Advanced Distributed Data Processing with PySpark

Tags: *Apache Spark | TF-IDF | Spark SQL | Spark Streaming | Log Analysis | Relevance Ranking*

🔍 Project Overview

This multi-part project demonstrates scalable data engineering and streaming analytics using Apache Spark. It includes offline TF-IDF ranking, document retrieval using term relevance, and real-time server log processing using structured streaming. The focus was on working with unstructured text data and live server logs at scale, combining batch and streaming workflows to simulate real-world scenarios in large, distributed environments.

📁 Modules

📘 1. TF-IDF Computation and Search Ranking using PySpark

Goal: Compute TF-IDF scores for a large collection of documents (e.g., Shakespeare, Austen) and rank documents based on term relevance.

Key Steps:

- Cleaned and tokenized document content stored in S3 using PySpark.
- Counted term frequencies and document frequencies across all files.
- Computed TF-IDF for each (doc_id, term) and ranked documents by relevance.
- Implemented custom query-based document scoring using normalized TF-IDF relevance.

Output:

- Top and bottom terms by TF-IDF weight
- Top 5 most relevant documents per query
- Functionality to rank based on any user query

📗 2. Real-Time Log Stream Analysis with Spark Structured Streaming

Goal: Process incoming server logs in real time and generate two reports: SEV2 volume and SEV0 error log.

Input Format: serverID,severity,timestamp

Key Features:

- **SEV2 Volume Report**: Tracks the number of SEV2 events per time unit for each server, updated incrementally in memory.
- **SEV0 Log Report**: Appends critical SEV0 events (fatal errors) to a persistent log stored as CSV in S3.
- Files are streamed into an S3 “live” folder to simulate real-time log ingestion.

Technologies Used:

- Spark Structured Streaming
- In-memory sinks (SEV2)
- File-based sink for persistent SEV0 logging
- S3 simulation for log delivery

Output:

- Running volume report (updated)
- Appended SEV0 event log saved to S3

📙 3. Spark TF-IDF Search + Streaming: Unified Notebook Architecture

Goal: Integrate document indexing and live querying into a single interactive pipeline.

Highlights:

- Full Spark Notebook environment with reusable functions:
    - indexDocuments() builds a TF-IDF index from a document corpus.
    - relevance(query, tfidf) ranks documents on-the-fly.
- Combined static indexing and live query logic
- Modular functions for reproducibility and experimentation

Real-World Use Case:

- Foundation for a lightweight **search engine backend** using PySpark.
- Seamless integration with S3-hosted document corpora for cloud-scale text search.

🛠 Tools & Technologies

- **Apache Spark** (RDD, SQL, Streaming APIs)
- **PySpark** for batch and real-time pipelines
- **S3** for document storage and simulated log ingestion
- **SparkSQL** for relevance ranking and metric aggregation

📊 Key Deliverables

- TF-IDF index for unstructured text data
- Real-time log analysis and error tracking dashboard
- Relevance-scored search results for any input query
- Spark notebook that integrates indexing, query, and reporting workflows

📎 How to Run

1. Upload text corpus to an S3 bucket
2. Run the indexDocuments() function to compute TF-IDF
3. Query with relevance(query, tfidf_index)
4. For logs: copy log files to LogDataLive bucket and observe updated streaming outputs