# Comprehensive Platform for Curating Stock Investment Strategy

Ajinkeya Chitrey (ac5166), Yatharth Bansal (yb2540), Aishwarya Sen (as6718), Manasi Khandekar (mk4679), Aishwarya Patange (aap2239)

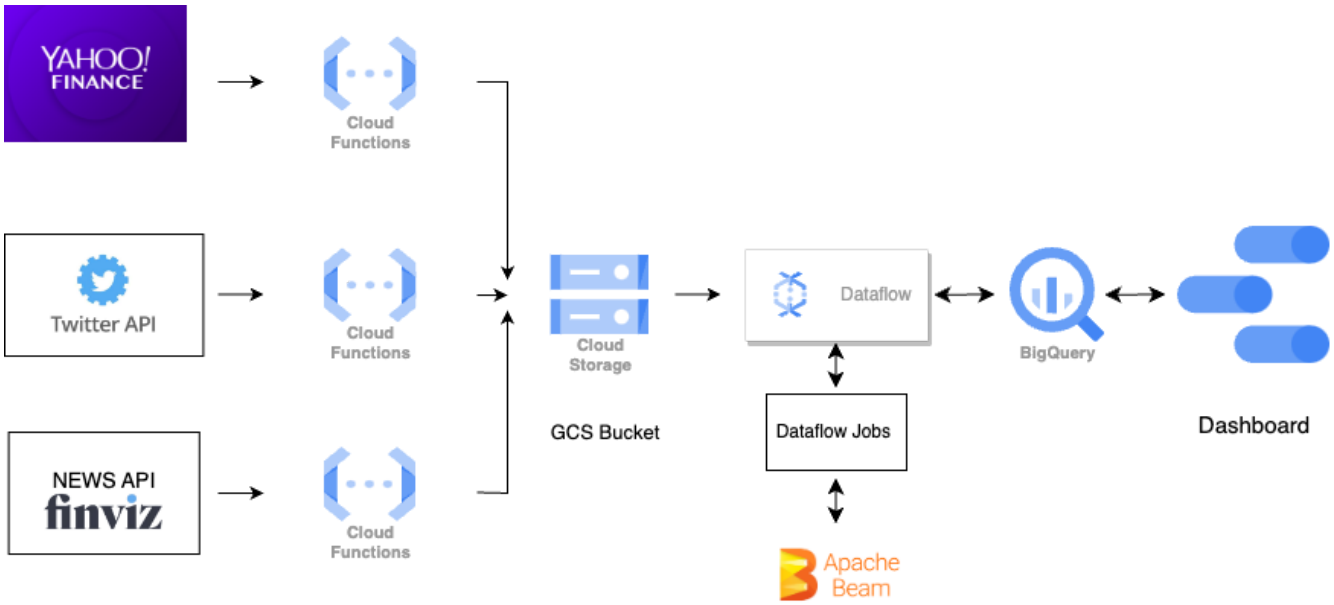
## Introduction:

In today's fast-paced financial world, keeping up with the latest market trends and information is crucial for successful trading. To address this need, SocialInvest - a platform for stock investment strategy, has been developed to display current and historical trends of stocks for 8 popular companies, along with the associated current market sentiment from conventional and social media.

The platform leverages advanced cloud computing technologies to process vast amounts of unstructured data in real-time, and extract valuable insights using machine learning algorithms. The platform uses Apache Beam in Cloud Dataflow to create a pipeline that encapsulates the entire data transformation workflow. Various PTransforms are performed on bounded PCollections of incremental data from Yahoo Finance along with the creation of sliding windows to compute 15-day moving averages over the stock information, in order to obtain a more consolidated view of the stock. The data is preprocessed and sentiment analysis scores are obtained using the SentimentAnalysisFunction in Python. Further optimizations such as load shedding and operator reordering will be incorporated as a part of future scope in order to improve the computational efficiency of the pipeline.

Overall, the platform provides a powerful tool for investors to make informed decisions based on accurate and timely information. The dynamic platform offers a user-friendly interface that consolidates various data sources and provides tools to analyze and interpret data, reducing the risk of financial losses and improving chances of achieving investment goals.

## System Architecture




Our system architecture consists of three phases: data pulling/fetching phase, data transformation and storage phase, and the front-end phase for displaying analysis and visualizations based on the transformed data.

### *Data Pulling/Fetching Phase:*

To collect data from multiple sources, we use three separate Cloud Functions, each responsible for pulling data from Yahoo Finance, FinViz, and Twitter. Cloud Functions provide a convenient and efficient way to execute small functions in response to events, making them ideal for building event-driven applications and microservices in the cloud. The pulled data is then stored in Google Cloud Storage (GCS) buckets to ensure control and reliability over the data.

```python
def event_handler(request):
    tickers = ['AAPL', 'GOOGL', 'AMZN', 'MSFT', 'TSLA', 'META', 'UBER', 'NFLX' ]
    df = pd.DataFrame(columns = ['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume',
      'company'])
    for ticker in tickers:
        data = yf.download(ticker, start="2012-01-01", end="2022-12-31")
        data = data.reset_index()
        data['Date'] = data['Date'].astype(str)
        data['company'] = [ticker]*len(data)
        df = pd.concat([df, data])
    file_name = "yfinance_data/" + "historical.txt"
    bucket_name = 'lssp_bucket'
    upload_blob(bucket_name, df, file_name)
```

```python
def upload_blob(bucket_name, source_data, destination_blob_name):


    """Uploads a file to the bucket. https://cloud.google.com/storage/docs/ """
    print('function upload_blob called')


    bucket = storage_client.get_bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_string(source_data.to_csv(index=False, header=False), 'text/csv')
    print('File {} uploaded to {}.'.format(
    destination_blob_name, bucket_name))
```

```python
def event_handler(request):
    queries = {
        "google stock": "GOOGL",
        "apple stock": "AAPL",
        "meta stock": "META",
        "tesla stock": "TSLA",
        "amazon stock": "AMZN",
        "microsoft stock": "MSFT",
        "uber stock": "UBER",
        "netflix stock": "NFLX"
    }

    bearer_token = "AAAAAAAAAAAAAAAAAA3yaIAEAAAAAaNqO8zh6dxIRAvAGj75TlmmBcrQ%3DwqDbQQqFzwFgonTEFnzblOl4R6PP1t5n1xqmy3yLdudpVA9MpU"

    headers = {"Authorization": f"Bearer {bearer_token}"}

    stop_words = stopwords.words("english")  # get the list of English stopwords
    emoji_pattern = re.compile(
        "["
```

```python
for ticker in tickers:
    # Make a request to fetch the news articles for the given ticker
    url = f'https://finviz.com/quote.ashx?t={ticker}'
    response = requests.get(url, headers=headers)

    # Parse the HTML response to extract the news table
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    news_table = soup.find(id='news-table')

    # Loop through each row in the news table
    for row in news_table.findAll('tr')[:5]:
        # Get the news article timestamp and text
        date = row.td.text.strip()
        div = row.find('div', class_='news-link-container')
        if div:
            td_text = div.find('div', class_='news-link-left').text
            text_cell = div.find('div', class_='news-link-left').find('a')
```

## *Data Transformation and Storage Phase:*

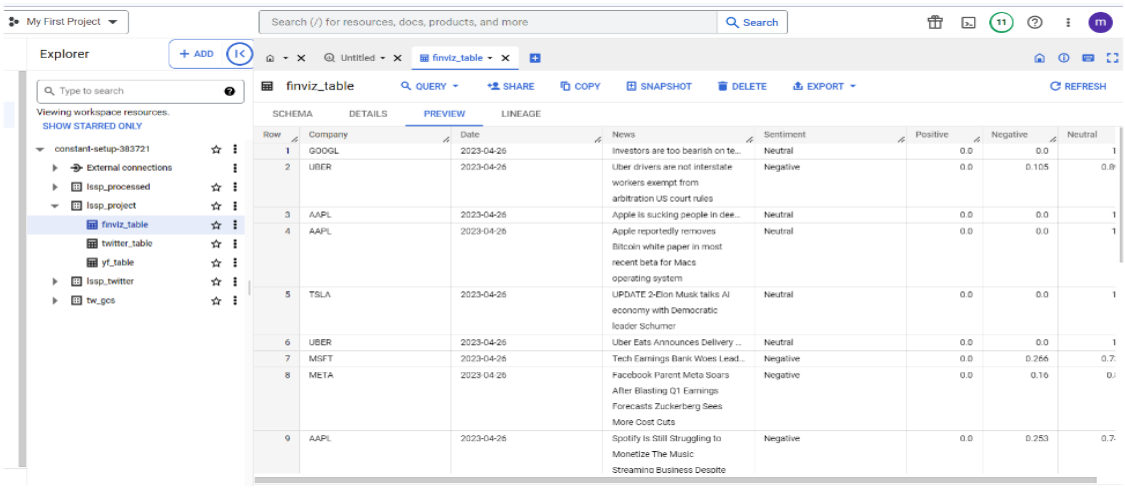
The transformed data is stored in BigQuery tables as part of our Dataflow jobs. We have three separate Dataflow jobs, each corresponding to one of the data sources. The Dataflow jobs use Apache Beam to apply various PTransforms to the data in GCS buckets, performing tasks like computing moving averages for Yahoo Finance data and performing sentiment analysis on Twitter and news data. This approach ensures low latency and reliability in terms of data fetching and processing.


Job info

| | | |
|---|---|---|
| bytes_written | 847,056 | InsertIntoBigQuery/.../ParMultiDo(BatchAndInsertElements) |

**Pipeline options**

| | |
|---|---|
| appName | TextToBigQueryStreaming |
| bigQueryLoadingTemporaryDirectory | gs://lssp_bucket/temp1/ |
| filesToStage | [/export/hda3/borglet/remote_hdd_fs_dirs/0.rapid.runner-tdnvt5aq-4y3w-rqmt-7ajq<br>...<br>SEE ALL |
| gcpTempLocation | gs://lssp_bucket/temp2/ |
| inputFilePattern | gs://lssp_bucket/twitter_data/* |
| javascriptTextTransformFunctionName | transform |
| javascriptTextTransformGcsPath | gs://lssp_bucket/twitter_datafiles/twitter_puller_transform_udf.js |
| jobName | twitter_puller |
| JSONPath | gs://lssp_bucket/twitter_datafiles/twitter_pullerbg_schema.json |
| labels | {goog-dataflow-provided-template-name=stream_gcs_text_to_bigquery, goog-datafl |
| outputTable | constant-setup-383721:lssp_project.twitter_table |
| pipelineUrl | gs://dataflow-templates-libraries/2023-04-11-00_RC00/pipeline-96VUXCsZvnadv7J |
| project | constant-setup-383721 |
| region | us-east1 |
| runner | org.apache.beam.runners.dataflow.DataflowRunner |
| sdkContainerImage | – |
| stagingLocation | gs://dataflow-templates-libraries/2023-04-11-00_RC00 |
| templateLocation | gs://dataflow-templates-us-east1/latest/Stream_GCS_Text_to_BigQuery |
| tempLocation | gs://lssp_bucket/temp2/ |
| userAgent | Apache_Beam_SDK_for_Java/2.46.0(JRE_11_environment) |




## *Front-End Phase:*

For the front-end dashboard, we use Streamlit to host an engaging user interface that displays real-time visualizations and analyses. The Streamlit dashboard connects to BigQuery to fetch the processed data and visualizes it in an interactive and user-friendly manner. This provides our users with a comprehensive view of the stock market trends, news sentiment, and public opinion, enabling them to make informed investment decisions.

Overall, our system architecture employs advanced cloud computing technologies like Cloud Functions, Dataflow, and BigQuery, to create an efficient pipeline for data processing, storage, and visualization. By seamlessly integrating these components, we are able to collect, process, store, and visualize data from multiple sources, providing valuable insights to our users in a scalable and efficient manner.

# Streaming Concepts Used:

1. **Apache Beam in Cloud Dataflow:**
    a. Pipeline: A Pipeline created to encapsulate entire data transformation workflow.
    b. PCollection: The incremental data for all 3 data sources are read in as bounded PCollections on which transformations are done.
    c. PTransform: The various PTransforms used include
        i. Yahoo Finance data:
            1. DatatypeConversion: To convert default string datatypes of each column to the correct dataype mapping.
            2. ComputeMovingAverages: 15 day moving averages were calculated to smoothen out the financial irregularities and help the user understand the general trend.
            3. WriteToBigQueryTransform: This PTransform writes the processed Yahoo Finance data to a BigQuery table.

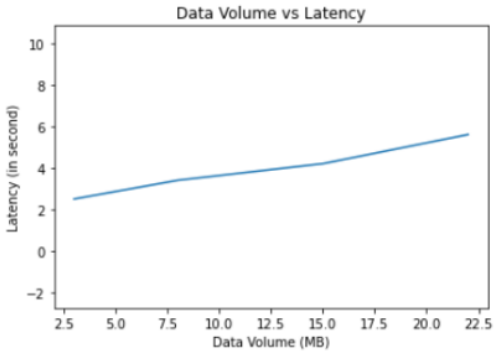
        Below transformations were done in Python and will be migrated to Beam in the future:
        ii. Twitter data: Functions Implemented currently in python
            1. PreprocessTweets: clean tweets using beam.Map() and regular expressions to remove URLs, mentions, hashtags, punctuation, change to lowercase. [Future Scope]
            2. SentimentAnalysisFunction : using sentimentAnalyzer, get the sentiment scores for each cleaned tweet.
        iii. News:
            1. PreprocessNews: clean news using beam.Map() and regular expressions to remove URLs, mentions, hashtags, punctuation.
            2. SentimentAnalysisFunction : using sentimentAnalyzer, get the sentiment scores for each cleaned tweet.
2. **Windowing:** Sliding Window: The 15 day moving average for all Yahoo Finance metrics were computed to smoothen out the trends and get a clearer representation for the user.
3. **Future Scope Items**
    i. Load Shedding: If a continuous stream for news and twitter data is setup then for computational optimization, periodic data points can be shed as each and every tweet will not have substantial value but their averaged out effect can help understand the overall social media and conventional media sentiments.
    ii. Operator Reordering: As the quantity of data scales, we can filter yahoo finance data on the date range before computing moving averages.
    iii. Fission can be used to split the loop over tickers into smaller chunks, which can be executed in parallel on multiple workers. This can be done using the Partition transform, which splits the input PCollection into smaller PCollections processed in parallel.

# Data Volume vs Latency Study

Latency is an important metric in cloud computing that refers to the time delay between a user's request for data and the data being received or displayed. In our experiment, we calculated latency by measuring the time between the GCS bucket being populated and the BigQuery table being populated. We conducted this measurement for different data sizes by changing the time window of the pulled data.

Based on the results obtained, we observed that there was no substantial change in latency when the size of the data being processed was increased. The measured latency values for different data sizes are presented in the table above.

| Volume of Data (MB) | Latency (GCS to Dataflow) |
|---|---|
| 3 MB | 2.8 seconds |
| 8 MB | 3.4 seconds |
| 15 MB | 4.2 seconds |
| 22 MB | 4.7 seconds |




As shown in the table, the latency values varied slightly as the volume of data increased. However, the increase in latency was not significant, indicating that the system can efficiently process and store data of varying sizes with minimal impact on latency. These observations suggest that our system architecture is well-suited for processing and storing large volumes of data in real-time without compromising performance or user experience.

# Future Work

- Implement optimization algorithms such as load shedding and operator reordering in order to improve efficiency of the entire process
- Set up the entire textual data transformation pipeline including preprocessing and sentiment analysis on Apache Beam
- Automate the orchestration of the pipeline using CRON job (Cloud Scheduler)
- In addition to the current data sources (Twitter API, Yahoo finance API, and Stock news API), the platform could integrate more data sources such as financial filings, insider trading data, and economic indicators.
- Improve the user interface platform by allowing users to customize the dashboard based on their preferences and interests, allowing them to track specific stocks or sectors that they are interested in.
- Identify machine learning use cases to reconcile current stock trends with social media and conventional media sentiments to recommend investment strategies to the users.