# PROJECT 2

"Top 20 Hashtag Counter using Amazon EMR"

Group 28

Spring 2024 TS: Cloud Computing: Fundamentals & Applications, SPRING 2024


**TEAM MEMBERS:**

- LAXMAN MUTHE
- PRERNA SEHRAWAT

# Table of Contents

# Data Processing Steps

For our project, we utilized Apache Spark with Python and the PySpark library, a potent tool for efficiently processing vast amounts of data. Spark's capability to distribute tasks across multiple machines greatly enhances speed, ideal for our task of extracting the top 20 frequent hashtags from Twitter data stored in a file named "smallTwitter.json."

Our initial steps involved setting up Spark locally and loading the JSON file for parsing. To streamline analysis, we consolidated all tweet information into single lines, treating each column as a string. This preprocessing step facilitated the identification of hashtags. Leveraging Spark SQL functions like explode, split, and lower, we segmented the text into individual words and standardized their casing to lowercase for consistency.

Subsequently, we sifted through the data, isolating hashtags by employing regular expressions to identify words beginning with '#' and counting their occurrences. To ensure accuracy, we used the '\w+' pattern, capturing hashtags with one or more letters. However, we avoided adding '\b' at the end, which might inadvertently include non-word characters following hashtags.

Upon completing the analysis, we identified the top 20 most frequently used hashtags. We generated an output folder containing a CSV file with this information and also displayed the results on the console for easy reference. This comprehensive approach enabled us to efficiently extract and analyze hashtag usage patterns from the Twitter dataset.

# Tool used

For our project, we opted for Apache Spark coupled with Python and the PySpark library. This combination proved to be a formidable tool for swiftly and effectively processing large volumes of data. With its adeptness at distributing tasks across multiple machines, Spark facilitated rapid execution, making it ideal for our endeavor. Leveraging this framework, we extracted the top 20 frequent hashtags from Twitter data stored in the file "smallTwitter.json".

# Why PySpark

PySpark stands out as the preferred choice for handling big data processing tasks on an Amazon EMR instance, thanks to its unique blend of scalability, user-friendliness, and performance advantages over alternative tools like Hadoop, Hive, and Presto. Unlike traditional Hadoop MapReduce, PySpark utilizes in-memory computing, which effectively minimizes latency and boosts processing speed by reducing disk I/O operations. Its Python API streamlines development and prototyping, empowering data engineers and scientists to craft concise and expressive code for intricate data processing tasks.

Furthermore, PySpark seamlessly integrates with the Hadoop ecosystem, ensuring compatibility with existing data infrastructure and tools. Its distributed computing model enables scalable processing of vast datasets across clusters of machines, making it adaptable to diverse workloads. With comprehensive documentation, active community support, and cost-effective deployment options on Amazon EMR, PySpark emerges as the top choice for efficiently managing large-scale data processing operations in a cloud environment.

# Data Processing Final Code

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, concat_ws, explode, lower, regexp_extract, split, expr

spark = SparkSession.builder.appName("hashtagApp").getOrCreate()

df = spark.read.json("smallTwitter.json")

all_text_df = df.select(concat_ws(" ", *[expr(f"CAST({c} AS STRING)") for c in df.columns]).alias("all_text"))

words_df = all_text_df.select(explode(split(lower(col("all_text")), "\\s+")).alias("word"))
hashtags_df = words_df.filter(col("word").rlike("^#\\w+")) \
    .select(regexp_extract('word', r'#(\w+)', 1).alias('hashtag'))

hashtag_counts = hashtags_df.groupBy("hashtag").count()

sorted_hashtags = hashtag_counts.orderBy(col("count").desc()).limit(20)

sorted_hashtags.coalesce(1).write.option("header", "true").csv("hashtags_output")
print(sorted_hashtags.show())

spark.stop()
```