Transforming and Caching Entries from Massive, Online Databases using Apache Spark and MongoDB

Mark E. McDermott

Volgenau School of Engineering
George Mason University
Fairfax, VA 22030
Email: mmcderm1@gmu.edu

Jose A. Velasquez-Principe

Volgenau School of Engineering
George Mason University
Fairfax, VA 22030
Email: jvelasq@gmu.edu

***Abstract*—Online databases with user generated content are increasingly common. If the number of users of such a database is large then the amount of content can quickly grow to be very large. This means that it is impractical and expensive to allow users to access the database in its entirety. To work around this many sites provide feature-rich search and query APIs that allow users to download only the portions of the database that they require. After downloading the data the users then processes it using their local computing resources. A very common operation to be performed by users is to perform preprocessing and cleaning of the data since, due the distributed sources of the data, it is likely to be of varying quality. This represents duplication of work that could be captured back into the public database. We present an architecture based on Apache Spark for performing this cleanup near the source of the data. We also present the creation of a cache of cleaned data so that other users can benefit from the work and analysis already performed. We demonstrate its effectiveness using a real-world example: We perform spectral filtering of recordings from the Xeno Canto database of bird-songs and then cache the filtered results.**

I. Introduction

One very interesting development to come from the cloud computing and data science explosion is crowd sourced data bases. These databases can grow very large as the work it takes to curate the databases is distributed around the globe. This distributed nature presents some challenges though.

One issue is that quality of individual entries in a database can vary since no single source or curator was involved in generating the entries. Some data providers, such as Kaggle [1], solve this by allowing users to rate the quality of datasets. However this approach, while it does provide useful information to potential users, does not do anything to improve quality. A dataset uploaded to a service such as Kaggle can only be updated by the user that uploaded it so improving the quality of a dataset after it is created can only be done by the original user. This reduces the benefits of distributed data curation.

Another challenge is that the meta data associated with particular data points can vary greatly based on the individual contributor's capacity to gather the data. This could be solved by careful curation of the submissions to ensure that each data point is properly labeled with metadata but this again reduces the benefits of crowd sourcing data collection.

We found a particular example on Kaggle that highlights this issue and inspired this project. A user of the very large, user generated database of bird call recordings - Xeno Cano [2] - uploaded a carefully curated subset of the Xeno Canto library to Kaggle so that it was ready for data science workflows [3]. This represents significant work that would have to duplicated in the future since the Xeno Canto database continues to grow as user's submit new recordings. The recordings are of many different lengths and have different levels of background noise, thus necessitating the cleanup step prior to use for data science.

In this paper we present an architecture that allows for crowd sourcing the curation efforts by caching cleaned up data points using a database based on flexible schema. This allows user's to perform the necessary data manipulation efforts and save the results for other's to use without requiring updating the source database's schema. We pair this cache with Apache Spark to allow for distributed processing of user's requests.

II. Technology Stack

Figure 1 shows the technologies we integrated for this project, each of which is introduced in this section. In the next section we show in detail how these technologies can be used together to create a cache of the Xeno Canto database.

A. MongoDB

MongoDB is a cross-platform program that allow users to create databases for very large data and is used in abundance of modern application. MongoDB specifically in this project will be used to create a database that will allow users to cache and query. This will also allow the project to have seemless connection from jupyter notebook to database. Also MongoDB will allow for very large files with millions of records of data to be handled with ease because of the structure of this application.

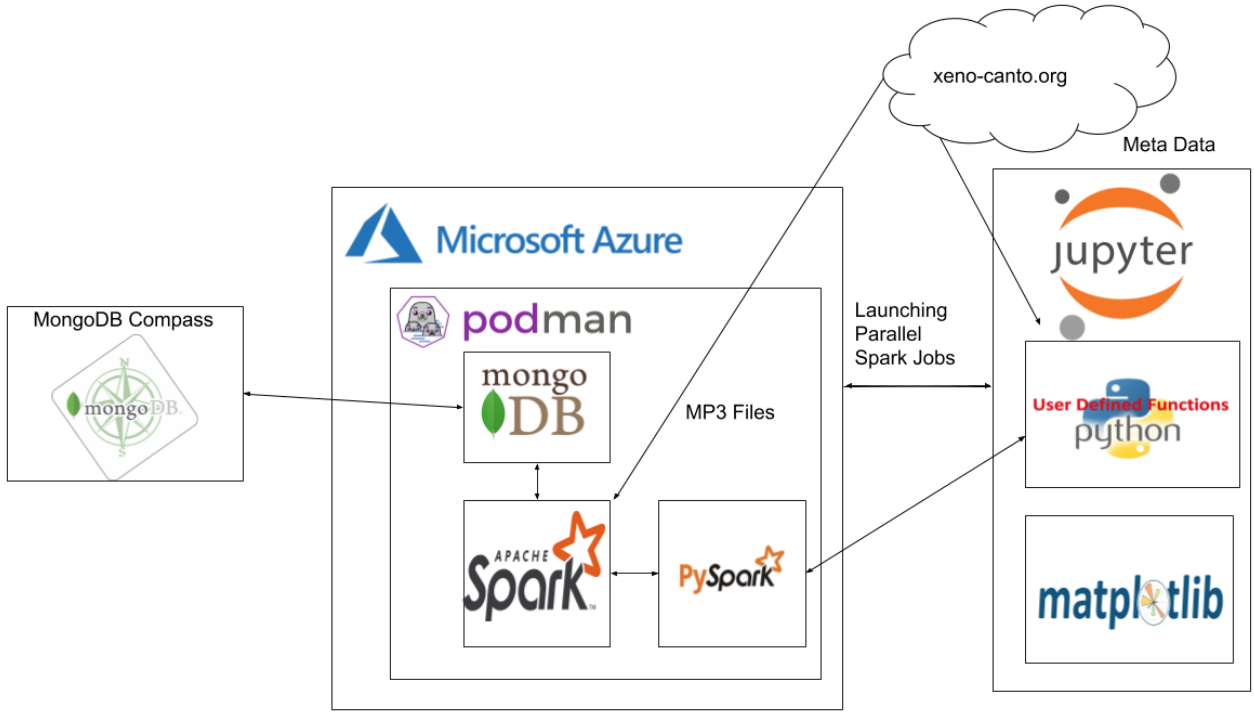

Fig. 1. Technology stack used in this project.

B. Apache Spark

Apache Spark is a data processing framework that allows users to perform operations on data sets that are very large. Apache Spark is a key application that allows big data and machine learning to be incorporated in a user-friendly application. In this project Apache Spark will allow operations such as spark user defined functions and parallelization of data to be incorporated into the project. Apache spark also allows for very tedious task to filter and examine data to be done seamlessly and is a huge contributor to this project.

C. Python

Python is a programming language that allows us to incorporate functions and predefined libraries to tackle use case at hand. Python will be used throughout the project because it very ease to incorporate Spark and python also the libraries in python for example numpy allow for solution to challenging problems. Finally Python allows for an ease of corporation of all programs to be used in this program.

D. Jupyter Notebooks

Jupyter Notebook is a user-friendly open source application that operates as a GUI or ”Notebook App” that allows users to write code similar to a text-editor and compiler. Juypter Notebook is a key foundation in this project that allows us to incorporate allow these programs previously mentioned to be used in practice and allow us to understand step by step how the solution is be used and incorporated.

E. OCI Containers

F. Microsoft Azure

Microsoft Azure is a subscription based application that allows users to create VM also know as virtual machines. These virtual machines allow users to save space in there individual computer and run very large and intensive applications through these virtual machines. Allowing users to have appropriate tools to complete the task at hand. Virtual machines are incorporated in this project by using the VM to be the bases for how everything is run and the operating system for this project.

III. Architecture

The previously mentioned technology stack is all the applications needed to complete the task at hand. Thankfully all the applications used were easily incorporated onto each other because of there design and layout. In this section we will explain how these application were incorporated into each other and how these applications collaborate together. Starting with the virtual machine that was created with the help of Microsoft Azure, allowed for the platform to be created and a base to begin using these other application. Secondly the Podman that are containers that allow these applications to be runned in the background . Also allows these applications to runned seamlessly, because all these application can be launched though a command prompt. Furthermore the database was created with MongoDB because MongoDB is a user-friendly application that works seamlessly with Apache Spark to cache and query to a very large database as needed. Next

the Apache Spark is used because this allows the data to be manipulated because Apache Spark is a efficient way to process large data sets, and consequently allowing for greater manipulation of the data. Which is where Juypter Notebook comes in as both a text editor and compiler for all these applications as it supports them and is a user-friendly GUI. Finally Python, Numpy, and Matplotlib are libraries were these manipulations can be done with processes that are pre-built and user defined functions to manipulate the data.

IV. Example Use Case

A. Xeno Canto Bird Call Database

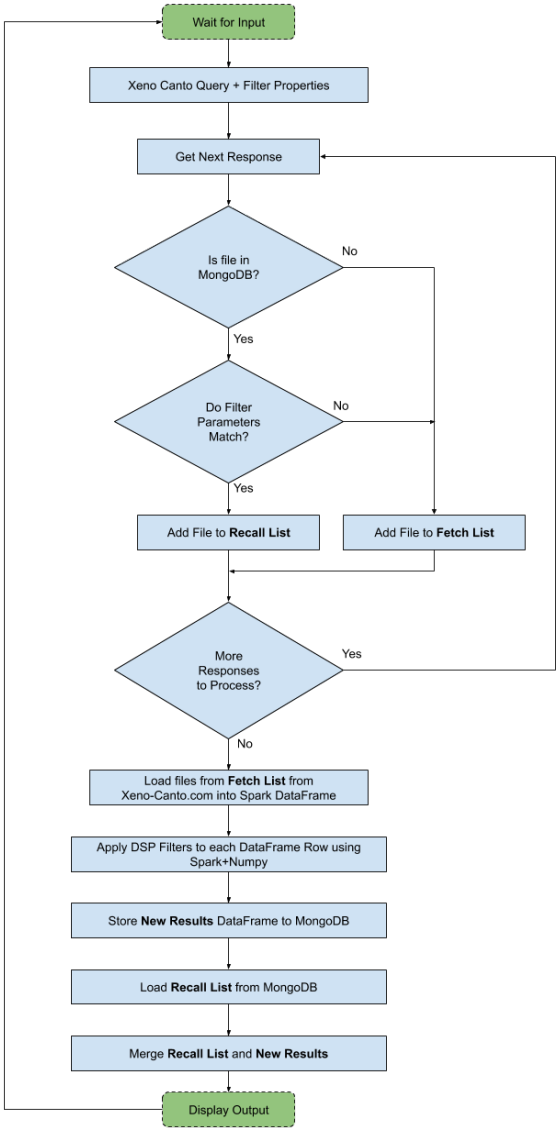

Fig. 2. Flow chart of the process used to merge querries from the cache and from the online database and then perform parallel processing using Apache Spark.

B. Digital Filtering

C. User Defined Functions

D. Viewing Results

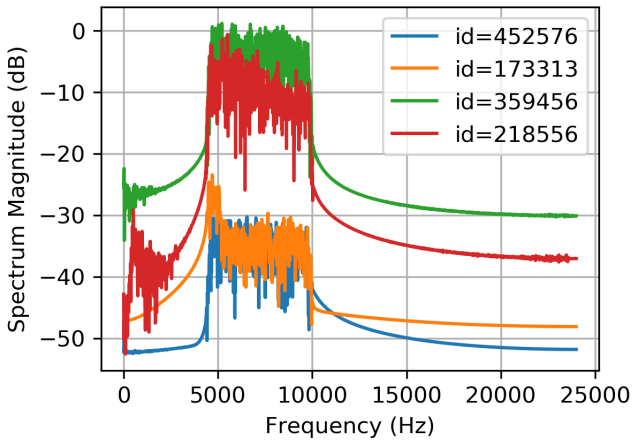

Fig. 3. Results generated by running two separate Apache Spark jobs: one to download and apply a bandpass filter to four different recordings from xeno-canto.org in parallel, and another to compute the magnitude spectra of the recordings.

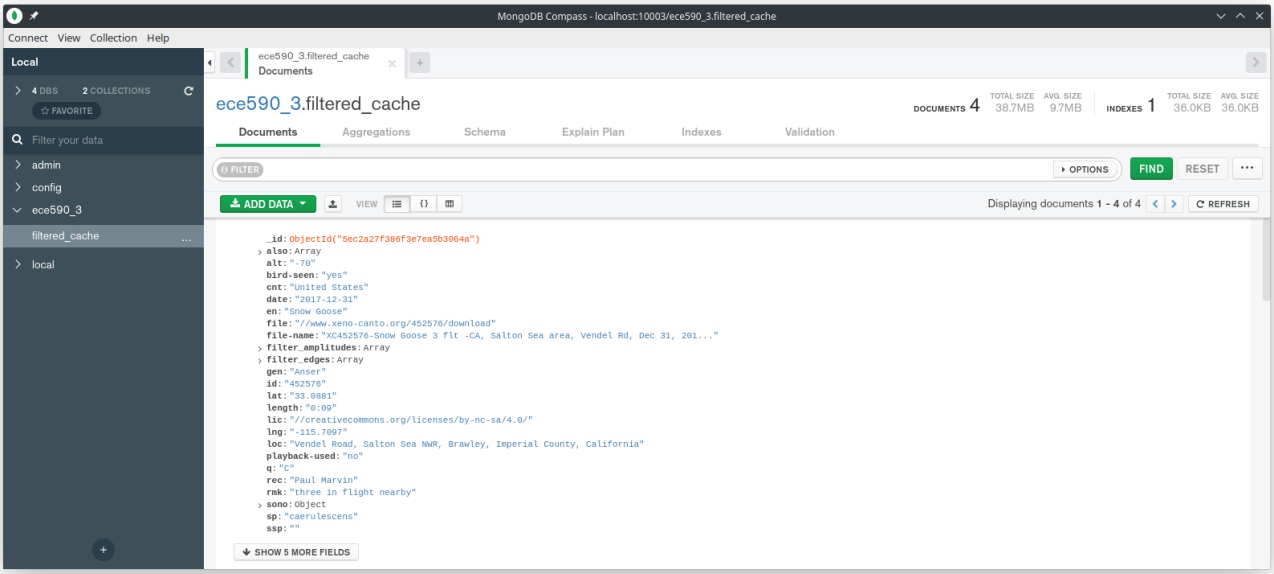

Fig. 4. MongoDB Compass Application for viewing and managing MongoDB databases.

V. Conclusion

In conclusion based on the projects result we can conclude that in fact the project operated as intended as seen in the data and graphs in the report. The scope of this project was to incorporate the techniques learned in ECE 499 and combine those techniques with Machine Learning and signal processing. Although this was just incorporated in something as simple as bird sounds these practices can be scaled onto data that is more rigorous and/or import. Finally this report hopes to inform the audience of the bridge between Signal Processing and Machine Learning and how they can be applied to uses cases.

References

[1] Kaggle, "Kaggle: Creating a dataset," https://www.kaggle.com/docs/datasets#creating-a-dataset, last visited on 2020/5/17.
[2] XenoCanto, "Kaggle: Sharing bird sounds from around the world," https://www.xeno-canto.org/, last visited on 2020/5/17.
[3] F. de Abreu e Lima, "Bird songs from europe (xeno-canto)," https://www.kaggle.com/monogenea/birdsongs-from-europe, last visited on 2020/5/17.