Date: 12/12/2019
Course: Engineering INFO7250 Big Data Systems
Name: Pramod Nagare
NUID: 001858910
Email: nagare.p@husky.ney.edu
Project: Streaming data pipeline for real-time analytics

Highlights:

- GCP Streaming and Batch Processing data analytical pipeline
- Bloom filter
- Hadoop cluster setup
- Google data explorer interactive dashboard
- Optimization (Combiner, distributed caches, cluster, compact data type)
- Multithread/multiprocessor scripts for data preparation

Dataset:

- https://registry.opendata.aws/amazon-reviews/
- https://github.com/awslabs/open-data-registry/blob/master/datasets/amazon-reviews.yaml
- https://s3.amazonaws.com/amazon-reviews-pds/readme.html
- 34+ GB Dataset .gzip format
- 130+ millions amazon customer reviews

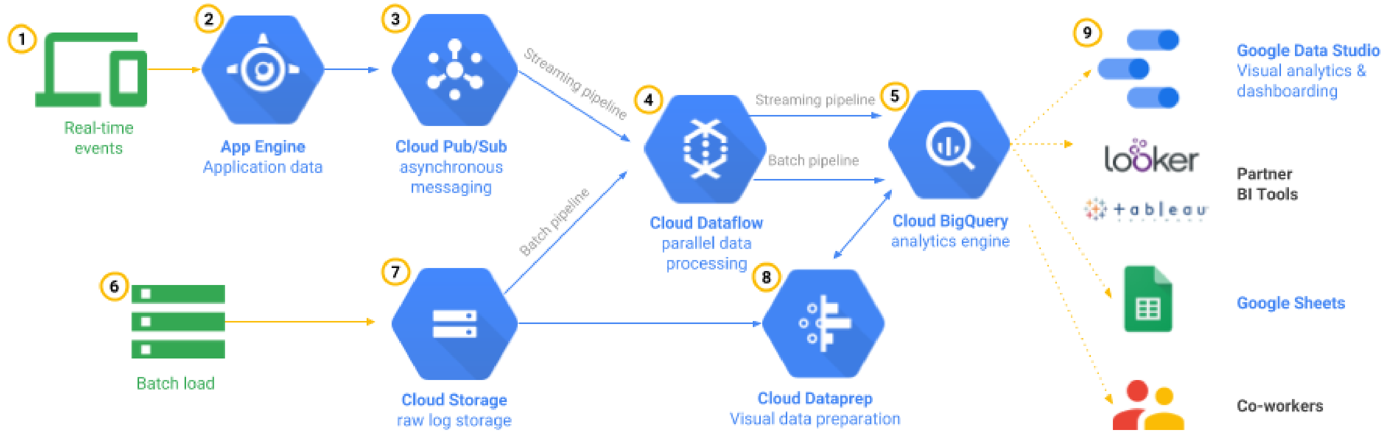
Task 1: Data download and extraction

Implementation:

1. Downloading AWS Customer review data from source (AWS S3 bucket)
   - Data Size: 130+ millions of rows, ~80GB of data over 51 files
   - More details can be found on index.txt file
   - Developed parallel processing python script
2. Extracting downloaded .gzip file into tsv
   - Developed parallel processing python script

Task 2: Building streaming data analytics pipeline

Implementation:



1. Architecture implementation:
    - Created streaming data analytical pipeline using GCP pub/sub and python
    - Developed a multi-process python script for data upload to GCP Big Query
    - This script can take .tsv.gz as an input, we are saving time of extracting multiple files
2. Data querying using Big Query:
    - Implemented simple queried to explore Big Query

Task 3: Real Time data visualization

Implementation:

1. Developed interactive dashboard using Google Data explorer
    - Created custom fields for detail data analysis
    - Data visualization using interactive graphs
    - Export and shared the dashboard

Task 4: Bloom filter

Implementation:

1. Designed and developed bloom filter for stop word, positive and negative word analysis
2. Tried importing and exporting bloom filter using pickle file in python and data output stream in java

Task 5: Map Reduce for Sentiment Analysis

Implementation:

1. Created map-reduce jobs in python on Hadoop using Hadoop streaming jar
2. Tested map-reduce on local machine
3. Implemented both MR jobs with bloom filter and without bloom filter
4. Optimized the MR jobs with Hadoop best practices
    - Using compact data type for the job
    - Using combiner
    - Running MR jobs on Hadoop cluster on GCP and AWS
    - Implemented distributed caches

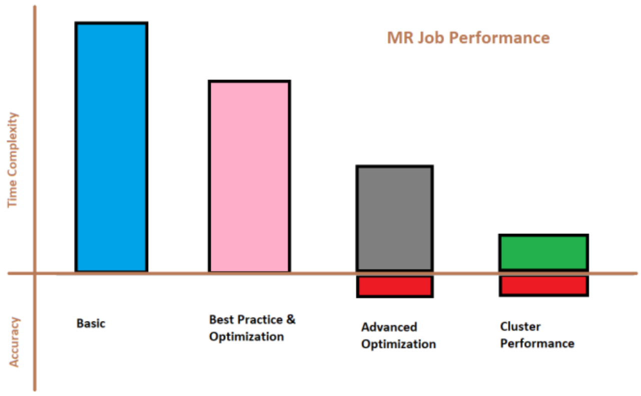
Task 6: Hadoop Cluster setup on GCP

Implementation:

1. Created a GCP Hadoop cluster with 2 worker nodes and 1 master
2. Submitted MR jobs on cluster using GCP utility
3. Implemented persistent storage for MR jobs
4. Ran some MR jobs on cluster with SSH to master node
5. Accessed the Hadoop dashboard for the job monitoring

Task 7: Conclusion

- **Parallel processes for file download are limited by internet speed**
- **Parallel processes for file extraction are limited by disk I/O**
- **MR Job Performance observation**



Best Practices:

1. Use compact / optimize data type in MR jobs
2. Use of combiners
3. Use file compressions format (ex. LZO)
4. Distributed caching
5. Multiple mapper and reducer

Advanced Optimization:

1. Bloom filter