Assignment 2
SENG5709
Daniel Koniar
5/9/2020

Steam Dataset Analysis

Introduction

For Assignment 2, I analyzed a steam dataset collected in 2016 by BYU University [6]. Steam is one of the largest gaming marketplaces, which offers a platform for organizing, playing, buying, and selling video games [8, 9]. With how popular Steam is across the world and my personal interest around gaming, I found it interesting to evaluate datasets around the popularity of certain games. It is also worth noting that I did find my Steam account in the dataset downloaded.

Setup

Preparing the Data

The dataset was hosted on a college domain with a size of 17GB zipped [6]. I could not actually get this data to download without failure. I ended up finding a website called *Academic Torrents*, which contained torrents for large datasets [7, 13]. The Steam dataset that I was going to analyze was included in these torrents and the dataset was seeded. Thanks to this website, I could download the dataset at a fast rate (and yes, I seeded the dataset to help lost souls like me).

After the extracting the compressed dataset, the format was a 160GB SQL file. From here, I had to investigate the SQL inside the file, which required me to download a large file editor. The large file editor that I decided on was Liquid Studio since I could use it for free (Community Edition) [1]. Upon further investigation, it appeared that the dataset was designed for MariaDB. Through some research, I found that MariaDB and MySQL shared syntax, so I installed MySQL and ran the script [3, 4].

After about twelve hours, the script had completed (and successfully, to my relief) and the data was within the MySQL database. To my demise, MySQL struggled to handle the large amount of data inserted. After some research online (again), I discovered that I could migrate the data to SQL Server with a tool called Sqlines [5]. The reason I migrated to SQL Server is because that is the main database I use at work and have the most knowledge around. Since the project was oriented towards using Spark, I decided this was the best route to save myself from having to learn a different flavor of SQL.

From SQL Server, I learned that the dataset needed to be contained within GitHub per this assignment's requirements. I knew that all the data I was using was far too big to include in the whole project, so I create a random subset of the original relational table and pulled data around it. In the code, it will be noticed that tables end in *r3pg*. *r3pg* stands for "Random 3 Percent Games", which means that I filtered the games dataset for *games_2* type only and extracted a random 3% of the rows into its own table. This provided me with around ~19 million rows of data for the *games_2* table alone.

Using the *r3pg* tables, I generated CSV files (which SQL Server doesn't like to properly escape) after trial and error. From there, I used a program called *CSV Splitter* to split the CSV files into multiple CSV files so that I could commit them to the GitHub repo [10]. Spark was very capable at combining multiple CSV files together and required minimal changes to the loading code.

The final transformation that the dataset went through was into Parquet files, which were generated through Spark API and stored to the repo for usage [11]. Parquet files stored the data in a columnar format, which generally performs better with analytics (at least in my case). The CSV file format generally completed tests within a few minutes. The Parquet file format (with the same size datasets and tests) completed tests within a few seconds. I was originally planning on having a caching system used for the repeat datasets. However, with the speed increase from Parquet files, it was deemed unnecessary.

Spark API

The Scala Spark API was used to perform analytics in code. The main Spark API that was used to perform analytics was DataFrames. I leaned away from SQL since I have a decent background in writing SQL and had SQL queries (in SQL Server) that generated answers for the analytics I was looking at to verify correctness. I originally attempted to use RDDs, but, with large amounts of data, the runtime for a test was not feasible. This led me to the usage of DataFrames. DataFrames seemed to be more performant and use less space than RDDs (while SQL Server was eating 40GB of RAM on my system) [12].

Analysis

Questions

The following subsections contain the answers to the questions present in the proposal.

**Note: Questions marked with an asterisk were slightly adjusted to align with the data in the dataset.*

Question #1: What game has the highest average play time?

The game that had the highest average playtime was **Dota 2**. The average playtime (forever) between all users in the dataset was about **17,680 minutes**. This is about 294.67 hours.

Question #2: What game has the highest amount of users in the dataset?

The game that had the highest amount of users in the dataset was **Counter-Strike**. There was **655,297 users** that had this game in the evaluated dataset.

Question #3: What user has the highest play time for a single game?

The user that had the highest play time for a single game was **[U][A]-VampeD**. The game that this user was playing was **Counter-Strike: Source**. The total playtime they had for this game was **2,434,347 minutes**, which is about 4.63 years. It is worth noting that Steam tracks playtime when the game is running; the user does not actually have to be playing the game.

Question #4: What user has the highest total play time?

The user that had the highest play time total among all their games was also **[U][A]-VampeD**. The total playtime for this user was **2,461,190 minutes**, which is about 4.68 years.

*Question #5: What game has the highest ratio of [number of minutes played] to [average minutes per user]?**

The game with the highest ratio of [total hours played] to [average hour per user] was **Counter-Strike**. The ratio came out to be **655,297**.

If you are getting déjà vu, it is because this question has the same exact answer as Question 2 above. To show you how this happened, lets set up a formula quick. If we can agree that the average playtime is computed from the total playtime divided by the total users with the game:

$$Playtime_{avg} = \frac{Playtime_{total}}{Users_{game}}$$

This implies the following:

$$\frac{Playtime_{total}}{Playtime_{avg}} \equiv \frac{Playtime_{total}}{\left(\frac{Playtime_{total}}{Users_{game}}\right)} \equiv \frac{Playtime_{total} * Users_{game}}{Playtime_{total}} \equiv Users_{game}$$

So, in a very roundabout fashion, I recomputed the answer to Question #2 above. Only after performing this analysis had I realized what I had done and the issue with this question.

*Question #6: What is the average amount of users per required age?**

The following are the required ages and the amount of users with the game for that required age:

Required Age	Total Users
0	15,799,912
6	289
10	2423
12	2347
13	70,745
15	10,680
16	125,695
17	3,172,141
18	151,991

The original question in the proposal used ESRB rating. However, in the dataset, I only find *required age*. From this data above, we can deduce what ESRB rating the game may fall within. It appears that ***Everyone* is the most popular** (granted that it is equivalent with the required age of 0). The **second most popular game rating is *Mature*** (granted that it is equivalent with the required ages of 17 and 18). The third most popular ESRB rating would have to be *Teen* (granted that it is equivalent with the required ages of 13 and 15).

Question #7: What game had the biggest increase in average playtime from the earliest to latest retrieval date?

The game that had the biggest increase in average playtime between *games_1* and *games_2* capture was **Darkfall Unholy Wars**. This game had an average playtime increase of **13,541 minutes**, which is about 225.68 hours. This makes sense since the game is an MMO that was released just before the capture date of *games_1*. Typically, MMOs will have a high amount of hours per user.

Question #8: Did games released during the data retrieval have a higher increase in average play time? Or did pre-existing games have a higher increase in average play time?

I found in my analytics that the games that were **released before *games_1* was captured had about a 254.32 minute increase** in total playtime on average. I found that the games that were **released after *games_1* was captured had about a 258.32 minute increase** in total playtime on average. **This means that games released between *games_1* and *games_2* had the greatest increase in playtime**. This makes sense since many games die off after a year or so for most players. New game releases tend to see a rush of players to begin with.

Spark Analysis

Most of the questions that I posed involved averages or sums, which means a lot of group-by are being performed. After performing these group-bys, aggregation is called on each group. One issue with group-bys with Spark is that it causes a shuffle. It causes extra data to keep track of these groupings. To reduce this overhead, I should have reduced the size of my data selecting only the necessary data.

Another element that I consistently used was transformation operations. I performed multiple filters and joins and stored those DataFrames. These operations can create new DataFrames, which can lead to more storage usage (which is bad for large datasets).

Lastly, another thing that I could improve on is using less variables if possible. Spark seems to optimize DataFrame query plans, regardless of order, by Spark's catalyst optimizer. One way that I disrupted this plan was by storing DataFrames that were only partially evaluated. It does make me question if readability should outweigh performance in the case of Big Data analytics. If I would have leaned away from readability, I could've increase performance by chaining all the queries together. This would reduce the necessity of storing more DataFrames and allow the catalyst optimizer to optimize the full query (instead of fragments).

Conclusion

I learned that Spark can handle large amounts of data well. I also learned that I should switch to the Parquet format ASAP when using Spark. The columnar format of these files (along with the other data they store and the native format) perform much better than the CSVs and the connection to SQL Server. Also, the size of the Parquet files vs the CSV is much smaller, which would have allowed me to use a larger amount of data than 3%. On top of this knowledge, I got more experience using DataFrames and their syntax. I found that DataFrames were the most performant when dealing with very large datasets from SQL Server (e.g. millions of rows). This usage translated through the transformation of my data to the committed format it is in now.

Also, I included the whole section about *Preparing the Data* because it was what I spent over half of my time doing with this assignment. I felt that it is important to emphasize the time spent transforming the data into consumable formats since it is also a key factor in Big Data analytics. I learned that data can be malleable given the right toolsets and that have basic tests (e.g. row and column counts) can help determine if the integrity of the data has been maintained.

References

[1] Liquid Technologies. (n.d.). Liquid Studio. Retrieved May 9, 2020, from https://www.liquid-technologies.com/xml-studio

[2] MariaDB Foundation. (n.d.). MariaDB Foundation. Retrieved May 9, 2020, from https://mariadb.org/

[3] yannis. (2011, November 17). What's the difference between MariaDB and MySQL? Retrieved May 9, 2020, from https://softwareengineering.stackexchange.com/a/120224

[4] Oracle. (n.d.). MySQL. Retrieved May 9, 2020, from https://www.mysql.com/

[5] Sqlines. (n.d.). MySQL to Microsoft SQL Server Migration. Retrieved May 9, 2020, from http://www.sqlines.com/mysql-to-sql-server

[6] O'Neill, M., Wu, J., Vaziripour, E., & Zappala, D. (n.d.). CONDENSING STEAM: DISTILLING THE DIVERSITY OF GAMER BEHAVIOR. Retrieved April 19, 2020, from https://steam.internet.byu.edu/

[7] Henry Z. Lo. and Cohen, Joseph Paul "Academic Torrents: Scalable Data Distribution." Neural Information Processing Systems Challenges in Machine Learning (CiML) Workshop, 2016, http://arxiv.org/abs/1603.04395

[8] Prescott, S. (2019, July 5). The most popular desktop gaming clients, ranked. Retrieved from https://www.pcgamer.com/the-most-popular-desktop-gaming-clients-ranked/

[9] Steam. (2020). Steam, The Ultimate Online Game Platform. Retrieved April 19, 2020, from https://store.steampowered.com/about/

[10] ICT, P., & ERD Concepts. (n.d.). DB Toolbox. Retrieved May 9, 2020, from https://www.erdconcepts.com/dbtoolbox.html

[11] Apache Software Foundation. (n.d.). Parquet. Retrieved May 9, 2020, from https://parquet.apache.org/documentation/latest/

[12] Damji, J. (2016, July 14). A Tale of Three Apache Spark APIs: RDDs vs DataFrames and Datasets. Retrieved May 9, 2020, from https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

[13] Cohen, Joseph Paul, and Henry Z. Lo. "Academic Torrents: A Community-Maintained Distributed Repository." Annual Conference of the Extreme Science and Engineering Discovery Environment, 2014, http://doi.org/10.1145/2616498.2616528.

[14] Bhatia, S. (2018, October 6). Catalyst Optimizer : The Power of Spark SQL. Retrieved May 9, 2020, from https://medium.com/@Shkha_24/catalyst-optimizer-the-power-of-spark-sql-cad8af46097f