# **Q1 - JDK Implementation of Phone Number service**

## **General Idea**

## Create a stream over the input file: "data.txt"

## Do some validations with the phone number of each line

## Determine if the phone number is a valid phone number using Google's libphonenumber

## Determine if the phone number is a cell phone by Call Twilio Lookup API

## Use LRUCache to save the phone number which determined by Twilio Lookup API

## Write the result of the calculation to a global Set<PhoneNumber> result

## Return the running time of the calculation as its result.

## Sum up all the running times to find out the total CPU time consumed.

## **Result**

Start Sequentially processing 1 round stream

Cell Phone Num: 4

Cores: 8

CPU time: 15.89 s

Real time: 15.90 s

CPU utilization: 12.49%

Start Parallel processing 1 round stream

Cell Phone Num: 4

Cores: 8

CPU time: 15.08 s

Real time: 5.48 s

CPU utilization: 34.41%

Start Sequentially processing 2 round stream

Cell Phone Num: 4

Cores: 8

CPU time: 16.48 s

Real time: 16.48 s

CPU utilization: 12.50%

Start Parallel processing 2 round stream

Cell Phone Num: 4

Cores: 8

CPU time: 14.13 s

Real time: 4.63 s

CPU utilization: 38.12%

Validated Cell Numbers

3605532393

2025550156

2025550197

2025550169

## **Source Code**

//Parallel stream processing

package com.hustbill;

import static java.util.concurrent.TimeUnit.SECONDS;

import static java.util.stream.StreamSupport.stream;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.HashSet;

import java.util.Set;

import java.util.stream.Stream;

import com.google.i18n.phonenumbers.PhoneNumberUtil;

import com.google.i18n.phonenumbers.Phonenumber.PhoneNumber;

import com.twilio.sdk.LookupsClient;

import com.twilio.sdk.TwilioRestException;

public class ParallelPhoneValidator {

static double sink;

static Set<PhoneNumber> result = new HashSet<PhoneNumber>();

// Find your Account Sid and Token at twilio.com/user/account

public static final String ACCOUNT\_SID = "ACbdc579bf50d3995b262a44f24d6f1491";

public static final String AUTH\_TOKEN = "c330fec63941a84f36729bc287c1a0ff";

public static void main(String[] args) throws IOException {

final Path inputPath = Paths.get("src/main/resources/data.txt");

for (int i = 1; i < 3; i++) {

System.out.printf("Start Sequentially processing %d round stream \n", i);

measureProcessing(Files.lines(inputPath));

System.out.printf("Start Parallel processing %d round stream \n", i);

measureProcessing(withBatchSize(Files.lines(inputPath), 10));

}

System.out.println("Validated Cell Numbers");

for (PhoneNumber phoneNum : result) {

long nationalNumber = phoneNum.getNationalNumber();

System.out.println(" " + nationalNumber );

}

}

private static void measureProcessing(Stream<String> input) throws IOException {

final long start = System.nanoTime();

try (Stream<String> lines = input) {

final long totalTime = lines.parallel().mapToLong(ParallelPhoneValidator::processLine).sum();

final double cpuTime = totalTime, realTime = System.nanoTime() - start;

final int cores = Runtime.getRuntime().availableProcessors();

System.out.println(" Cell Phone Num: " + result.size());

System.out.println(" Cores: " + cores);

System.out.format(" CPU time: %.2f s\n", cpuTime / SECONDS.toNanos(1));

System.out.format(" Real time: %.2f s\n", realTime / SECONDS.toNanos(1));

System.out.format("CPU utilization: %.2f%%\n\n", 100.0 \* cpuTime / realTime / cores);

}

}

/\*

\* Determine if the phone number is a valid phone number using Google's

\* libphonenumber

\*

\* Google's common Java, C++ and JavaScript library for parsing, formatting,

\* and validating international phone numbers.

\*/

private static long processLine(String usaNumberStr) {

final long localStart = System.nanoTime();

PhoneNumberUtil phoneUtil = PhoneNumberUtil.getInstance();

LRUCache cache = new LRUCache(1000);

try {

PhoneNumber usaNumberProto = phoneUtil.parse(usaNumberStr, "US");

boolean isValid = phoneUtil.isValidNumber(usaNumberProto);

if (isValid) {

long key = usaNumberProto.getNationalNumber();

if (cache.get(key) != -1) {

if (cache.get(key) == 1) { // 1 - MOBILE

result.add(usaNumberProto);

}

} else {

if (twilioLookup(usaNumberProto) == "MOBILE") {

result.add(usaNumberProto);

cache.set(key, 1);

} else {

cache.set(key, 0); // 0 - LANDLINE

}

}

}

} catch (Exception e) {

System.err.println("NumberParseException was thrown: " + e.toString());

}

return System.nanoTime() - localStart;

}

public static <T> Stream<T> withBatchSize(Stream<T> in, int batchSize) {

return stream(new FixedBatchSpliterator<>(in.spliterator(), batchSize), true);

}

/\*

\* Get the type of a phone : Landline or Mobile by Twilio Lookup API.

\*

\* Twilio Lookup is a simple REST API with a ton of utility. Twilio

\* https://www.twilio.com/lookup determine whether a phone is a landline or

\* can receive text messages, and even discover information about the

\* carrier associated with that phone number.

\*/

public static String twilioLookup(PhoneNumber number) throws TwilioRestException {

String strNum = String.valueOf(number.getNationalNumber());

LookupsClient client = new LookupsClient(ACCOUNT\_SID, AUTH\_TOKEN);

com.twilio.sdk.resource.instance.lookups.PhoneNumber phoneNumber = client.getPhoneNumber(strNum, true);

return phoneNumber.getType().toString();

}

}