Sarah Martin
Northwestern University, School of Professional Studies
Foundations of Data Engineering, Professor Lance Levenson
March 11, 2019

Project 2: Systems Design and Specification for a Movie Recommender Engine

Section 1: Management Overview and Recommendations

The task at hand is to define general requirements for the technical system and related business processes for a new movie recommender engine. This document builds upon the previous document and focuses on specifying a set of viable products, platforms, and solutions to fully create a working production system that will scale to accommodate hundreds of thousands of potential users.

Section 2: Data and Database Software

Section 2.1: IMDb and MovieLens.org Data

Data from at least two of the data sources –IMDb and MovieLens.org -- for this recommender engine is relational in nature. Based on a preliminary understanding the business case, SQLite is likely a prime candidate for our relational database needs. It is open source, free, and, like virtually all relational databases, supports indexing to assist with query runtimes. In addition, the curated IMDbPy package which assists with retrieving the IMDb data, unpacking it, and storing it in database solutions works with SQLite.

Our system needs to allow for regularly scheduled updates of the data from IMDb and GroupLens at roughly 10-day intervals. IMDb's publicly available data is updated frequently and consistently so, presumably, every 10 days, we will be pulling new versions of the IMDb files and updating our SQLite database. The updates of the IMDb data will require additional processing power. Because SQLite is a cloud-based software, it can more easily and programmatically scale to

accommodate additional processing needs concurrently. Once our engine is launched, the precise cadence of IMDb data acquisition should be periodically reassessed leveraging input from the modelers as well as business decision makers.

Alternatively, much of the MovieLens data is not updated very frequently and updates do not abide by a predetermined schedule. As such, our data update every 10 days should first check to ensure that new data is posted by GroupLens. If so, a data acquisition and loading process should commence. However, most of our scheduled data updates will determine that no new MovieLens data is available in which case no further action should be taken until the next data update 10 days later. Avoiding unnecessary data pulls and updates will save time and money for the company.

A technical resource[1] should be put in charge of developing, executing, and monitoring data updates which will include checking for new data, acquiring it, any basic re-structuring needed, and batch loading into our SQLite database. This resource must easily communicate with those managing any IMDb relationship *and* with the modelers to ensure that expectations regarding the availability of new data are in alignment.

Section 2.2: User Input Data

The last data source required for the movie recommender engine is user input. To make the product user friendly, and to leverage previous user movie preferences in model validation and periodic re-training, we should update and store existing users' movie preferences every time input is received. User input data lends itself to being stored in a document model. Recommendation generation should be a quick process and having to sift through hundreds of thousands of users' data in a relational model will increase recommendation generation runtimes. With a document model, all data from a single user is stored in one place making for faster retrieval and each time a recommendation request is made, we can easily pull that user's data along with pre-fitted models to produce the recommendations.

[1] Resource may always refer to a single individual or a team of people

MongoDB and Couchbase are both strong candidates for cloud-based open source document style enabled databases. MongoDB Atlas is the managed cloud version of MongoDB. Both solutions can be run on any of the three major cloud service providers -- AWS, Google Cloud Platform, or Microsoft Azure – and allow for easy transferring of data between the three.

MongoDB is a lot faster on small amounts of data but can sometimes struggle with scalability as the end product's user base and usage rates grow. MongoDB appears to be quicker to set up and start using from a data loading and querying perspective, though some note that Couchbase's architecture simplicity makes it easier to manage from an architecture and administration standpoint. However, if MongoDB Atlas is used, many of these architecture decisions and tasks are provided as part of the managed service. Ultimately, both should be explored as viable options to handle our user (and potentially model) data needs.

A database administrator will need to monitor our database software, track issues, perform upgrades, instantiate data checks, ensure that any changes to the structure of incoming data can be handled appropriately, and monitor read/write volume and performance over time to ensure database tuning parameters are properly set to handle our growing throughput.

Section 3: Analytics and Modeling Software

Once the relational data has been loaded into SQLite, it will need to be processed, manipulated, transformed, and available for analysis. Given the volume of data we may need to accommodate and the non-constant periodic increase in computing and analysis requirements (every ten days), leveraging a big data solution on the cloud to act as a warehousing tool is a viable option for our analytical needs. The Apache Hadoop ecosystem provides a comprehensive open source solution to our warehousing and analytical needs. Hive and Spark in addition to Hadoop's HDFS offer additional tools to assist with querying, managing, and facilitating analysis on the data. Apache Spark can also access user data from MongoDB allowing us to consolidate data from across all sources for analysis. It can also interface with

TensorFlow, Keras, and Caffe, prominent open source frameworks for AI learning. These three neural network frameworks can also be accessed as modules in Python and can be supported by the 3 leading public cloud providers – Amazon's AWS, the Google Cloud Platform, and Microsoft Azure.

Once the most recently acquired data in SQLite has been processed and appended to existing data in Hadoop's HDFS and new IMDb and/or MovieLens.org data is available, we can overwrite existing data in the SQLite database solution since Hadoop will be our primary active storage solution for all analytical work.

The modelers or those who work closely with them in modeling support roles should be in charge of periodically monitoring new data arriving in Hadoop for statistical consistency and conducting frequent model validation and tuning to ensure our recommendation engine continually maintains or increases its efficacy.

Section 4: Computing and Communications Systems

Thus, both document database solutions are able to avoid getting locked in to a single cloud service provider.

Because data acquisition, loading, processing, and modeling checks and updates will be intermittent on a 10-day schedule and because load estimation for an entirely new product is often inaccurate and highly unpredictable, an elastic cloud computing infrastructure will be critical in allowing us to respond periodic increases in computing needs and changes in load more quickly and efficiently. In addition, frequently monitoring throughput is necessary to prepare for growth in the product and anticipate and remedy scaling challenges before they impact the system.

The storage and analytical activities discussed will be easily scalable for periods of higher computing if hosted and leveraged on one of the three leading public clouds. The Google Cloud Platform would be a good place to start since many of the open source solutions mentioned above were originally developed by Google. Amazon's AWS is also a roughly equitable contender. What makes our

storage, warehousing, and analytical solutions so attractive is the ability to easily switch between any of the 3 leading public cloud providers allowing us to experiment and determine through firsthand experience, which is best suited to our needs.

In thinking about the flow of data between systems, when encoding is required, Apache Avro is a robust and compelling open source binary encoding option specifically designed for Hadoop use cases. Schemas are required to tell different version of code how to decode and encode various structural versions of historical data. However, Avro makes schema construction easy by supporting dynamically generated schema directly from data tables. Avro comes with RPC support included, the Avro RPC protocol. The backward and forward compatibility properties of an RPC scheme are inherited from Avro's encoding format. In network calls with a web application taking in user input and sending out movie recommendations upon request, a RESTful API with JSON encoding may be more appropriate. Where an encoding format is used, runtime checks should be put in place to catch bugs in the data and its encoding.

Works Cited

Burt, Jeffrey. “Battle Of The Document Databases.” The Next Platform -, The Next Platform, 22 Jan. 2019, www.nextplatform.com/2019/01/22/battle-of-the-document-databases/.

Designing Data-Intensive Applications: the Big Ideas behind Reliable, Scalable, and Maintainable Systems.” Martin Kleppmann, O'Reilly Media, 2018, pp. 3–197.

Garbade Feed, Michael J. “Top 8 Open Source AI Technologies in Machine Learning.” Opensource.com, Opensource.com, 15 May 2018, opensource.com/article/18/5/top-8-open-source-ai-technologies-machine-learning.

Hummel, Guy, and Jeremy Cook. “What Are the Benefits of Machine Learning in the Cloud?” Cloud Academy, Cloud Academy, 23 Aug. 2018, cloudacademy.com/blog/what-are-the-benefits-of-machine-learning-in-the-cloud/.

“IMDb Datasets.” IMDb, IMDb.com, www.imdb.com/interfaces/.

Purkait, Niloy. “How to Train Your Neural Networks in Parallel with Keras and Apache Spark.” Towards Data Science, Towards Data Science, 14 Oct. 2018, towardsdatascience.com/how-to-train-your-neural-networks-in-parallel-with-keras-and-apache-spark-ea8a3f48cae6.

Tellicherry, Tharika. “11 Open-Source Frameworks for AI and Machine Learning Models - DZone AI.” Dzone.com, DZone, 5 Apr. 2018, dzone.com/articles/11-open-source-frameworks-for-ai-and-machine-learn.