Spark case study code:

package com.intellipaat.apps.logs.chapter1;

import com.intellipaat.apps.logs.ApacheAccessLog;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.Function2;

import scala.Tuple2;

import java.io.Serializable;

import java.util.Comparator;

import java.util.List;

/\*\*

\* The LogAnalyzer takes in an apache access log file and

\* computes some statistics on them.

\*

\* Example command to run:

\* % ${YOUR\_SPARK\_HOME}/bin/spark-submit

\* --class "com.intellipaat.apps.logs.chapter1.LogsAnalyzer"

\* --master local[4]

\* target/log-analyzer-1.0.jar

\* ../../data/apache.accesslog

\*/

public class LogAnalyzer {

private static Function2<Long, Long, Long> SUM\_REDUCER = (a, b) -> a + b;

private static class ValueComparator<K, V>

implements Comparator<Tuple2<K, V>>, Serializable {

private Comparator<V> comparator;

public ValueComparator(Comparator<V> comparator) {

this.comparator = comparator;

}

@Override

public int compare(Tuple2<K, V> o1, Tuple2<K, V> o2) {

return comparator.compare(o1.\_2(), o2.\_2());

}

}

public static void main(String[] args) {

// Create a Spark Context.

SparkConf conf = new SparkConf().setAppName("Log Analyzer");

JavaSparkContext sc = new JavaSparkContext(conf);

// Load the text file into Spark.

if (args.length == 0) {

System.out.println("Must specify an access logs file.");

System.exit(-1);

}

String logFile = args[0];

JavaRDD<String> logLines = sc.textFile(logFile);

// Convert the text log lines to ApacheAccessLog objects and cache them

// since multiple transformations and actions will be called on that data.

JavaRDD<ApacheAccessLog> accessLogs =

logLines.map(ApacheAccessLog::parseFromLogLine).cache();

// Calculate statistics based on the content size.

// Note how the contentSizes are cached as well since multiple actions

// are called on that RDD.

JavaRDD<Long> contentSizes =

accessLogs.map(ApacheAccessLog::getContentSize).cache();

System.out.println(String.format("Content Size Avg: %s, Min: %s, Max: %s",

contentSizes.reduce(SUM\_REDUCER) / contentSizes.count(),

contentSizes.min(Comparator.naturalOrder()),

contentSizes.max(Comparator.naturalOrder())));

// Compute Response Code to Count.

List<Tuple2<Integer, Long>> responseCodeToCount =

accessLogs.mapToPair(log -> new Tuple2<>(log.getResponseCode(), 1L))

.reduceByKey(SUM\_REDUCER)

.take(100);

System.out.println(String.format("Response code counts: %s", responseCodeToCount));

// Any IPAddress that has accessed the server more than 10 times.

List<String> ipAddresses =

accessLogs.mapToPair(log -> new Tuple2<>(log.getIpAddress(), 1L))

.reduceByKey(SUM\_REDUCER)

.filter(tuple -> tuple.\_2() > 10)

.map(Tuple2::\_1)

.take(100);

System.out.println(String.format("IPAddresses > 10 times: %s", ipAddresses));

// Top Endpoints.

List<Tuple2<String, Long>> topEndpoints = accessLogs

.mapToPair(log -> new Tuple2<>(log.getEndpoint(), 1L))

.reduceByKey(SUM\_REDUCER)

.top(10, new ValueComparator<>(Comparator.<Long>naturalOrder()));

System.out.println(String.format("Top Endpoints: %s", topEndpoints));

// Stop the Spark Context before exiting.

sc.stop();

}

}