

# 1. Laboratorijska vježba - ROVKP

---

Vinko Kolobara  
2. travnja 2017.

## 1 ZADATAK: UPOZNAVANJE SA HADOOP GROZDOM

1. Hadoop grozd sastoji se od 8 računala.
2. IP adresa imenskog čvora je 10.19.4.52
3. Podatkovnih čvorova ima 7 i svi su u funkciji.
4. Ukupni spremišni kapacitet grozda je 3.0 TiB.
5. Postavljena veličina HDFS bloka je 128 MiB.

## 2 ZADATAK: RAD SA DATOTEKAMA U HADOOP GROZDU

Listing 1: Programski kod 2. zadatka

```
package hr.vinko.rovkv.lab1.zad2;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.FileVisitResult;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.SimpleFileVisitor;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.concurrent.atomic.AtomicInteger;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;

public class GutenbergToTxt {
    public static void main(String[] args) throws IOException,
        URISyntaxException {

        Configuration conf = new Configuration();

        final AtomicInteger lineCounter = new AtomicInteger(0);
        final AtomicInteger fileCounter = new AtomicInteger(0);

        org.apache.hadoop.fs.Path path = new org.apache.hadoop.fs.Path("/
            user/rovkv/vkolobara/gutenberg_books.txt");

        FileSystem hdfs = FileSystem.get(new URI("hdfs://cloudera2:8020"),
            conf);

        long startTime = System.currentTimeMillis();

        try (BufferedWriter out = new BufferedWriter(new OutputStreamWriter(
            hdfs.create(path)))) {
```

```

Files.walkFileTree(Paths.get("gutenberg"), new SimpleFileVisitor<
    Path>() {
    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes
        attrs) throws IOException {

        try (BufferedReader in = new BufferedReader(new FileReader(
            file.toFile())) {
            String line;

            while ((line = in.readLine()) != null) {
                out.write(line + "\n");
                lineCounter.getAndIncrement();
            }

        }

        fileCounter.getAndIncrement();
        return FileVisitResult.CONTINUE;
    }
});
}

System.out.println("DURATION:_ " + (System.currentTimeMillis() -
    startTime) + "ms.");
System.out.println("LINE_NUMBERS:_ " + lineCounter);
System.out.println("FILES_READ:_ " + fileCounter);

hdfs.close();

Configuration conf = new Configuration();

}
}

```

Listing 2: Ispis programa

```

DURATION: 5776ms
LINE NUMBERS: 8481553
FILES READ: 594

```

Veličina ciljne datoteke iznosi 418.242.611 byte-a, pohranjena je u  $4 \cdot 3 = 12$  blokova.

### 3 ZADATAK: SERIJALIZACIJA OBJEKATA U HADOOP GROZDU

Listing 3: Programski kod 2. zadatka

```
package hr.vinko.rovkv.lab1.zad3;

import java.io.IOException;
import java.util.Random;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.SequenceFile;

public class SensorGenerator {

    private final static int NUM_READINGS = 100_000;

    private final static int MIN_SENSOR_ID = 1;
    private final static int MAX_SENSOR_ID = 100;

    private final static double MIN_SENSOR_VALUE = 0.00;
    private final static double MAX_SENSOR_VALUE = 99.99;

    private final static String OUT_FILE_PATH = "/user/rovkv/vkolobara/
        objects.bin";

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

        Random rand = new Random();

        Configuration conf = new Configuration();

        Path outputPath = new Path(OUT_FILE_PATH);

        SequenceFile.Writer writer = SequenceFile.createWriter(conf,
            SequenceFile.Writer.file(outputPath),
            SequenceFile.Writer.keyClass(IntWritable.class), SequenceFile.
                Writer.valueClass(DoubleWritable.class));

        for (int i = 0; i < NUM_READINGS; i++) {
            IntWritable key = new IntWritable(rand.nextInt(MAX_SENSOR_ID -
                MIN_SENSOR_ID + 1) + MIN_SENSOR_ID);
```

```

        DoubleWritable val = new DoubleWritable(rand.nextDouble() * (
            MAX_SENSOR_VALUE - MIN_SENSOR_VALUE) + MIN_SENSOR_VALUE);
        writer.append(key, val);
    }

    writer.close();

    int[] sensorCounts = new int[MAX_SENSOR_ID - MIN_SENSOR_ID + 1];
    double[] sensorSums = new double[sensorCounts.length];

    SequenceFile.Reader reader = new SequenceFile.Reader(conf,
        SequenceFile.Reader.file(outputPath));

    IntWritable key = new IntWritable();
    DoubleWritable value = new DoubleWritable();

    while(reader.next(key, value)) {
        sensorCounts[key.get() - MIN_SENSOR_ID]++;
        sensorSums[key.get() - MIN_SENSOR_ID] += value.get();
    }

    reader.close();

    for (int i=0; i<sensorCounts.length; i++) {
        if (sensorCounts[i] > 0) {
            System.out.printf("Sensor_%d: %.6f\n", i + MIN_SENSOR_ID,
                sensorSums[i] / sensorCounts[i]);
        }
    }
}

```