Distributed Computing with Spark: Analyzing Reddit and Netflix Data

Introduction

In this project we will be exploring using Spark to process data in a distributed manner. With data sets getting larger and larger, it may not be enough to process data with a single system. With Spark we'll be able to automatically speed up processing time by parallelizing data processing among multiple processes/systems. Be sure to read the whole project description before starting.

Background: Apache Spark

Apache Spark is an open-source data processing framework that allows for processing large-scale data sets. Spark provides an intuitive interface to scale data processing while providing flexible data parallelism as well as fault tolerance. Compared to other distributed data processing frameworks such as Hadoop, Spark provides a powerful caching layer and allows for in-memory data processing which can speed up computation and support real-time data processing. Because of these features, Spark has become one of the most prominent frameworks in today's field of large-scale data processing and data science.

In its basic form, Spark's architecture is centered around the *resilient distributed dataset* (RDD). Simply put, an RDD is ready only (immutable) dataset. Processing of an RDD consists of transformations that take the data and create new RDDs that can further be transformed. The use of transformations allows Spark to model data processing as a *directed acyclic graph* (DAG). With a DAG, Spark can maximize data parallelism not only for a single transformation, but across multiple transformations. You can think of this as Spark being able to automatically parallelize data processing by creating a pipeline of transformation stages where multiple nodes can carry out transformations simultaneously in each stage.

Over the years, Spark has expanded its APIs to provide programmers more flexibility with how they manipulate data by adding *DataFrames* and *Datasets* along side RDDs. Spark has also expanded its data processing capabilities by adding libraries such as *Spark SQL*, which provides more a comprehensive interface for processing structured data, *Spark MLlib*, which enables machine learning to be easily applied to a data processing pipeline, *Spark GraphX*, which provides a framework for distributed graph processing, and finally, *Spark Streaming*, which allows for near real-time data processing. In this project, you will need to familiarize yourself on the basics of Spark and understand how to use its interfaces (in Java) as you will be building Spark programs do some of your own data processing and analysis.

Helpful References:

- **Apache Spark -** https://en.wikipedia.org/wiki/Apache_Spark
- **RDDs** - https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm
- **Spark Architecture -** https://www.edureka.co/blog/spark-architecture/
- **Spark Parallel Processing -** https://www.simplilearn.com/spark-parallel-processing-tutorial
- **Spark Examples -** https://spark.apache.org/examples.html (See Java Examples)
- **Spark Java API -** https://spark.apache.org/docs/latest/api/java/index.html

1 Analyzing Reddit Data

Intro to Reddit

Reddit is a popular social news website where content is organized into user created boards called "subreddits". Users can add posts to subreddits. Users can subscribe to subreddits to see new posts posted within the subreddit on their homepage. In its basic form, the homepage of Reddit would present a curated feed of popular posts among the subreddits the user is subscribed to. Each post can be "upvoted" or "downvoted". A post can be commented on and each comment can also be "upvoted" or "downvoted". An upvote and downvote can be compared to "like" and "dislike" on Facebook. A user can"upvote" a post if they believe it is contributes to the current "conversation". Likewise, users can "downvote" a post if they believe it does not contribute to the current "conversation". Through the use of upvoting and downvoting mechanisms, users help Reddit algorithms rank and determine quality content that users would be interested in.

Learn more about Reddit here: https://en.wikipedia.org/wiki/Reddit

Reddit as a Basis for Social Media Research

Within social media, certain content can be posted multiple times. In the case of Reddit, similar content can be posted multiple times in more than one subreddit. In the context of Reddit, when some content is resubmitted in another post, it is called a "repost". Analysis of reposts help provide a means in understanding social media content placement. Even though two users can submit two different posts using the same content, each post could perform differently. It becomes apparent that there is a multitude of factors that contribute to the performance of a post besides the content such as: the number of user subscribed to the subreddit, how attractive the post title is, the time of day the post was uploaded, and so on. With the large user base of Reddit along with Reddit's board system of subreddits providing simple modeling of social communities, it has been seen to be a good platform to research social media interactions, patterns, and phenomena.

You can learn more about this kind of research here: http://i.stanford.edu/ julian/pdfs/icwsm13.pdf

Goal:

The goal of this section is to create Spark programs to process the Reddit repost data in a way that will help us do some data analysis of our own.

Data Format

Every row within the Reddit data set represents a repost of a particular photo and the format of each row is listed in ***RedditData-Cols.csv***:

```
<image id>, <unixtime>, <post title>, <subreddit>, <# upvotes>, <# downvotes>, <# comments>
```

Provided are three different sized Reddit data files:

1. **RedditData-Small.csv** - reposts of 10 unique photos (∼200 rows)(∼16KB in size)
2. **RedditData-Medium.csv** - reposts of 1,000 unique photos (∼12k rows)(∼1MB in size)
3. **RedditData-Large.csv** - reposts of 16,736 unique photos (∼130k rows)(∼9MB in size)

Note: The small and medium data sets are simply subsets of the large data set. As you start to implement your spark programs, you should start off by running your programs on the smaller data sets then move up to the larger ones.

1.1 Image Impact (20 points)

Understanding Popular Image Content:

A photo can be reposted in multiple subreddits and the level of interactions and impact the photo has is highly dependent on which subreddit it is posted to. We can safely assume that a picture will do better in a more relevant subreddit, but what can we say about the photo in relation to the overall Reddit audience? If we post a gif of a baby elephant in multiple subreddits, would it make a bigger impact than posting a picture of a kitten in the same subreddits?

In this part, we want to answer the question:

How impactful is a particular kind of photo?

In this part we will compare images that have been reposted based on the amount of impact they had on Reddit. We can do this by measuring the amount of engagement each photo induced. This can be done by through simply counting user interactions for each photo.

Calculating Image Impact:

As you've learned, users on Reddit can interact with posts by upvoting, downvoting, or commenting. We can simply define a post's impact to be the number of times it was upvoted, downvoted, or commented on:

$$Impact = upvotes + downvotes + comments$$

This would calculate the impact of a particular post, not a particular image. To calculate the impact of an image, we can simply extend our definition of the impact of a post to define the impact of a particular image to be the sum of all impact values for all the posts associated to a particular image:

$$Impact_{id} = \sum upvotes_p + downvotes_p + comments_p, \forall p \in P_{id}$$

In this equation, p is a post within P_{id}, the set all of posts associated with some image with an id id.

Output and Analysis:

The output of your computation should be formatted as rows where each row consists of:

`<image id>  <impact score>`

For Example:
As a basic example lets say we have three entries representing three posts:

```
image_id, unixtime, title, subreddit, #_upvotes, #_downvotes, #_comments
123, 1733286229, A, funny, 200, 50, 6
152, 1176793353, B, cute, 350, 100, 12
123, 1559764551, C, pics, 450, 5, 12
```

We can see the first and third entry are posts associated to an image with the id 123 and the second entry is a post associated with a image 152.

In the end, the result of our program should look like such:

```
123 723
152 462
```

From this we can see more people interacted with posts associated with the image of id 123 compared to the posts associated with image of id 152. We could naively say the image of id 123 had more impact than the image of id 152. If image of id 123 was perhaps an image of a cat and image of id 152 was an image of a shoe, we can predict that if we posted a similar image of a cat to that in image of id 123, it could potentially be more engaging and get more exposure than if we posted a picture of a shoe similar to the image of 152.

1.2 Repost Timing (25 points)

Understanding Diurnal Patterns:

Almost half of Reddit's traffic is from the United States. Given this and given the United States only spans 6 out of the ∼24 timezones in the world, perhaps there are some diurnal patterns that can be found with regards to post performance. A diurnal pattern is any pattern that occurs every 24 hours. One example of a potential diurnal pattern is that perhaps between 6PM and 8PM EST, Reddit posts see more interaction because a majority of users are off from work, school, etc. However our data is centered around reposts so we want to find if there are particular diurnal patterns with regards to repost performance.

In this part, we want to answer the question:

Is there an ideal time of the day to post a repost?

In this part we will compare the hours of the day based on the amount of impact reposts posted within a particular hour had on Reddit. We can do this by measuring the amount of engagement/impact reposts posted within a particular hour had.

Calculating the Most Impactful Repost Time:

Calculating the an hour's impact is similar to how you calculated a particular photo's impact in the previous section, except that the impact values are summed based on which hour of the day a repost was posted.

$$Impact_{hour} = \sum Impact_p, \forall p \in P_{hour}$$

In this equation, p is a post within P_{hour}, the set all of posts submitted within the same hour of the day.

Output and Analysis:

The output of your computation should be formatted as 24 rows where each row consists of:

`<hour offset>  <impact score>`

where the hour offset is based on EST (the timezone Rutgers is located in).

For Example:
As a basic example lets say we have three entries representing three posts:

```
image_id, unixtime, title, subreddit, #_upvotes, #_downvotes, #_comments
123, 1616703628, A, funny, 200, 50, 6
152, 1616779228, B, cute, 350, 100, 12
123, 1616780728, C, pics, 450, 5, 12
```

Here, when we convert the unix timestamps to EST, we see the first post was posted at 4:20PM EST on 3/25 and the second and third posts were posted at 1:20PM EST and 1:45PM EST respectively on 3/26.

In the end, the result of our program should look like such:

```
0 0
....
13 929
....
16 256
....
23 0
```

In this example output we can see that reposts submitted between 1PM-2PM EST (13th hour of the day) ended up having the most impact/reach. Perhaps we can deduce that people like to interact on Reddit when it's around lunch time, as people each typically lunch between 10AM-2PM within the United States (Remember 1-2PM EST is equivalent to 10-11AM PST).

2 Analyzing Netflix Data

Intro to Netflix

Netflix is a popular subscription-based streaming service where users can watch and rate movies/tv shows that are present in their library of tens of thousands of titles. Users can navigate the large library of titles in multiple ways; they can search titles manually, navigate through titles within a particular category, or sift through titles that Netflix recommends them. Overall, Netflix's recommendation system is the biggest feature that sets them apart from other streaming services. By using information about the user, their watch history, and title ratings, Netflix can recommend titles that the user may like with high probability, leading to a positive platform experience.

You can learn more about Netflix here: https://en.wikipedia.org/wiki/Netflix

Netflix as a Basis for Developing Prediction Algorithms

In order to maintain customers and ensure subscribers keep using their service, it is important that Netflix's recommendation and discovery mechanisms are highly effective in helping users find quality content. A large area that supports these mechanisms are predictive algorithms. Predictive algorithms use statistical analysis, machine learning, and perhaps even AI to make predictions about the future. In the context of Netflix, they want to predict what a user would rate a particular movie to help the recommendation system determine whether or not to recommend a particular title to a particular user.

Netflix's proprietary recommendation algorithm known as *CineMatch* had already been proven to be very good at predicting movies that subscribers would like. However at some point, Netflix realized that there could be better prediction algorithms out there. Because of this, they released a public data set of user reviews as part of a contest to see if people could come up with better algorithms. This contest, known as the "Netflix Prize", was held over multiple years from 2007 to 2009 and yielded algorithms and improvements that has helped Netflix's prediction and recommendation algorithms become one of the best among streaming services today.

You can learn more about the Netflix Prize here: https://en.wikipedia.org/wiki/Netflix_Prize

Goal:

The goal of this section is to create our own Spark programs to process the Netflix data set to generate some useful insights of our own.

Data Format

Every row within the Netflix data set represents a movie review and the format of each row is listed in ***NetflixData-Cols.csv***:

```
<movie_id>, <customer_id>, <rating>, <date>
```

Provided are three different sized Reddit data files:

- **NetflixData-Small.csv** - reviews of 10 movies (∼20k rows)(∼450KB in size)
- **NetflixData-Medium.csv** - reviews of 1,000 movies (∼5m rows)(∼120MB in size)
- **NetflixData-Large.csv** - reviews of 18k movies (∼100m rows)(∼2.5GB in size)

Note: The small and medium data sets are simply subsets of the large data set. As you start to implement your spark programs, you should start off by running your programs on the smaller data sets then move up to the larger ones.

2.1 Average Movie Rating (20 points)

Who cares about the Average Movie Rating?

Remember the goal of any subscription-based service is user retention. The longer a user is subscribed, the more money the company gets. In the context of Netflix, if a user is happy with how and what they watch, they will most likely keep using the service in the future and maintain their subscription.

So where does the average rating fit into this? So the average rating for a movie/show fits into helping users in "how" they consume media on Netflix. Since the average rating of a movie/show is a good indicator of the quality of that movie/show with respect to the general audience, users can use these ratings to guide the way they choose the next movie or show to watch and identify whether or not a movie or television show is worth watching.

Calculating the Average Rating for a Movie:

Although calculating average ratings for each movie is straight forward, things could be more scalable if we computed the average ratings for each movie in a distributed way. That way, if we were to get an even larger data set, the time it takes to compute the average rating for each movie wouldn't scale as linearly as if we used a single process/system.

Given this set of customers, the average rating of each movie of id id is the sum of all ratings for that movie divided by the number of ratings:

$$AvgRating_{id} = \frac{\sum Rating_i}{\#\ of\ ratings} \forall i \in R_{id}$$

The average rating for a movie with id id would be the summation of all ratings of each review i in the set R_{id} which is the set of all reviews for a movie with id id.

Output and Analysis:

In the end, the result of our program should be the list of all movies and their average rating in the format:

```
<movie id> <average rating>
```

where the average rating is truncated to 2 decimal places.

For example:
As a basic example lets say we have 5 reviews spanning two different movies:

```
movie_id, customer_id, rating, date
201, 1062, 2, 3-13-20
973, 7028, 5, 4-23-20
201, 1637, 5, 3-24-20
973, 1644, 4, 5-1-20
201, 2023, 3, 5-19-20
```

We can see that three of the reviews are for movie with id X and two of the the reviews are for the movie with id Y.

In the end, the result of our program should look like such:

```
201 3.33
973 4.5
```

Based on the results we can see that, according to our general public sample of 5 reviews on 2 movies example, a user would probably like movie 973 compared to 201.

2.2 Recommendation Graph (35 points)

Understanding Customer Similarities:

In the last exercise you calculated the average rating for each movie, which is a good metric to help guide Netflix users to pick content that will be of high quality to them. However, another mechanism that is typically used to guide users to quality content is a recommendation system. Recommendation systems typically take into account a user's interests, watch history, as well other user's reviews to recommend content that the user may like.

Netflix currently has their own recommendation system that millions of users are exposed to everyday. These algorithms are proprietary, but perhaps we can build towards our own recommendation system and algorithms by generating our own naive recommendation graph.

The Recommendation Graph:

The idea of a recommendation graph is simple. We want a graph that will encapsulate similarities between any two customers so that we can recommend content to similar customers.

The structure of the graph is as follows:

1. A node within the graph represents a customer.
2. Each edge between nodes is undirected and represents similar taste between two customers of some weight.

To makes things simple, let us assume that two customers A and B have similar taste if they gave the same rating r to the same movie m:

$$Similar(A, B, m) = \begin{cases} 1 & \text{if } r_{m,A} = r_{m,B} \\ 0 & \text{otherwise} \end{cases}$$

Assuming the weight of the edge between A and B would be the number of times A and B had similar tastes, then the weight of the edge between A and B would simply be the total number of all similar ratings:

$$EdgeWeight(A, B) = \sum Similar(A, B, m), \forall m \in M$$

where M is the set of all movies.

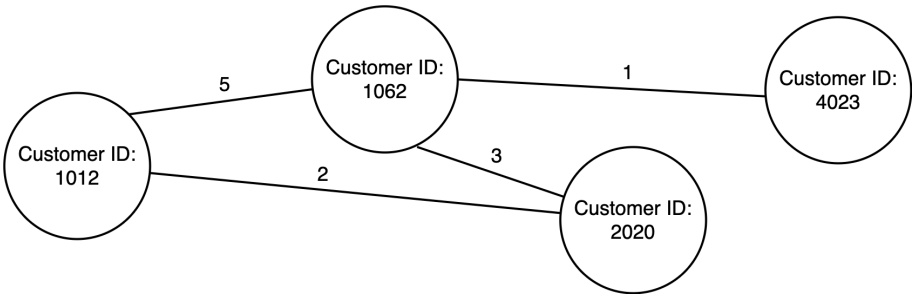

Figure 1: Example Recommendation Graph

In the end, with this graph, if we find that a customer highly rated a particular movie/show, we can recommend it to their neighbors in the graph that have a high edge weight since that means they have similar tastes.

Generating a Graph:

Generating a graph from our data set will be a little bit more involved than our other analysis. However the main idea is that for each movie we will:

1. For each (movie, rating) pair, identify the set of customers who rated the movie with that particular rating value
2. For each (movie, rating) pair, generate an edge between each pair of customers within the set of customers that gave the movie the same rating
3. Aggregate the edges for each customer pair to find the edge weights.

Hint: Uniform Edges. Since edges are undirected, edges between customer A and B can take the form (A, B) or (B, A). However this could lead to some issues when aggregating edges since there are two possible forms to consider. To mitigate this issue and ensure edges maintain a single form between any two customers, format the edge in such a way where the first customer id of is always less than the second customer id, as shown in the example output in the next section.

Output and Analysis:

In the end, the result of our program should be a list of all the weighted edges in our graph in the format:

`(<customer_id_1>, <customer_id_2>) <weight>`

where *customer_id_1* is less than *customer_id_2*.

For example:
The output for the graph in Figure 1 would look something like:

```
(1012,1062) 5
(1012,2020) 2
(1062,2020) 3
(1062,4023) 1
```

Here we can see that customer 1012 has 5 similar ratings to 1062. So, if customer 1012 highly rates a movie/show, we can recommend that movie/show with relatively high confidence to customer 1062 and vice versa.

3 Getting Started

3.1 Project Template Overview

The following is the bare structure of the project template:

```
project-template
├── src
│   └── main
│       └── java
│           └── com
│               └── RUSpark
│                   ├── WordCount.java
│                   ├── RedditPhotoImpact.java
│                   ├── RedditHourImpact.java
│                   ├── NetflixMovieAverage.java
│                   └── NetflixGraphGenerate.java
└── pom.xml
```

Notice: If you imported the project into Eclipse, additional folders may be present. This is due to Eclipse realizing that it is a Maven project, auto generating extra folders that it may use.

Main Implementation Files

Within the /src/main/java/com/RUSpark directory the five main Java files:

- ***WordCount.java*** - This is an example Spark program that counts the numbers of occurrences of each word in a text file.
- ***RedditPhotoImpact.java*** - This is where you will implement the Spark program that calculates the impact of every photo within the Reddit data set
- ***RedditHourImpact.java*** - This is where you will implement the Spark program that calculates the impact of all posts submitted within the same hour of the day
- ***NetflixMovieAverage.java*** - This is where you will implement the Spark program that calculates the average rating of all movies within the Netflix data set
- ***NetflixGraphGenerate.java*** - This is where you will implement the Spark program that generates a recommendation graph as a list of weighted edges between customers

Pom Configuration File

Lastly, you will notice at the root of the project there is a file called ***pom.xml***. This file is for Maven. Like mentioned in the previous section notice, Maven is a build tool used for Java projects and uses the pom.xml file to know how to build the Java project as well as know what dependencies need to downloaded before hand. **DO NOT EDIT THIS FILE**. The pom.xml file has been preconfigured to generate Runnable JARs every time the project is compiled *for your convenience*.

3.2 Reddit Dataset

Contents and Description:

The Reddit Dataset (***RedditData.zip***) includes the following files:

- ***RedditData-Cols.csv*** - file of column names (1 row) (~100B in size)
- ***RedditData-Small.csv*** - reposts of 10 unique photos (~200 rows)(~16KB in size)
- ***RedditData-Medium.csv*** - reposts of 1,000 unique photos (~12k rows)(~1MB in size)
- ***RedditData-Large.csv*** - reposts of 16,736 unique photos (~130k rows)(~9MB in size)

Data Format:

Every row within the Reddit Data represents a repost and the format of each row is listed in ***RedditData-Cols.csv***:

```
<image id>, <unixtime>, <post title>, <subreddit>, <# upvotes>, <# downvotes>, <# comments>
```

Space Requirements:

The compressed dataset (***RedditData.zip***) is appoximately ~4MB in size and uncompressed is around ~10MB, so make sure you have about ~15MB of free space when downloading.

Downloading the Dataset:

- **Via Web Browser:**
  To download the data set from a browser, use the following link:

  https://drive.google.com/uc?export=download&id=1zhtVn8-64QKpjl_y0uHWImooeSJkv7bc

- **Via Command Line (WGET):**
  To download the dataset using *wget*, copy and paste the following command and run it:

```
Command Line
  $ wget -O RedditData.zip "https://docs.google.com/uc?export=download&id=
  "1zhtVn8-64QKpjl_y0uHWImooeSJkv7bc
```

- **Via Command Line (CURL):**
  To download the dataset using *curl*, copy and paste the following command and run it:

```
Command Line
  $ curl -L -o RedditData.zip "https://drive.google.com/uc?export=download
  &id="1zhtVn8-64QKpjl_y0uHWImooeSJkv7bc
```

Extracting the Dataset:

To extract the dataset, simply use the *unzip* command:

```
Command Line
  $ unzip RedditData.zip
```

3.3 Netflix Dataset

Contents and Description:

The Netflix Dataset (***NetflixData.zip***) includes the following files:

- ***NetflixData-Cols.csv*** - file of column names (1 row) (∼30B in size)
- ***NetflixData-Small.csv*** - reviews of 10 movies (∼20k rows)(∼450KB in size)
- ***NetflixData-Medium.csv*** - reviews of 1,000 movies (∼5m rows)(∼120MB in size)
- ***NetflixData-Large.csv*** - reviews of 18k movies (∼100m rows)(∼2.5GB in size)
- ***NetflixData-Titles.csv*** - list of movie indexes and titles (∼18k rows)(∼600KB in size)

Data Format:

Every row within the Netflix Data represents a movie review and the format of each row is listed in ***NetflixData-Cols.csv***:

```
<movie id>, <customer id>, <rating>, <date>
```

Space Requirements:

The compressed dataset (***NetflixData.zip***) is appoximately ∼700MB in size and uncompressed is around ∼2.6GB, so make sure you have about ∼3.5GB of free space when downloading.

Downloading the Dataset:

- **Download via Browser:**
  To download the data set from a browser use the following link:

  https://drive.google.com/uc?export=download&id=1RcLSh1EuY5iL48ldqiTYfZaM7zw7KASi

- **Via Command Line (WGET):**
  To download the dataset using *wget*, copy and paste the following command and run it:

```
Command Line
$ wget --save-cookies cookies.txt "https://docs.google.com/uc?export=dow
nload&id="1RcLSh1EuY5iL48ldqiTYfZaM7zw7KASi -O- | sed -rn "s/.*confirm=(
[0-9A-Za-z_]+).*/\1/p" > confirm.txt; wget --load-cookies cookies.txt -
O NetflixData.zip "https://docs.google.com/uc?export=download&id="1RcLSh
1EuY5iL48ldqiTYfZaM7zw7KASi"&confirm="$(<confirm.txt); rm -f confirm.tx
t cookies.txt
```

- **Via Command Line (CURL):**
  To download the dataset using *curl*, copy and paste the following command and run it:

```
Command Line
$ curl -L -c cookies.txt "https://docs.google.com/uc?export=download&id=
"1RcLSh1EuY5iL48ldqiTYfZaM7zw7KASi | sed -rn "s/.*confirm=([0-9A-Za-z_]+
).*/\1/p" > confirm.txt; curl -L -b cookies.txt -o NetflixData.zip "http
s://docs.google.com/uc?export=download&id="1RcLSh1EuY5iL48ldqiTYfZaM7zw7
KASi"&confirm="$(<confirm.txt); rm -f confirm.txt cookies.txt
```

Extracting the Dataset:

To extract the dataset, simply use the *unzip* command:

```
Command Line
$ unzip NetflixData.zip
```

3.4 Building the Project

To build your project, you will need to use Maven and run "goals". Goals are procedures that carry out actions related to the project lifecycle. There are three Maven goals you will need will to familiarize yourself with in order to build and compile your project:

- install - this goal downloads any dependencies for your project
- package - this goal compiles the project and packages artifacts
- clean - this goal cleans the project of any artifacts and code generated by the project

Install

The install goal downloads any dependencies and plugins you will need for your project. You will need to run this goal when you first import your project so the necessary Apache Spark libraries can be downloaded. If you don't do this, your IDE may give you errors or you may get errors when you try and build the project.

- To do this in Eclipse:
    1. In Eclipse, select the project folder within the Package Explorer window.
    2. Go to "Run" > "Run As" > "Maven Install"
- To do this in Command Line:
    1. Navigate to the root of the project directory.
    2. Run the following: "mvn install"

Package

The package goal will carry out the Maven build process: compiling project code, packaging artifacts, as well as carry out any additional tasks defined within the pom.xml file. You will need to run this goal for two reasons. The first is to generate gRPC Java code. Within the pom.xml there is a task that invokes the Google Protocol Buffer compiler to take the proto file you defined and generate Java classes. These classes are what you will use to implement your service. The second reason to run the package goal is to compile your server and client code so that you can run them.

- To do this in Eclipse:
    1. In Eclipse, select the project folder within the Package Explorer window.
    2. Go to "Run" > "Run As" > "Maven build..."
    3. Within the Goals text box, type in "package"
    4. Click "Run"
- To do this in Command Line:
    1. Navigate to the root of the project directory.
    2. Run the following: "mvn package"

Clean

The clean goal will clean the project directory of any generated project artifacts and source code. Project artifacts and generated code are usually placed within a folder named "target". You will need to run the clean goal before you build your project with the package goal to ensure any files from your old code is gone. You can run the package goal without running the clean goal as it will just overwrite the existing files, but there may be some files that were not overwritten still lying around.

- To do this in Eclipse:
    1. In Eclipse, select the project folder within the Package Explorer window.
    2. Go to "Run" > "Run As" > "Maven clean"
- To do this in Command Line:
    1. Navigate to the root of the project directory.
    2. Run the following: "mvn clean"

Note: Sometimes when running the previous goals in Eclipse, it may seem like nothing changed within the project structure. An example would be the target folder not showing up after you built your project. This is because Eclipse may have failed to update the window to reflect the new files. If this happens, try refreshing by right-clicking the project folder and selecting refresh.

3.5 Running the Project

After compiling your project, there will be generated jars within the */target* folder within the project directory. These *.jar files are the compiled Spark programs that you will submit to the Spark cluster to run.

Program Usage

To run a Spark program on a Spark cluster, you will have to use the *spark-submit* command to submit a *.jar Spark program to run. Each program in this project is structured to take in a single argument which specifies the data set to process. Thus the usage is as follows:

```
spark-submit  <path to Spark program>  <path to data set>
```

For example:

Command Line

```
$ spark-submit RedditPhotoImpact.jar /common/users/djd240/RedditData-Large.csv
```

Available Spark Clusters:

You can run your project in two different places:

1. **The iLab Machines**
   The iLab machines at Rutgers are equipped with Spark running in standalone mode. To run your program: (1) Ensure your compiled jar and data set are uploaded and present on the iLab machine then (2) invoke the spark-submit command like shown above.

2. **Amazon Web Services**
   If you haven't already used AWS, this could be a great opportunity to understand how AWS works. To learn how to run set up a Spark cluster on AWS and run your project take a look at the following guide here: https://github.com/DaveedDomingo/Spark-Reddit-Netflix-Project/blob/master/SparkOnAWS.pdf

Note on large data sets on iLab machines: The data sets within this project are somewhat large. By default on the iLab machines, you have approximately ~6GB of storage in your home directory (`/ilab/users/<your netid>`). However this may not be enough space. There is another filesystem where you have more space located at (`/common/users/<your netid>`). In this file system you have approximately ~100GB of storage space available to you. This should be enough to store anything that won't fit in your home directory. **If using the iLab machines for this project, I recommend you download the data sets directly to your** `/common/users/<your netid>` **directory to ensure you don't take up space in your main home directory**. For more details on all the iLab storage options, see here: https://resources.cs.rutgers.edu/docs/file-storage/storage-technology-options/

Saving Output to File: So the results of your program should print out in STDOUT. However you'll notice when running your Spark program, you'll also see output given from spark. You can save and isolate your output by redirecting it to a file like:

```
$ spark-submit myjar.jar inputdata.csv > out.txt
```

4 Recommended Steps

The hardest part would be to learn how Apache Spark works. I recommend spending a good amount of time in the beginning to fully understand Apache Spark and try and get an test Spark program working before working on any of the parts of this project:

1. Read and learn about Apache Spark
2. Take a look a word count example and get it running on the iLab Machines and AWS
3. Implement Reddit Data programs
4. Implement Netflix Data programs
5. Write Writeup
6. Submit on time

5 Submission

To submit your project, submit the following items:

1. **The following raw files:**
   - RedditPhotoImpact.java
   - RedditHourImpact.java
   - NetflixMovieAverage.java
   - NetflixGraphGenerate.java
2. **A pdf document consisting of the following things:**
   - A few pages describing your implementation and analysis:
     - For each program, describe what transformations you did to get to the result.
     - Based on the result from RedditPhotoImpact, what was the most impactful photo for the whole data set?
     - Based on the result from RedditHourImpact, what hours of the day (in EST) did submitted posts have the most impact/reach?
     - Based on the result from NetflixMovieAverage, name 2-3 movies that had the highest average rating.
   - Finally, a couple of paragraphs of reflection describing: What was the hardest parts about implementing your project and how did you overcome them? If you had difficulties and couldn't finish the project, describe what you think you could've done better or what you think would've helped.

Please follow all submission guidelines. Failure to do so will result in points being taken away.

Don't slack on the doc. The write up is very important when it comes to evaluating your project. Not only does it help us understand your approach and how your implementation works, it also shows us the amount of effort you put into the project. The write up also helps us survey the amount of effort needed to complete the project as well as gauge the project's overall difficulty. Because of these things, the write up in general allows for as much partial credit as possible as well as reasonably scaling grades across all submissions.

6 Additional Tips and Tricks

When in doubt, Google (*responsibly*): If you have an issue or your program is throwing an error that you don't know how to fix, Google It. Someone, somewhere, probably faced the same issue at some point. However do so responsibly and don't cheat. If you are caught cheating, your case will be handled according to Rutgers' academic integrity policy (http://academicintegrity.rutgers.edu/).

Combining Maven Goals: You may find yourself running Maven goals frequently to build the project. You might find yourself running them so frequently it may just interfere with your precious development time. Luckily you can group goals together. For example, since it is good practice run the clean goal before the package goal, we should group them. In Eclipse this is done by running a Maven build like we are running the package goal, but instead of writing "package" in the Goals text box, we write "clean package". In Command Line, this is done by navigating the root of the project directory and running "mvn clean package". These will both run the clean goal first then the package goal.

Frequently Asked Questions

- **Can I use external libraries other than the built in java libraries in my implementations?** No. You may only use the standard Java Libraries and anything else within the Apache Spark libraries.
- **Can I use helper methods in my implementations?** Yes, as long as the program works.
- **Can I make other java files to support my implementation?** No, please only write your implementation within the supplied template java files.
- **Can we implement the programs on my own computer?** Yes, as long as it can compile and run on the iLab machines as that is where they will be compiled, ran, and graded.

Additional Questions

If you have any questions about the project or are having any issues, email me at David.Domingo@rutgers.edu