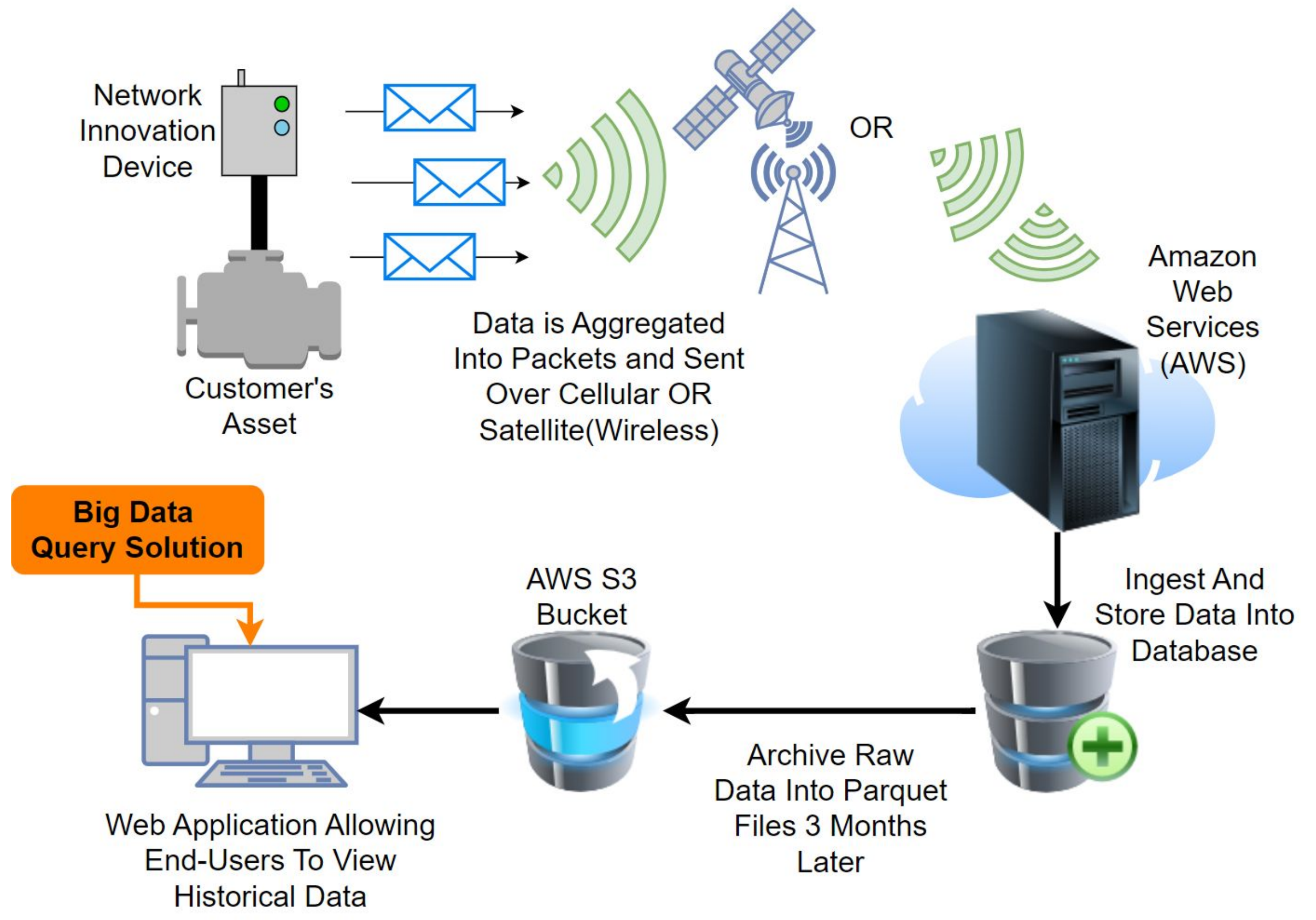
- The Big Data Query project provides a custom web-based solution for a unique big data problem. Network Innovations has many years worth of archived network data, recorded from customer devices such as engines, water pumps, fishing vessels, and more. Considering nearly 2 billion recorded data entries adding up to hundreds of gigabytes of data, the developed application can read and display desired data for any customer use case in a remarkably fast time. Queried results can be filtered and sorted dynamically based on any possible parameters, allowing for efficient and adaptable data investigation. A custom-built export feature allows for uncomplicated data delivery to clients.

- An iterative design and development process was implemented, allowing for the gradual improvement of performance metrics, until the benchmark goal of a regular query time under 30 seconds was achieved.

INTRODUCTION

- Big data management is a challenging issue that is becoming more prominent by the year in the software industry.
- As cloud storage solutions grow in popularity and availability, companies are increasingly encountering issues with software that cannot scale to handle bloated databases or huge archives.
- Network Innovations works with customer data from a large array of devices, providing web-based access to this data.
- After three months customer data is sent to AWS S3, and can no longer be accessed.
- Our team is tasked with building a web-based solution that provides easy and fast access to this huge data archive.

SYSTEM CONTEXT DIAGRAM


Network Innovation Device
Customer's Asset
Data is Aggregated Into Packets and Sent Over Cellular OR Satellite(Wireless)
OR
Amazon Web Services (AWS)
Ingest And Store Data Into Database
Archive Raw Data Into Parquet Files 3 Months Later
AWS S3 Bucket
Big Data Query Solution
Web Application Allowing End-Users To View Historical Data

OBJECTIVE AND SCOPE

The project objective was to create an easy-to-use custom web application to retrieve, filter, and display a user specified subset of data from an AWS S3 bucket containing raw data in parquet files. One new file is stored each day, spanning years of historical data, with a count of nearly 2 billion recorded entries (hundreds of gigabytes!).

Most importantly, the server-side queries are required to run very fast relative to this large amount of data. The primary goal is for results to return in **30 seconds or less for a 1-3 month time period** (which represents the average load on the system for a normal use case). Each fetch query is required to evaluate data for a single device over a user-specified date range. From here, users must be able to interact with data in a tabular format, with the ability to apply custom filters to any column. Finally, data results must be easily exportable to a .csv format when requested.

METHODS AND TECHNOLOGIES

- Development started by building an MVP product using a simple React front-end and Flask back-end using Pandas and a small set of sample data. With the MVP created, development followed a cycle of: Test -> Explore changes & alternatives -> Improve solution -> Repeat
- A full set of mock data was created using the provided sample and a custom Python script.

Various back-end solutions were explored to improve query times and memory optimization:

- **Pandas** - Easy to use library, but has limitations with speed and scalability
- **Dask** - Very good at handling large datasets but poor performance for frontend integration
- **Polars** - Optimal balance between speed and scalability when implemented correctly

The technologies included in our final solution include Python, Pandas, and Polars in a Flask back-end, with a React.js front-end. Code is hosted on an AWS EC2 cloud computing instance, deployed using Docker, and is built to access parquet data in an AWS S3 storage bucket.

Flask
pandas
React
docker
Amazon S3
Amazon ECS

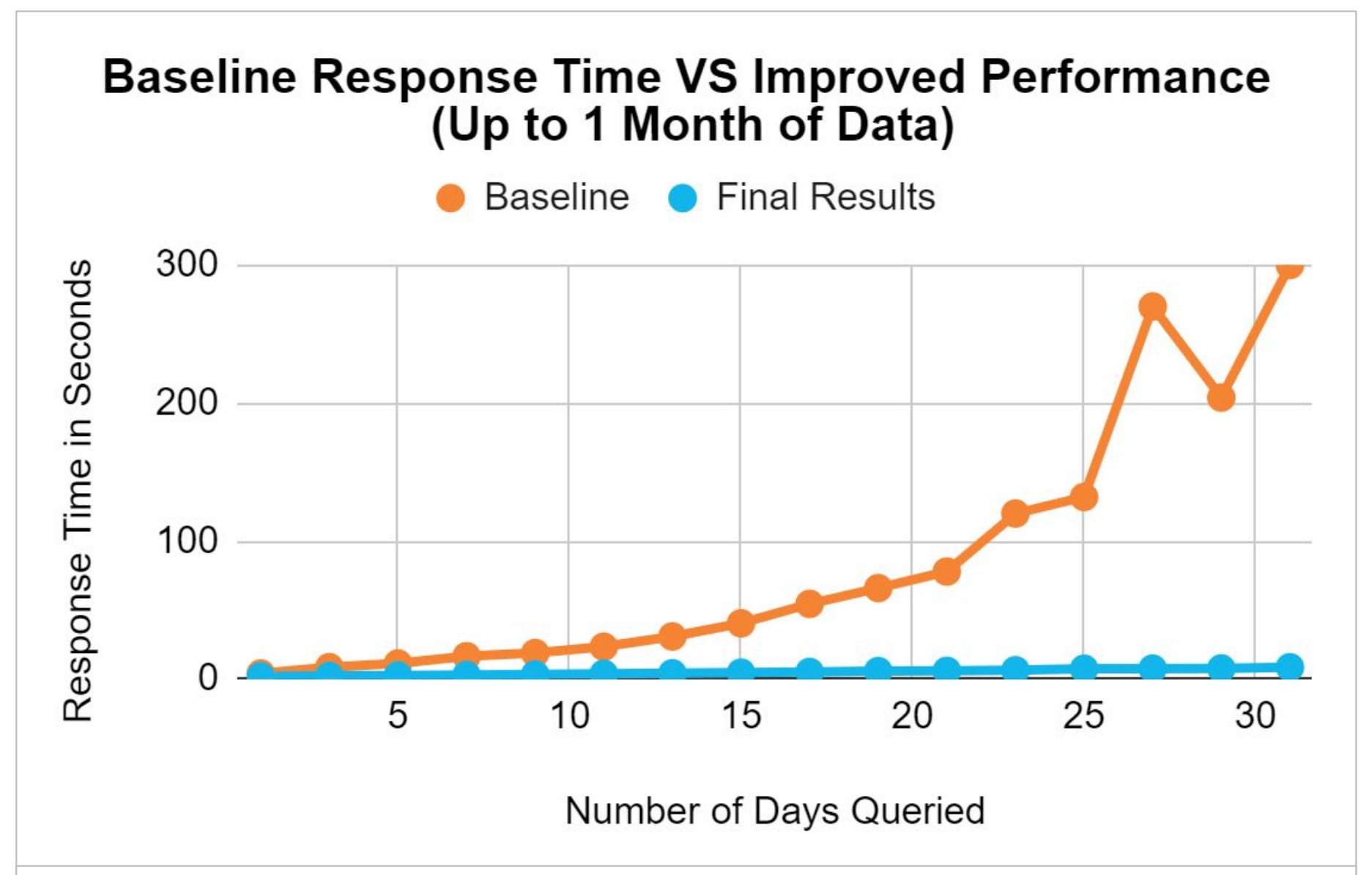
TESTING AND RESULTS

After creating the project MVP, initial testing results showed many limitations. Query response time for 1 month (31 days) of data was approximately 5 minutes (300 seconds). Further, the application was only able to support up to 2 months of data at a time before encountering memory issues that caused the application to fail.

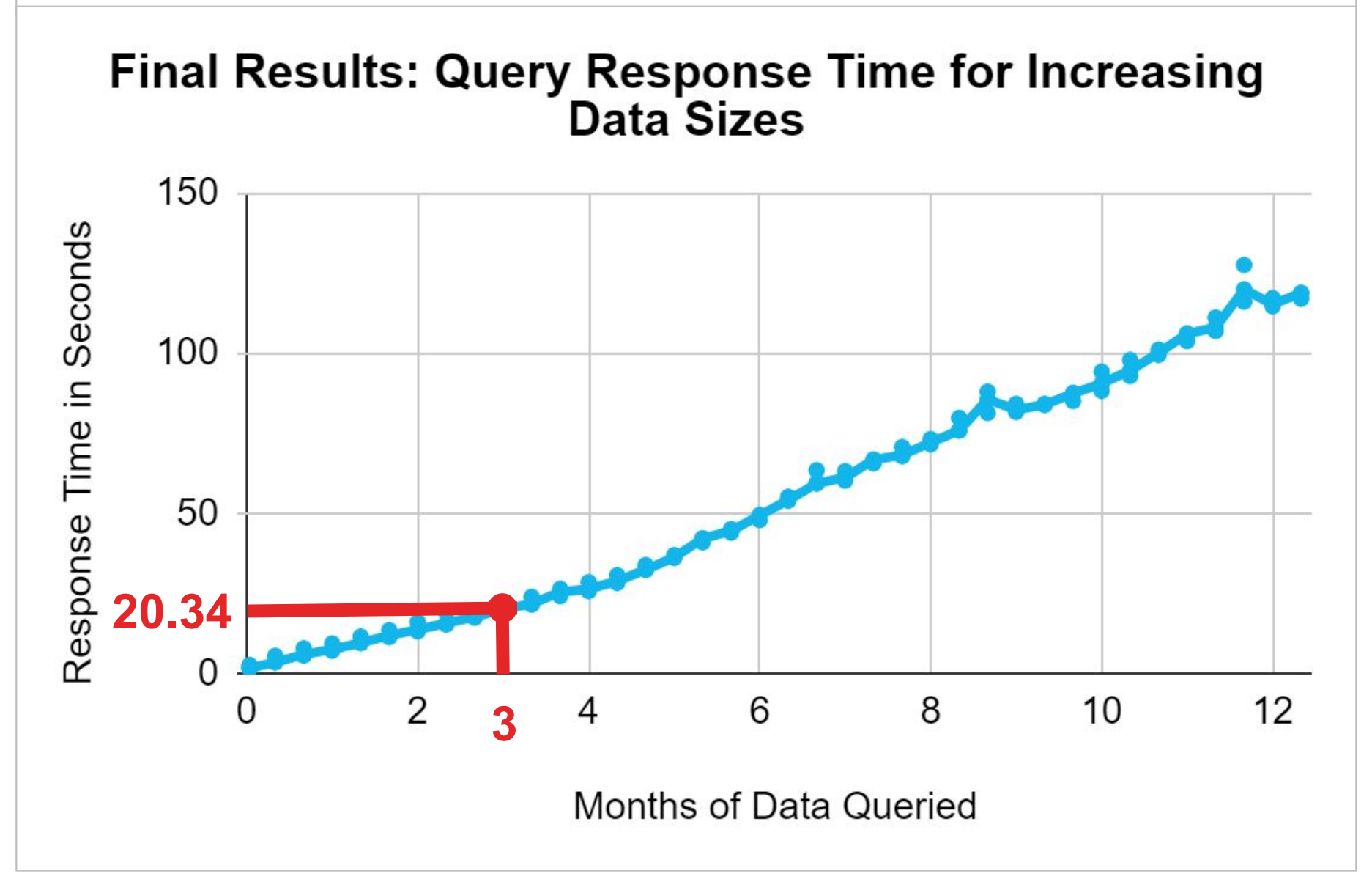
Months of testing and iterative improvements followed the initial results, involving the introduction of various new tools and techniques (Dask, Polars, EC2, and widespread code restructuring).

These changes resulted in large performance improvements, with the response time for 1 month of data dropping to ~9 seconds. The benchmark goal of 3 months of data in less than 30 seconds has been surpassed, now taking only ~20 seconds on average!

Comparing MVP Performance To Final Results:



Performance of Final Results up to 1 Year of Data:



CONCLUSION

Our team has successfully created a custom web-based solution that meets all of the needs of Network Innovations while overcoming the numerous challenges of big data management.

An iterative development cycle was executed within our team, resulting in large performance improvements over time.

The primary goal of returning 1-3 months worth of data in under 30 seconds was surpassed, with the application being able to fetch **3 months of requested data in an average time of ~20 seconds**.