Stack Overflow Data Analysis and Processing Report

By Zhenghan Nan(netId: zn2145)

Summary

This report describes how MapReduce was used to build a distributed architecture on which to deconstruct Stack Overflow post data. This architecture restructures the post data through profiling, cleaning and ingestion, aiming to fulfil big data processing demands.

1. Introduction

1. Overview

The objective of the project is to deal with and examine the Stack Overflow posts data using the MapReduce framework while ensuring the ability to work with very large datasets robustly. The implementation incorporates data profiling for quality control, cleaning for data homogenization, and ingestion for subsequent analysis.

2. Data Source Description

2.1 Source Overview

Data is sourced from the Stack Exchange Data Dump, specifically focusing on Stack Overflow posts. The schema includes:

Primary Tables:

- Posts
- Users
- Comments
- Tags

2.2 Schema Analysis

Data Features:

- Id (int): Unique identifier
- PostTypeId (tinyint): Type of post (1=Question, 2=Answer)
- ParentId (int): Reference to parent post for answers
- CreationDate (datetime): Post creation timestamp
- Score (int): Post score from voting
- ViewCount (int): Number of views

- Title (nvarchar): Post title
- Tags (nvarchar): Associated tags
- Body (nvarchar): Main content
- Additional metadata fields (e.g., OwnerUserId, CommentCount)

2.3 Data Sample

```
Id,PostTypeId,ParentId,CreationDate,Score,ViewCount,Title,Tags,Body,OwnerUserId,AnswerCount,CommentCount,FavoriteCount
74972583,2,74961937,2023/1/1 00:00,1,0,unknown,unknown,"check if the case matters, use java...",6309,0,2,0
74972584,2,74972430,2023/1/1 00:00,3,0,unknown,unknown,"As an alternative you can create...",17949945,0,0,0
```

3. Implementation

3.1 Data Profiling

Mapper Implementation

1. Completeness Analysis
2. Data Type Validation
3. Content Quality Assessment
4. Value Distribution Check

Reducer Implementation

The reducer aggregates statistics about:

- Missing value counts
- Invalid data types
- Score distribution
- Data quality metrics

3.2 Data Cleaning

Cleaning Process

1. **Text Normalization**
    - HTML tag removal
    - Code block extraction
    - Special character handling
    - Whitespace normalization
2. **Missing Value Handling**

- Critical fields (Id, PostTypeId, CreationDate): Records dropped if missing
- Non-critical fields: Default values assigned
- Numeric fields: Defaulted to 0
- Text fields: Defaulted to "unknown"

3. **Data Type Standardization**
   - Score validation and conversion
   - Date format standardization
   - Tag format normalization

3.3 Data Ingestion

Shell commands for data processing:

```
# Upload data to HDFS
hdfs dfs -put stackoverflow_data.csv /user/yourusername/stackoverflow_data.csv

# Run data profiling
hadoop jar /usr/lib/hadoop/hadoop-streaming.jar \
    -D mapreduce.job.name="StackOverflow Data Profiling" \
    -input /user/yourusername/stackoverflow_data.csv \
    -output /user/yourusername/stackoverflow_profile_output \
    -mapper ./data_profiling_mapper.py \
    -reducer ./data_profiling_reducer.py \
    -file ./data_profiling_mapper.py \
    -file ./data_profiling_reducer.py
    -cmdenv PYTHONIOENCODING=utf-8

# Run data cleaning
hadoop jar /usr/lib/hadoop/hadoop-streaming.jar \
    -D mapreduce.job.name="StackOverflow Data Cleaning" \
    -input /user/yourusername/stackoverflow_data.csv \
    -output /user/yourusername/stackoverflow_clean_output \
    -mapper ./data_cleaning_mapper.py \
    -reducer ./data_cleaning_reducer.py \
    -file ./data_cleaning_mapper.py \
    -file ./data_cleaning_reducer.py
    -cmdenv PYTHONIOENCODING=utf-8
```

4. Conclusion

The applied MapReduce pipeline copes with the Stack Overflow data in bulk, producing uniform and tidy dataset for the purpose of analysis.

Appendix: Code Documentation

MapReduce Implementation Details

The implementation consists of four main Python scripts:

1. data_profiling_mapper.py
2. data_profiling_reducer.py
3. data_cleaning_mapper.py
4. data_cleaning_reducer.py