Projekat 2

Sistemi za obradu i analizu velike količine podataka

Nemanja Petrović
1499

Stream obrada podataka - Spark

- Kafka Producer i Consumer su napisani u Pythonu
- Straming obrada podataka se vrši na 2 sekunde
- Za prvi zadatak se posmatraju vožnje sa polaznom stanicom sa id-jem 487
- Racuna se min, max, avg trajanje vožnje sa tom stanicom
- Za drugi zadatak se gledaju sve stanice i traži se koje su u datom periodu imale najviše vožnji
- Rezultat se upisuje u Cassandru

Izvršenje na Spark clusteru

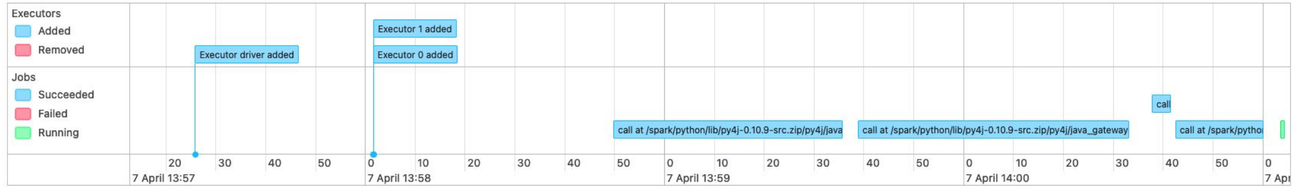
Spark 3.1.2 | Jobs | Stages | Storage | Environment | Executors | SQL | Structured Streaming | BigDataP2Nemanja application UI

Spark Jobs (?)

User: root
Total Uptime: 3.9 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 4

▾ Event Timeline

☐ Enable zooming



▾ Active Jobs (1)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 (kill)	2023/04/07 14:01:03	0.6 s	0/2	0/2 (1 running)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

▾ Completed Jobs (4)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442	2023/04/07 14:00:42	18 s	2/2	201/201
2	call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442	2023/04/07 14:00:37	4 s	2/2	2/2
1	call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442	2023/04/07 13:59:38	54 s	2/2	201/201
0	call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442 call at /spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:2442	2023/04/07 13:58:49	46 s	2/2	2/2

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Spark 3.1.2

Spark Master at spark://aa8c270665db:7077

URL: spark://aa8c270665db:7077
Alive Workers: 2
Cores in use: 8 Total, 0 Used
Memory in use: 13.5 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 4 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▾ Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20230407132509-192.168.0.10-38587	192.168.0.10:38587	ALIVE	4 (0 Used)	6.8 GiB (0.0 B Used)	
worker-20230407132509-192.168.0.11-41153	192.168.0.11:41153	ALIVE	4 (0 Used)	6.8 GiB (0.0 B Used)	

▾ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

▾ Completed Applications (4)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230407135724-0003	BigDataP2Nemanja	8	1024.0 MiB		2023/04/07 13:57:24	root	FINISHED	4.7 min

Spark 3.1.2 | Jobs | Stages | Storage | Environment | Executors | SQL | Structured Streaming | BigDataP2Nemanja application UI

Streaming Query

▾ Active Streaming Queries (1)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Name	Status	ID	Run ID	Start Time ▾	Duration	Avg Input /sec	Avg Process /sec	Latest Batch
<no name>	RUNNING	6fa39d38-4c0c-4413-96cc-0eba65ae7a75	e00b4b7a-308e-4e3d-b909-589a483ea1e9	2023/04/07 13:58:15	3 minutes 12 seconds	2.53	9.04	2

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Delovi koda za Spark aplikaciju

```python
if __name__ == '__main__':
    msgProducer = KafkaProducer(bootstrap_servers=['localhost:9092'],
                                value_serializer=lambda x: x.encode('utf-8'), api_version=(0,11,5))

    with open('oslo-bikes.csv') as csvFile:
        data = csv.DictReader(csvFile)
        for row in data:
            msgProducer.send('t2', json.dumps(row))
            msgProducer.flush()

            print('Message: ' + json.dumps(row))
            time.sleep(2)

    print('Kafka message producer done!')
```

Producer

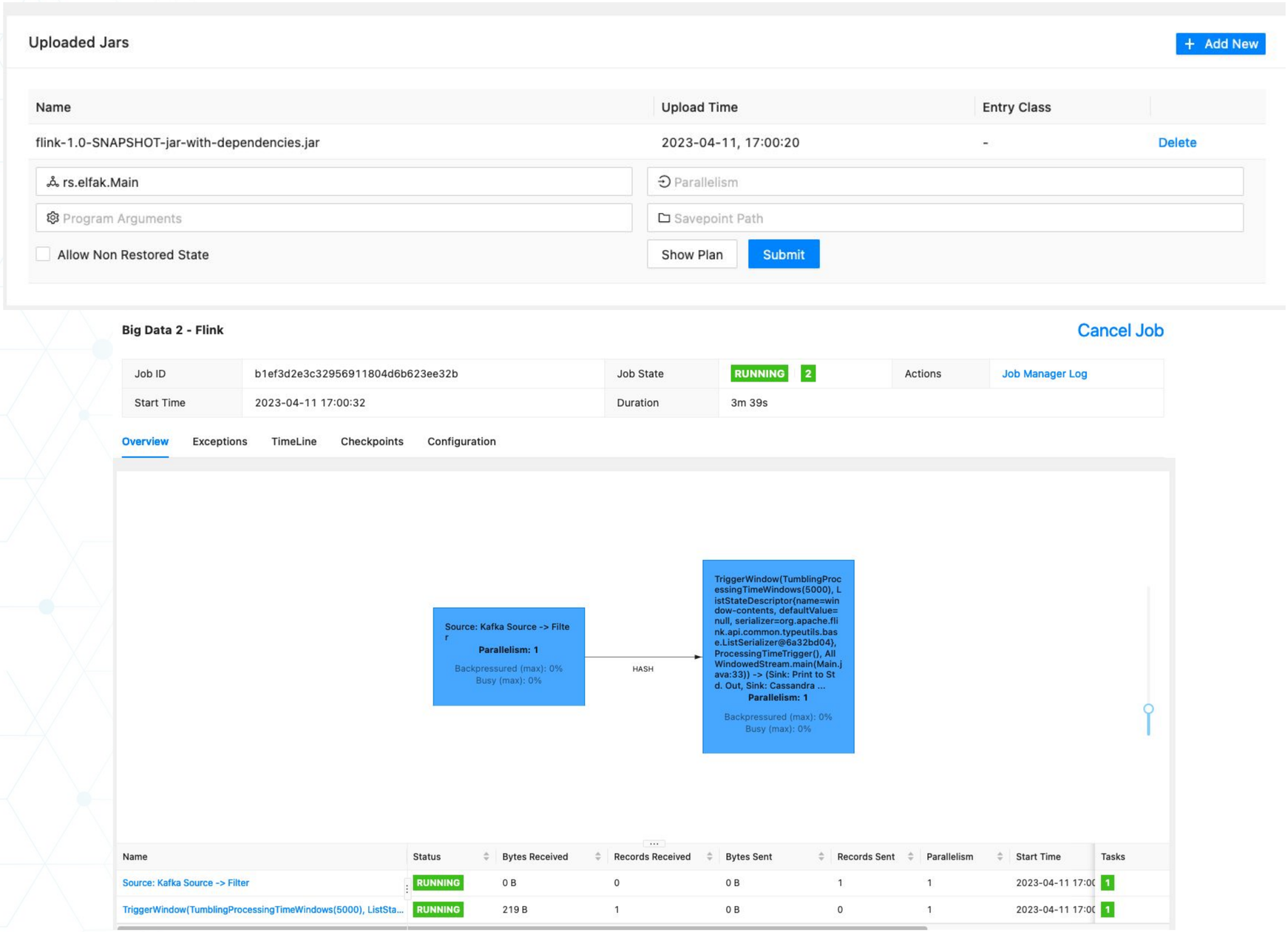
```python
sampleDataframe = (
    spark.readStream.format("kafka")
        .option("kafka.bootstrap.servers", kafka_url)
        .option("subscribe", kafka_topic)
        .option("startingOffsets", "earliest")
        .load()
).selectExpr("CAST(value as STRING)", "timestamp").select(
    from_json(col("value"), dataSchema).alias("sample"), "timestamp"
).select("sample.*")


sampleDataframe.writeStream \
    .option("spark.cassandra.connection.host", cassandra_host+':'+str(9042)) \
    .foreachBatch(lambda df, epoch_id: write_to_cassandra(df, epoch_id, start_station_id)) \
    .outputMode("update") \
    .trigger(processingTime=process_time) \
    .start().awaitTermination()
```

Čitanje stream-a

```python
def write_to_cassandra(df, epoch, station_id):
    print("Epoch " + str(epoch))
    zad1 = df
    zad2 = df
    zad1 = zad1.filter(
        zad1.start_station_id == station_id) \
        .agg(
        min(col("duration").cast('int')).alias('duration_min'), max(col("duration").cast('int')).alias('duration_max'),
        mean(col("duration").cast('int')).alias('duration_avg'),
        count("start_station_id").alias('num_of_rides')
    ).collect()

    zad2 = zad2.groupBy(
        zad2.start_station_name) \
        .agg(count('start_station_name').alias('num_of_rides')).sort(desc('num_of_rides')).take(3)

    if zad1[0]['duration_avg']:
        davg = zad1[0]['duration_avg']
        dmax = zad1[0]['duration_max']
        dmin = zad1[0]['duration_min']
        rides = zad1[0]['num_of_rides']

        cassandra_session.execute(f"""
                            INSERT INTO npetrovic_p2_keyspace.statistics(time, duration_avg, duration_max, duration_min, num_of_rides)
                            VALUES (toTimeStamp(now()), {davg}, {dmax}, {dmin}, {rides})
                            """)

    stations = ['Unknown', 'Unknown', 'Unknown']
    num_rides = [-1, -1, -1]
    for i, row in enumerate(zad2):
        stations[i] = row['start_station_name']
        num_rides[i] = row['num_of_rides']

    cassandra_session.execute(f"""
                    INSERT INTO npetrovic_p2_keyspace.popular_stations(time, start_station_name1, num_of_rides1, start_station_name2, num_of_rides2, sta
                    VALUES (toTimeStamp(now()), '{stations[0]}', {num_rides[0]}, '{stations[1]}', {num_rides[1]}, '{stations[2]}', {num_rides[2]})
                    """)
```

Analiza i upis u Cassandru

Izvršenje na Flink clusteru



Delovi koda Flink aplikacije

```java
@Table(keyspace = "flink", name = "stats")
public class TripDurationStatistics {
    3 usages
    @Column(name = "time")
    public Date  time;
    3 usages
    @Column(name = "max_duration")
    public Float max_duration;
    3 usages
    @Column(name = "min_duration")
    public Float min_duration;
    3 usages
    @Column(name = "avg_duration")
    public Float avg_duration;
    3 usages
    @Column(name = "station1")
    public String station1;
    3 usages
    @Column(name = "num_rides1")
    public Integer num_rides1;
    3 usages
    @Column(name = "station2")
    public String station2;
    3 usages
    @Column(name = "num_rides2")
    public Integer num_rides2;
    3 usages
    @Column(name = "station3")
    public String station3;
    3 usages
    @Column(name = "num_rides3")
    public Integer num_rides3;
```

Klasa koja predstavlja Cassandra teabelu

```java
@Override
public void process(ProcessAllWindowFunction<OsloRide, TripDurationStatistics, TimeWindow>.Context context,
                    Iterable<OsloRide> elements, Collector  <TripDurationStatistics> out) throws Exception {
    float sum = 0;
    float max = 0;
    float min = 50000;
    float avg = 0;
    String station1 = "";
    int numRides1 = 0;
    String station2 = "";
    int numRides2 = 0;
    String station3 = "";
    int numRides3 = 0;
    float count = 0;

    HashMap<String, Integer> popular = new HashMap<>();

    for (OsloRide msg : elements) {
        count ++;
        sum += msg.duration;
        if (msg.duration > max)
            max = msg.duration;
        if (msg.duration < min)
            min = msg.duration;
        if(!popular.containsKey(msg.end_station_name)) {
            popular.put(msg.end_station_name, 1);
        } else {
            int newValue = popular.get(msg.end_station_name) + 1;
            popular.replace(msg.end_station_name, newValue);
        }
    }
    avg = sum / count;

    if (popular.keySet().size() > 0) {
        station1 = (String) popular.keySet().toArray()[0];
        numRides1 = popular.get(station1);
    }
    if (popular.keySet().size() > 1) {
        station2 = (String) popular.keySet().toArray()[1];
        numRides2 = popular.get(station2);
    }
    if (popular.keySet().size() > 2) {
        station3 = (String) popular.keySet().toArray()[2];
        numRides3 = popular.get(station3);
    }


    Date date = new Date();

    TripDurationStatistics res = new TripDurationStatistics(date, max, min, avg, station1, numRides1, station2, numRides2, station3, numRides3);
//      System.out.println("final res ---> " + res);
    out.collect(res);
```

Analiza podataka

```java
KafkaSource<OsloRide> source = KafkaSource.<OsloRide>builder()
        .setBootstrapServers("kafka:29092")
        .setTopics("flink")
        .setStartingOffsets(OffsetsInitializer.earliest())
        .setDeserializer(KafkaRecordDeserializationSchema.valueOnly(schema))
        .build();

DataStream<OsloRide> ds = env.fromSource(source, WatermarkStrategy.noWatermarks(), sourceName: "Kafka Source").
        filter((FilterFunction<OsloRide>) value -> (value.start_station_id.equals("480")));
DataStream<TripDurationStatistics> res = ds.windowAll(TumblingProcessingTimeWindows.of(Time.seconds(2)))
        .process(new StatisticsStream());
res.print();
CassandraSink.addSink(res)
        .setMapperOptions(() -> new Mapper.Option[] {
                Mapper.Option.saveNullFields( enabled: true)
        })
        nemanja
        .setClusterBuilder(new ClusterBuilder() {
            no usages
            private static final long serialVersionUID = 1L;

            no usages  nemanja
            @Override
            protected Cluster buildCluster(Cluster.Builder builder) {

                return builder.addContactPoints( ...addresses: "cassandra-node").withPort(9042).build();
            }
        })
        .build();
env.setParallelism(2);
env.execute( jobName: "Big Data 2 - Flink");
```

Citanje podataka, prosleđivanje Flinku na anlizu i upis rezultata u Cassandru

Performanse aplikacije

Obe aplikacije su pokrenute na docker clusteru na istoj mašini, Flink aplikacija je imala bolje performanse, analiza se izvršavala brže nego kroz Spark.
S obzirom da je Spark imao problema na ovoj mašini zbog procesora i da zbog tog radi usporeno to je ujedno i jedan od razloga zašto Flink radi brže u ovom slučaju

BADIN