

Αναφορά Εργαστηριακής Άσκησης

Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου

Εργαστηριακή Άσκηση 2023/24

Όνομα	Επώνυμο	ΑΜ
Δημήτριος	Στασινός	1084643
Βασίλειος-Μάριος	Κουρτάκης	1090061

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμπει σε αυτήν, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για το συγκεκριμένο μάθημα/σεμινάριο/πρόγραμμα σπουδών.

Έχω ενημερωθεί ότι σύμφωνα με τον εσωτερικό κανονισμό λειτουργίας του Πανεπιστημίου Πατρών άρθρο 50§6, τυχόν προσπάθεια αντιγραφής ή εν γένει φалκίδευσης της εξεταστικής και εκπαιδευτικής διαδικασίας από οιονδήποτε εξεταζόμενο, πέραν του μηδενισμού, συνιστά βαρύ πειθαρχικό παράπτωμα.

Υπογραφή

21 / 6 / 2024

Υπογραφή

21 / 6 / 2024

Συνημμένα αρχεία κώδικα

Μαζί με την παρούσα αναφορά υποβάλλουμε τα παρακάτω αρχεία κώδικα

Αρχείο	Αφορά το ερώτημα	Περιγραφή/Σχόλιο
<i>kafkaProducer.py</i>	Ερώτημα 1	Περιέχει τα ερωτήματα 1.1, 1.4
<i>kafkaConsumer.py</i>	Ερώτημα 1	Περιέχει το ερώτημα 1.5
<i>spark.py</i>	Ερώτημα 2	Περιέχει το ερώτημα 2.2
<i>mongodbQuery.py</i>	Ερώτημα 3	Περιέχει το ερώτημα 3.1, 3.2, 3.3 (το <i>python script</i> με τα <i>queries</i> στη <i>MongoDB</i>)

Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

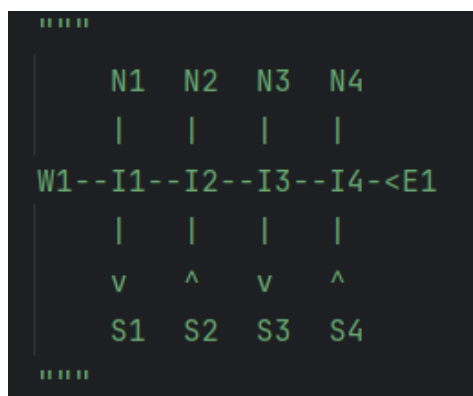
Χαρακτηριστικό	Τιμή
CPU model	AMD Ryzen 5 Mobile 4600H
CPU clock speed	3.0 GHz
Physical CPU cores	6
Logical CPU cores	12
RAM	16 GB
Secondary Storage Type	SSD

Ερώτημα 1: Παραγωγή δεδομένων

Για τη δημιουργία των δεδομένων χρησιμοποιούμε τον εξομοιωτή UXSIM και τον κώδικα που παρέχεται. Το simulation μας έχει tmax=600, δηλαδή διάρκεια 10 λεπτών και τα αποτελέσματα που παρέχονται τα τοποθετούμε σε Dataframe. Επίσης παρατηρούμε ότι κάθε ακμή έχει μέγεθος 500 μέτρα και όριο ταχύτητας 30χμλ και 50χμλ για τους παράδρομους και τον κεντρικό δρόμο αντίστοιχα. Έπειτα, δημιουργούμε έναν Kafka producer ο οποίος συνδέεται με τον Kafka cluster που έχουμε δημιουργήσει. Επιπλέον παίρνουμε δύο datetime, ένα για την αρχή της εξομοίωσης (sim_start) και ένα που δείχνει τότε ο producer ξεκίνησε την αποστολή δεδομένων (start_time).

Με την συνάρτηση “sim_time_send” στέλνουμε αρχικά τότε ξεκίνησε η παραγωγή των δεδομένων για να την χρησιμοποιήσουμε στην spark. Μετά με την συνάρτηση “data_to_kafka” στέλνουμε τα δεδομένα στο kafka broker. Σε αυτή την συνάρτηση ελέγχουμε την μέγιστη τιμή της στήλης time και όσο η μεταβλητή i (αρχικά i=0) δεν την έχει φτάσει, στέλνουμε τις στήλες όπου time=i. Ο simulator συλλέγει τα δεδομένα ανά 5 seconds ως αποτέλεσμα αυξάνουμε το i κατά 5. Επιπλέον σε αυτή την συνάρτηση αφαιρούμε την στήλη dh όπως φαίνεται στο παράδειγμα της εκφώνησης, μετατρέπουμε τα δεδομένα σε JSON και προσθέτουμε σε κάθε γραμμή το datetime έναρξης του producer με την μεταβλητή που περιέχεται στη στήλη time όπως ζητείται. Αγνοούμε τις γραμμές όπου το όχημα δεν έχει ξεκινήσει καθώς και αυτές που έχει τερματίσει την διαδρομή του (δηλαδή τις γραμμές όπου link= “Waiting_at_origin_node” ή “trip_end”). Η αποστολή των δεδομένων στον kafka broker γίνεται ανά 4 δευτερόλεπτα με κάθε θέση οχήματος να είναι ξεχωριστό JSON αντικείμενο.

Network



Παράμετροι

Εκτέλεση Kafka:

Ακολουθούμε τον οδηγό που μας παρέχεται «Οδηγίες εγκατάστασης Kafka» εκτελώντας τις εντολές που μας παρέχει. Πρώτα κατεβάζουμε τον kafka 3.7.0 με scala 2.13. Για να ξεκινήσουμε τον kafka server και να δημιουργήσουμε το topic vehicle_positions τρέχουμε τις εντολές:

- 1) bin/zookeeper-server-start.sh config/zookeeper.properties:

```

[2024-06-26 21:25:24,885] INFO bin/zookeeper-server-start.sh config/zookeeper.properties
[2024-06-26 21:25:24,885] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,886] WARN config/zookeeper.properties is relative. Prepend . to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,816] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,816] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,816] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,816] INFO metricsProvider.className is set to org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,814] INFO autopurge.snapRetainCount is set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-06-26 21:25:24,814] INFO autopurge.purgeInterval is set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-06-26 21:25:24,814] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-06-26 21:25:24,814] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2024-06-26 21:25:24,817] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-06-26 21:25:24,818] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,818] WARN config/zookeeper.properties is relative. Prepend . to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,818] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,819] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,819] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,819] INFO metricsProvider.className is set to org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-06-26 21:25:24,819] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-06-26 21:25:24,819] INFO Server Metrics Initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider#7921b0a2 (org.apache.zookeeper.server.ServerMetrics)
[2024-06-26 21:25:24,842] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-06-26 21:25:24,842] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-06-26 21:25:24,862] INFO zookeeper.snapshot.trust.empty = false (org.apache.zookeeper.server.persistence.FileZooKeeper)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,862] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-06-26 21:25:24,866] INFO Server environment: zookeeper.version=3.8.3-6add364c7c0bc4fe52d54ebfa38589fa65b, built on 2023-10-18 15:34 UTC (org.apache.zookeeper)

```

- 2) `bin/kafka-server-start.sh -daemon config/server.properties:`

```

[2024-06-20 21:27:43,299] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration)
[2024-06-20 21:27:43,705] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-06-20 21:27:43,818] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2024-06-20 21:27:43,820] INFO Starting (kafka.server.KafkaServer)
[2024-06-20 21:27:43,841] INFO Connecting to localhost:2181 (kafka.server.KafkaServer)
[2024-06-20 21:27:43,841] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2024-06-20 21:27:43,846] INFO Client environment: zookeeper.version=3.8.3-6ad6d36c7c8bCf8de452d54ebefca35899ba56, built on 2023-10-05 18:34 UTC (org.apache.zookeeper.ZooKeeper)
[2024-06-20 21:27:43,846] INFO Client environment:host.name=Dimitris-Laptop. (org.apache.zookeeper.ZooKeeper)
[2024-06-20 21:27:43,846] INFO Client environment:java.version=1.8.0_412 (org.apache.zookeeper.ZooKeeper)
[2024-06-20 21:27:43,847] INFO Client environment:java.vendor=Private Build (org.apache.zookeeper.ZooKeeper)
[2024-06-20 21:27:43,847] INFO Client environment:java.home=/usr/jdk/unsafejdk-8u412-linux/bin (org.apache.zookeeper.ZooKeeper)
[2024-06-20 21:27:43,847] INFO Client environment:java.class.path=/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/activation-1.1.1.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/aopalliance-repackaged-2.6.1.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/argparse4j-0.7.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/audience-annotations-0.12.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/caffeine-2.9.3.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/checker-qual-3.19.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-beanutils-1.9.4.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-cli-1.4.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-collections-3.2.2.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-digester-2.1.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-io-2.11.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-lang-3.8.1.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-logging-3.2.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/commons-validator-1.7.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/connect-api-3.0.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/connect-basic-auth-extension-3.7.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/connect-json-3.7.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/connect-mirror-3.7.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.:/lib:/connect-mirror-connector-3.7.0.jar:/home/dimitris/kafka_2.13-3.7.0/bin/.

```

- ```
3) bin/kafka-topics.sh --create --topic vehicle_positions --bootstrap-server
localhost:9092:
```

```

dimitris@Dimitris-Laptop: ~/ka/kafka-2.13-3.7.0$ bin/ka/kafka-topics.sh --create --topic vehicle_positions --bootstrap-server localhost:9092
WARNING: Due to limitations in metric names, topics with a period '.' or underscore '_' could collide. To avoid issues it is best to use either, but not both.
Created topic vehicle_positions.

```

Για την γρήγορη έναρξη του kafka broker έχουμε φτιάξει ένα αρχείο .sh που επιταχύνει αυτή την διαδικασία:

```
#!/bin/bash
echo "Starting Zookeeper..."
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
sleep 3
echo "Starting Kafka..."
bin/kafka-server-start.sh -daemon config/server.properties
sleep 3
echo "Create Topic vehicle_positions..."
bin/kafka-topics.sh --create --topic vehicle_positions --bootstrap-server localhost:9092
sleep 3
echo "List of Topics"
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
echo "Ready"
```

```
dimitris@Dimitris-Laptop:~/kafka_2.13-3.7.0$./startKafka.sh
Starting Zookeeper...
Starting Kafka...
Create Topic vehicle_positions...
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic vehicle_positions.
List of Topics
vehicle_positions
Ready
```

Από το παραπάνω script, φαίνεται πως δημιουργούμε το topic με όνομα vehicle\_positions.

### Λειτουργία μοντέλου consumer-producer

#### Producer

```
/home/dimitris/big_data/venv/bin/python /home/dimitris/big_data/kafkaProducer.py
simulation setting:
scenario name:
simulation duration: 600 s
number of vehicles: 15105 veh
total road length: 6500 m
time discret. width: 5 s
platoon size: 5 veh
number of timesteps: 120
number of platoons: 3021
number of links: 13
number of nodes: 14
setup time: 0.46 s
simulating...
time| # of vehicles| ave speed| computation time
0 s| 0 vehs| 0.0 m/s| 0.00 s
595 s| 580 vehs| 2.5 m/s| 0.52 s
simulation finished
```

#### Consumer

```
/home/dimitris/big_data/venv/bin/python /home/dimitris/big_data/kafkaConsumer.py
Received message: {'name': 'null', 'origin': 'null', 'destination': 'null', 'time': '20/06/2024 21:35:07', 'link': 'null', 'position': 'null', 'spacing': 'null', 'speed': 'null'}
Received message: {'name': '8', 'origin': 'E1', 'destination': 'W1', 'time': '20/06/2024 21:35:13', 'link': 'E1I4', 'position': 0.0, 'spacing': -1.0, 'speed': 50.0}
Received message: {'name': '15', 'origin': 'N1', 'destination': 'W1', 'time': '20/06/2024 21:35:13', 'link': 'N1I1', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '22', 'origin': 'S2', 'destination': 'W1', 'time': '20/06/2024 21:35:13', 'link': 'S2I2', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '29', 'origin': 'N3', 'destination': 'W1', 'time': '20/06/2024 21:35:13', 'link': 'N3I3', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '36', 'origin': 'S4', 'destination': 'W1', 'time': '20/06/2024 21:35:13', 'link': 'S4I4', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '8', 'origin': 'E1', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'E1I4', 'position': 250.0, 'spacing': -1.0, 'speed': 50.0}
Received message: {'name': '9', 'origin': 'E1', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'E1I4', 'position': 0.0, 'spacing': -1.0, 'speed': 50.0}
Received message: {'name': '15', 'origin': 'N1', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'N1I1', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '16', 'origin': 'N1', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'N1I1', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
Received message: {'name': '22', 'origin': 'S2', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'S2I2', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '23', 'origin': 'S2', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'S2I2', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
Received message: {'name': '29', 'origin': 'N3', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'N3I3', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '30', 'origin': 'N3', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'N3I3', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
Received message: {'name': '36', 'origin': 'S4', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'S4I4', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
Received message: {'name': '37', 'origin': 'S4', 'destination': 'W1', 'time': '20/06/2024 21:35:18', 'link': 'S4I4', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
```

## Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

Αρχικά, αρχικοποιούμε τις μεταβλητές `simulation_start`, `date_check`, `start_time_tmp` τις οποίες χρησιμοποιούμε για να βρούμε την αρχή της εξομοίωσης, τον έλεγχο αν βρήκαμε το `timestamp` που έχουμε θέσει ως αρχή της εξομοίωσης και τη λίστα που αποθηκεύει αυτή τη τιμή αντίστοιχα. Έπειτα, φτιάχνουμε το `schema`, το οποίο χρησιμοποιούμε για να ορίσουμε σαν αρχικό `Column` στο `Dataframe` που δεχόμαστε από τον `Kafka`.

Φτιάχνουμε το `sparkSession` (αν δεν υπάρχει ήδη, το δημιουργούμε) και συνδεόμαστε στο `stream` του `Kafka Server`, για να διαβάσουμε τα δεδομένα. Μετά, χειριζόμαστε τα δεδομένα σε `Batches` (`".forEachBatch(process_data_send_db)"`), χρησιμοποιούμε τα `triggers` για να σιγουρευτούμε ότι λαμβάνουμε σωστά τα δεδομένα και κάνουμε τις ανάλογες τροποποιήσεις μέσα στην `process_data_send_db`.

Μέσα στην συνάρτηση `process_data_send_db`, χρησιμοποιώντας το `Dataframe API`, κάνουμε `selectExpr` και `Cast` κάθε `value` σε `String` από το `JSON` που λαμβάνουμε, τροποποιούμε το `Column` του `time`, μετατρέποντάς το σε `timestamp` και όλα αυτά τα δεδομένα επιστρέφονται ως `Dataframe`. Για να μπορέσουμε να διαχειριστούμε τη πρώτη τιμή που λαμβάνουμε, το οποίο είναι η αρχή της εξομοίωσης, θέτουμε ένα καινούργιο `column`, το οποίο το έχουμε αποκλειστικά για να θέσουμε τη τιμή της αρχής της εξομοίωσης (αρχικά είναι `"00/00/00 00:00:00"`). Όταν λάβουμε από τον `Kafka` αυτή τη τιμή μέσω του `.where(" name=='null' ")` το οποίο επιστρέφει αποτέλεσμα μόνο αν η στήλη `name` έχει τη τιμή του `string 'null'`, (ισχύει μόνο στη συγκεκριμένη περίπτωση) και έπειτα με τη χρήση του `collect()` το συλλέγει και το αποθηκεύει σαν λίστα. Το `if` υπάρχει καθώς δε θέλουμε να κάνουμε αχρείαστα `queries` στο `Dataframe`.

Για να σιγουρευτούμε ότι υπάρχουν αντικείμενα στη λίστα αυτή, ελέγχουμε πρώτα αν είναι κενή και αν έχουμε λάβει επιτυχώς αυτή τη τιμή με τη `Boolean date_check`. Αποθηκεύουμε το `value` και δε το στέλνουμε. Έπειτα, αφότου λάβαμε επιτυχώς αυτή τη τιμή, στέλνουμε σαν `batches` τα ωμά δεδομένα στη συλλογή `"Cars_raw"` της βάσης `"Big_Data_Project"`. Τα ωμά δεδομένα περιέχουν το `timestamp`, χωρίς κάποια επιπλέον επεξεργασία. Όμως, στα επεξεργασμένα δεδομένα, αλλάζουμε τη στήλη `time` στην οποία αφαιρούμε το `timestamp` που περιέχει με αυτό της αρχής της εξομοίωσης και το αποτέλεσμα που βρίσκουμε είναι σε `seconds` το οποίο είναι η χρονική στιγμή από την αρχή της εξομοίωσης. Για να στείλουμε σωστά τα δεδομένα, στη μεταβλητή `processed_dataframe`, διαλέγουμε μόνο τα `columns` που μας ενδιαφέρουν και κάνουμε `groupby` με τα `link`, `time` και έπειτα βρίσκουμε το πλήθος των οχημάτων που περνάνε μέσα από ένα `link` καθώς και τη μέση ταχύτητα των οχημάτων αυτών. Τέλος, προσθέτουμε συνεχώς `batches` δεδομένων στη `MongoDB` τα δεδομένα της `processed_dataframe` στη συλλογή `"Cars_processed"` της βάσης `"Big_Data_Project"` στον `local MongoDB server` μας. Αν δεν υπάρχουν η συλλογή και η βάση, δημιουργούνται μόνες τους. Ακολουθούν `screenshot` εκτέλεσης και τα ερωτήματα επεξεργασίας των `Dataframe`.



## Screenshots εκτέλεσης spark:

```
dimitris@Dimitris-Laptop: ~/ivy2/art $ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.13:3.5.1,org.mongodb.spark:mongo-spark-connector_2.13:10.3.0 spark.py
:: loading settings :: url = jar:file:/etc/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/dimitris/.ivy2/cache
The jars for the packages stored in: /home/dimitris/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.13 added as a dependency
org.mongodb.spark#mongo-spark-connector_2.13 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f3bccc26-6aa4-4a04-b7a4-b3d57dceb4e3;1.0
 confs: [default]
 found org.apache.spark#spark-sql-kafka-0-10_2.13:3.5.1 in central
 found org.apache.spark#spark-token-provider-kafka-0-10_2.13:3.5.1 in central
 found org.apache.kafka#kafka-clients;3.4.1 in central
 found org.lz4#lz4-java;1.8.0 in central
 found org.xerial.snappy#snappy-java;1.1.10.3 in central
 found org.slf4j#slf4j-api;2.0.7 in central
 found org.apache.hadoop#hadoop-client-runtime;3.3.4 in central
 found org.apache.hadoop#hadoop-client-api;3.3.4 in central
 found commons-logging#commons-logging;1.1.3 in central
 found com.google.code.findbugs#jsr305;3.0.0 in central
 found org.scala-lang.modules#scala-parallel-collections_2.13;1.0.4 in central
 found org.apache.commons#commons-pool2;2.11.1 in central
 found org.mongodb.spark#mongo-spark-connector_2.13;10.3.0 in central
 found org.mongodb#mongodb-driver-sync;4.8.2 in central
 [4.8.2] org.mongodb#mongodb-driver-sync;[4.8.1,4.8.99)
 found org.mongodb#bson;4.8.2 in central
 found org.mongodb#mongodb-driver-core;4.8.2 in central
 found org.mongodb#bson-record-codec;4.8.2 in central
:: resolution report :: resolve 2144ms :: artifacts dl 18ms
 :: modules in use:
 com.google.code.findbugs#jsr305;3.0.0 from central in [default]
 commons-logging#commons-logging;1.1.3 from central in [default]
 org.apache.commons#commons-pool2;2.11.1 from central in [default]
 org.apache.hadoop#hadoop-client-api;3.3.4 from central in [default]
 org.apache.hadoop#hadoop-client-runtime;3.3.4 from central in [default]
 org.apache.kafka#kafka-clients;3.4.1 from central in [default]
 org.apache.spark#spark-sql-kafka-0-10_2.13;3.5.1 from central in [default]
 org.apache.spark#spark-token-provider-kafka-0-10_2.13;3.5.1 from central in [default]
 org.lz4#lz4-java;1.8.0 from central in [default]
 org.mongodb#bson;4.8.2 from central in [default]
 org.mongodb#bson-record-codec;4.8.2 from central in [default]
 org.mongodb#mongodb-driver-core;4.8.2 from central in [default]
 org.mongodb#mongodb-driver-sync;4.8.2 from central in [default]
 org.mongodb.spark#mongo-spark-connector_2.13;10.3.0 from central in [default]
 org.scala-lang.modules#scala-parallel-collections_2.13;1.0.4 from central in [default]
 org.slf4j#slf4j-api;2.0.7 from central in [default]
 org.xerial.snappy#snappy-java;1.1.10.3 from central in [default]
 |-----|
conf		modules		artifacts			
number	search	dwnlded	evicted		number	dwnlded	
-----	-----						
default	17	1	0	0		17	0
-----	-----						
:: retrieving :: org.apache.spark#spark-submit-parent-f3bccc26-6aa4-4a04-b7a4-b3d57dceb4e3
 confs: [default]
 0 artifacts copied, 17 already retrieved (0KB/12ms)

name	origin	destination	time	link	position	spacing	speed
4	E1	W1	2024-06-20 22:42:03	E1I4	0.0	-1.0	50.0
9	N1	W1	2024-06-20 22:42:03	N1I1	0.0	-1.0	30.0
14	S2	W1	2024-06-20 22:42:03	S2I2	0.0	-1.0	30.0
19	N3	W1	2024-06-20 22:42:03	N3I3	0.0	-1.0	30.0
24	S4	W1	2024-06-20 22:42:03	S4I4	0.0	-1.0	30.0

link	time	vcount	vspeed
N3I3	11	1	30.0
N1I1	11	1	30.0
S4I4	11	1	30.0
E1I4	11	1	50.0
S2I2	11	1	30.0

name	origin	destination	time	link	position	spacing	speed
4	E1	W1	2024-06-20 22:42:08	E1I4	250.0	-1.0	50.0
5	E1	W1	2024-06-20 22:42:08	E1I4	0.0	-1.0	50.0
9	N1	W1	2024-06-20 22:42:08	N1I1	150.0	-1.0	30.0
10	N1	W1	2024-06-20 22:42:08	N1I1	0.0	300.0	30.0
14	S2	W1	2024-06-20 22:42:08	S2I2	150.0	-1.0	30.0
15	S2	W1	2024-06-20 22:42:08	S2I2	0.0	300.0	30.0
19	N3	W1	2024-06-20 22:42:08	N3I3	150.0	-1.0	30.0
20	N3	W1	2024-06-20 22:42:08	N3I3	0.0	300.0	30.0
24	S4	W1	2024-06-20 22:42:08	S4I4	150.0	-1.0	30.0
25	S4	W1	2024-06-20 22:42:08	S4I4	0.0	300.0	30.0

link	time	vcount	vspeed
N1I1	16	2	30.0
E1I4	16	2	50.0
N3I3	16	2	30.0
S4I4	16	2	30.0
S2I2	16	2	30.0

name	origin	destination	time	link	position	spacing	speed
0	N1	S1	2024-06-20 22:42:13	N1I1	0.0	125.0	30.0
1	S2	N2	2024-06-20 22:42:13	S2I2	0.0	125.0	30.0
2	N3	S3	2024-06-20 22:42:13	N3I3	0.0	125.0	30.0
3	S4	N4	2024-06-20 22:42:13	S4I4	0.0	125.0	30.0
4	E1	W1	2024-06-20 22:42:13	E1I4	0.0	-1.0	50.0
5	E1	W1	2024-06-20 22:42:13	E1I4	250.0	-1.0	50.0
6	E1	W1	2024-06-20 22:42:13	E1I4	0.0	-1.0	50.0
9	N1	W1	2024-06-20 22:42:13	N1I1	300.0	-1.0	30.0
10	N1	W1	2024-06-20 22:42:13	N1I1	125.0	325.0	25.0
14	S2	W1	2024-06-20 22:42:13	S2I2	300.0	-1.0	30.0
15	S2	W1	2024-06-20 22:42:13	S2I2	125.0	325.0	25.0
19	N3	W1	2024-06-20 22:42:13	N3I3	300.0	-1.0	30.0
20	N3	W1	2024-06-20 22:42:13	N3I3	125.0	325.0	25.0
24	S4	W1	2024-06-20 22:42:13	S4I4	300.0	-1.0	30.0
25	S4	W1	2024-06-20 22:42:13	S4I4	125.0	325.0	25.0
```

## Ωμά δεδομένα σε Dataframe:

```
data_dataframe_raw = data.selectExpr("CAST(value AS STRING)") \
 .select(from_json(col("value"), schema).alias("data")) \
 .select("data.*") \
 .withColumn("time", to_timestamp("time", "dd/MM/yyyy HH:mm:ss")) # Transform column value of time to timestamp
```

## Χειρισμός επεξεργασμένων δεδομένων:

```
Change time column to t, where t the difference between simulation start and current timestamp of time
data_dataframe = data_dataframe.withColumn("time", (unix_timestamp(col("time")) -
 unix_timestamp(
 "simulation_start")))

Query only asked values (time,link,speed) and create columns with count of link for grouped by link,
time and avg of speed
processed_dataframe = data_dataframe.selectExpr("time", "link", "speed") \
 .groupby("link", "time") \
 .agg(count("link").alias("vcount"),
 avg("speed").alias(
 "vspeed"))
```

## Προσθήκη simulation\_start:

```
data_dataframe = data_dataframe_raw.withColumn("simulation_start",
 lit(simulation_start))
```

Το οποίο παίρνει την τιμή της αρχής της εξομοίωσης, αφότου τη λάβουμε, όπως αναφέρεται παραπάνω.

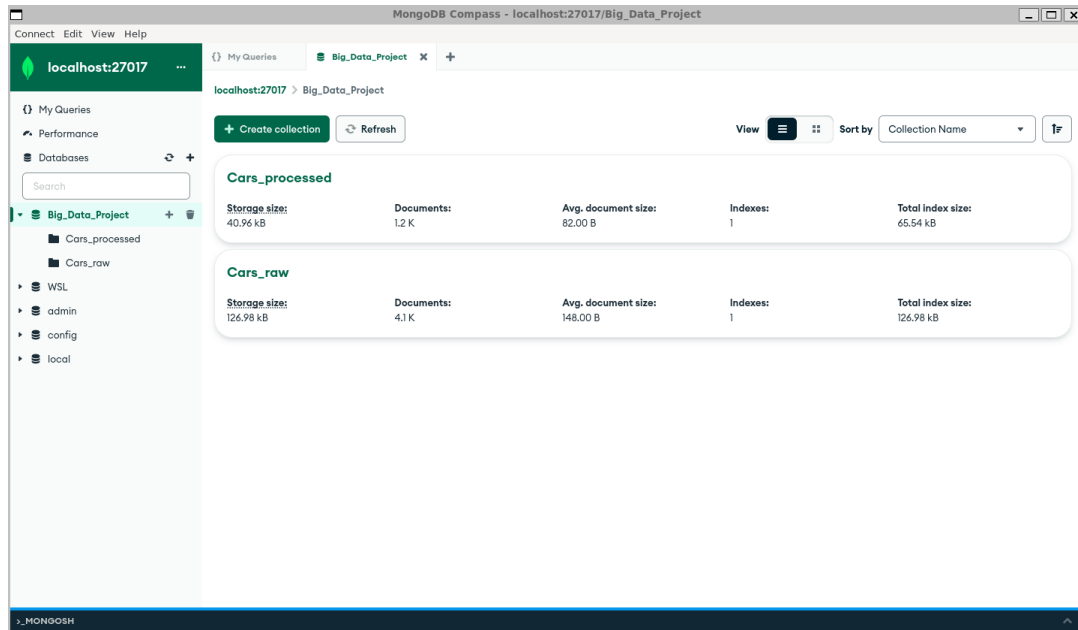
## Εγκατάσταση του Spark:

Αρχικά κατεβάσαμε την έκδοση 3.5.1 της spark (spark-3.5.1-bin-hadoop3-scala2.13.tgz) με Scala 2.13. Ακολουθώντας τον οδηγό που μας παρέχεται (Κατανάλωση δεδομένων από Kafka και επεξεργασία JSON σε Spark) εγκαθιστούμε την έκδοση του spark και τρέχουμε τον spark μαζί με τα packages της Kafka και του MongoDB-Spark connector με την εντολή:

```
dimitris@Dimitris-Laptop:~/big_data$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.13:3.5.1,org.mongodb.spark:mongo-spark-connector_2.13:10.3.0 spark.py
```

## Ερώτημα 3: Αποθήκευση σε MongoDB

Η βάση δεδομένων αποτελείται από το Database `Big_Data_Project` και δύο Collection, το `Cars_processed` που περιέχει τα επεξεργασμένα δεδομένα και το `Cars_raw` που περιέχει τα ωμά δεδομένα.

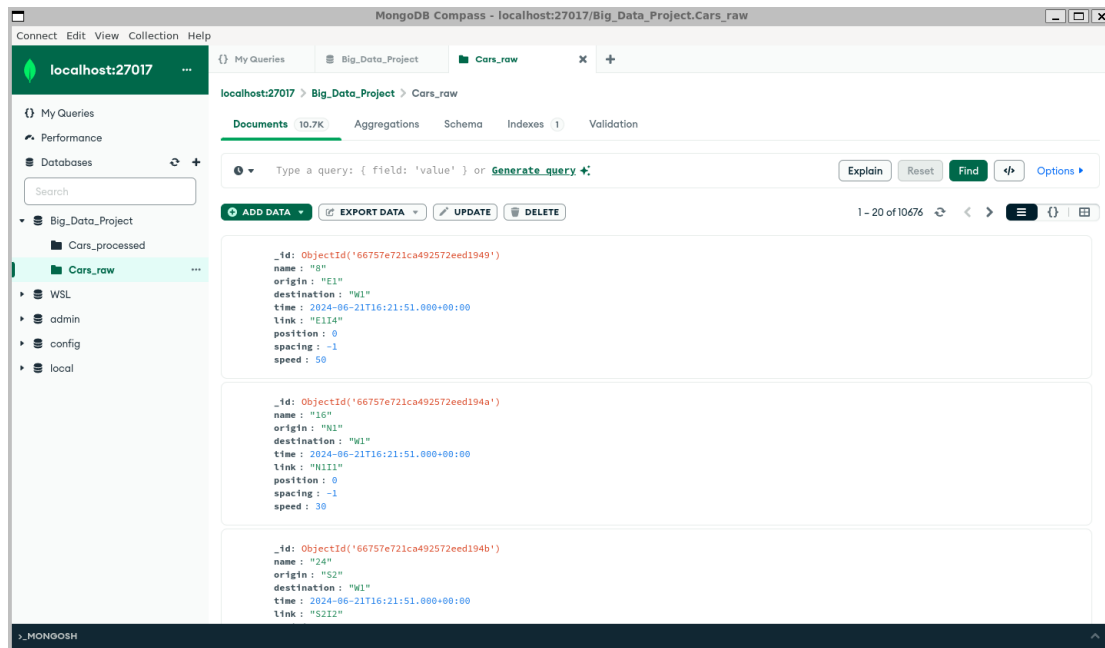


Την Database και τα Collection φτιάχνονται από την εκτέλεση του spark .Εάν δεν υπάρχουν, τα φτιάχνουμε μέσω του mongodb-compass(το GUI που παρουσιάζεται στο παραπάνω screenshot). Δημιουργία μέσω spark:

### Ωμά δεδομένα

```
Send raw_data to collection Cars_raw of Big_Data_Project at our local MongoDB database (create database and
collection, if they don't exist)
1 usage
def raw_data_db_send(raw_dataframe):
 raw_dataframe.write \
 .mode("append") \
 .format("mongodb") \
 .option("spark.mongodb.output.uri", "mongodb://localhost:27017/") \
 .option("spark.mongodb.database", "Big_Data_Project") \
 .option("spark.mongodb.collection", "Cars_raw") \
 .save()
```

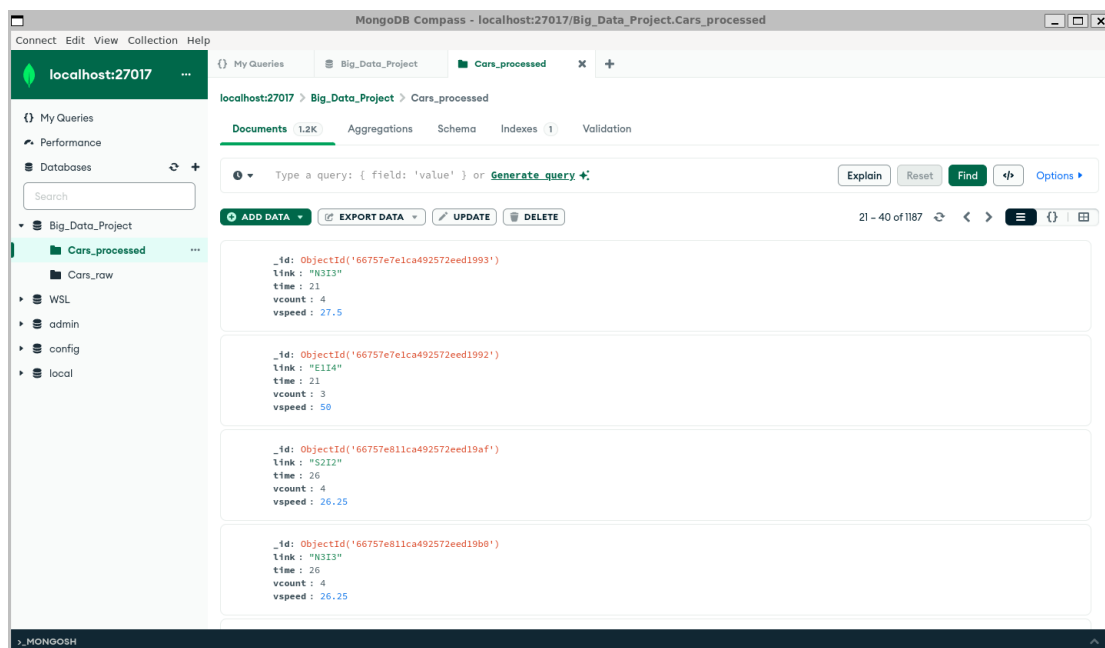




Εδώ αποθηκεύουμε τα ωμά δεδομένα στο Collection Cars\_raw της Mongoddb της βάσης δεδομένων Big\_Data\_Project σε append mode και format mongodb.

### Επεξεργασμένα δεδομένα

```
Send values to collection Cars_processed of Big_Data_Project at our local MongoDB database (create database
and collection, if they don't exist)
processed_dataframe.write \
 .format("mongodb") \
 .mode("append") \
 .option("spark.mongodb.output.uri", "mongodb://localhost:27017/") \
 .option("spark.mongodb.database", "Big_Data_Project") \
 .option("spark.mongodb.collection", "Cars_processed") \
 .save()
```



Αποθηκεύουμε τα επεξεργασμένα δεδομένα στο Collection Car\_processed της Mongoddb της βάσης δεδομένων Big\_Data\_Project σε append mode και format mongodb

Για την εκτέλεση των queries στην MongoDB χρησιμοποιούμε το script `mongodbQuery.py` χρησιμοποιώντας aggregation pipelines. Το πρώτο query βρίσκουμε την ακμή με το μικρότερο πλήθος οχημάτων μεταξύ μιας προκαθορισμένης χρονικής περιόδου μέσω της συνάρτησης `min_vcount` χρησιμοποιώντας το Collection `Cars_processed`. Αρχικά παίρνουμε τα documents που βρίσκονται στο χρονικό διάστημα που εισάγουμε (`match`), τα κάνουμε `groupby` με το `link` και παίρνουμε το άθροισμα του `vcount` για κάθε `link`. Έπειτα βρίσκουμε το μικρότερο άθροισμα `vcount` και κάνουμε `lookup` στο ίδιο Collection ακολουθώντας την ίδια διαδικασία για να βρούμε το άθροισμα `vcount` για κάθε `link`. Τέλος για κάθε `link` παίρνουμε τα αποτελέσματα που έχουν το ελάχιστο `vcount` που έχουμε βρει αρχικά.

```
The link with the smallest number of vehicles between 200 sec and 500 sec was I3S3 with 15 vehicle(s)
```

Για το δεύτερο query, για να βρούμε την ακμή με την μεγαλύτερη μέση ταχύτητα χρησιμοποιούμε το Collection `Cars_raw`. Σε αυτό πρώτα βρίσκουμε τα document που είναι μέσα στο χρονικό διάστημα που θέλουμε, τα κάνουμε `groupby` το `link` βρίσκοντας το μέσο όρο της ταχύτητας κάθε `link`. Έπειτα βρίσκουμε τον μεγαλύτερο μέσο όρο και κάνουμε `lookup` στο ίδιο Collection ώστε να βρούμε με τον ίδιο τρόπο το μέσο όρο κάθε ακμής και να πάρουμε αυτά που έχουν ίδια με το μέγιστο που έχουμε ήδη βρει.

```
The link with the highest average speed between 2024-06-21 16:22:00 sec and 2024-06-21 16:24:05 sec was I1W1 with avgSpeed 33.439153466905864
```

Για το τελευταίο query χρησιμοποιούμε το Collection `Cars_raw`. Μέσα από αυτό βρίσκουμε ποιο όχημα έχει διανύσει την μεγαλύτερη διαδρομή μέσα στο χρονικό διάστημα που δίνουμε. Αρχικά παίρνουμε τα documents που βρίσκονται μέσα στο χρονικό διάστημα που θέλουμε, τα κάνουμε `group by` το όνομα κάθε οχήματος τοποθετούμε σε ένα array τα μοναδικά `link` που έχει περάσει (`$addToSet`) και όλες τις τοποθεσίες του (`$push`) σε ένα δεύτερο array. Έπειτα βρίσκουμε το μέγεθος και των δύο arrays για κάθε `link`. Με την `$addFields` υπολογίζουμε την απόσταση που έχει διανύσει ελέγχοντας εάν έχει διανύσει 1 ή παραπάνω links. Εάν έχει διανύσει ένα τότε ελέγχουμε πόσες τοποθεσίες έχουμε για αυτό το όχημα, αν είναι μια έχει διανύσει 0 μέτρα ενώ εάν έχουμε παραπάνω αφαιρούμε από την τελική την αρχική απόσταση. Μετά εάν έχει περάσει παραπάνω links προσθέτουμε 500 μέτρα (κάθε ακμή έχει μέγεθος 500 μέτρα) για κάθε ένα από ενδιάμεσα links και για το `link` που ξεκίνησε αφαιρούμε 500 μείον την αρχική θέση του στο `link`. Για το τελευταίο `link` που βρίσκονταν απλά βρίσκουμε την τελική του θέση και τελικά προσθέτουμε και τα τρία αποτελέσματα βρίσκοντας την συνολική απόσταση που έχει διανύσει το όχημα. Τέλος βρίσκουμε την μεγαλύτερη απόσταση και παίρνουμε τα links που έχουν διανύσει απόσταση ίση με αυτή.

```
The car with the biggest route distance between 2024-06-21 16:22:00 and 2024-06-21 16:24:05 was 11 with 2433.3333435058594
```

## Αποτελέσματα

```
/home/dimitris/big_data/venv/bin/python /home/dimitris/big_data/mongodbQuery.py
The link with the smallest number of vehicles between 200 sec and 500 sec was I3S3 with 15 vehicle(s)
The link with the highest average speed between 2024-06-21 16:22:00 sec and 2024-06-21 16:24:05 sec was I1W1 with avgSpeed 33.439153466905864
The car with the biggest route distance between 2024-06-21 16:22:00 and 2024-06-21 16:24:05 was 11 with 2433.3333435058594
```

## Πρώτο Query

```
Query
pipeline = [
 {
 "$match": {
 "time": {"$gte": time_from, "$lte": time_to}
 }
 },
 {
 "$group": {
 "_id": "$link",
 "sumCount": {"$sum": "$vcount"}
 }
 },
 {
 "$group": {
 "_id": None,
 "minCount": {"$min": "$sumCount"}
 }
 },
 {
 "$lookup": {
 "from": "Cars_processed",
 "let": {"minCount": "$minCount"},
 "pipeline": [
 {
 "$match": {
 "time": {"$gte": time_from, "$lte": time_to}
 }
 },
 {
 "$group": {
 "_id": "$link",
 "sumCount": {"$sum": "$vcount"}
 }
 },
 {
 "$match": {
 '$expr': {"$eq": ["$sumCount", "$$minCount"]}
 }
 }
],
 "as": "result"
 }
 },
 {
 "$unwind": '$result'
 },
 {
 "$replaceRoot": {
 'newRoot': '$result'
 }
 }
]
```

## Δεύτερο Query

```
Query
pipeline = [
 {
 "$match": {
 "time": {"$gte": date_from, "$lte": date_to}
 }
 },
 {
 "$group": {
 "_id": "$link",
 "avgSpeed": {"$avg": "$speed"}
 }
 },
 {
 "$group": {
 "_id": None,
 "maxSpeed": {"$max": "$avgSpeed"}
 }
 },
 {
 "$lookup": {
 "from": "Cars_raw",
 "let": {"maxSpeed": "$maxSpeed"},
 "pipeline": [
 {
 "$match": {
 "time": {"$gte": date_from, "$lte": date_to}
 }
 },
 {
 "$group": {
 "_id": "$link",
 "avgSpeed": {"$avg": "$speed"}
 }
 },
 {
 "$match": {
 '$expr': {"$eq": ["$avgSpeed", "$$maxSpeed"]}
 }
 }
],
 "as": "result"
 }
 },
 {
 '$unwind': '$result'
 },
 {
 '$replaceRoot': {
 'newRoot': '$result'
 }
 }
]
```

## Trilo Query

```
Query
pipeline = [
 {
 "$match": {
 "time": {"$gte": date_from, "$lte": date_to}
 }
 },
 {
 "$group": {
 "_id": "$name",
 "link": {"$addToSet": "$link"},
 "position": {"$push": "$position"},
 }
 },
 {
 "$addFields": {
 "link_size": {"$size": "$link"},
 "pos_size": {"$size": "$position"}
 }
 },
 {
 "$addFields": {
 "route_distance": {
 "$cond": {
 "if": {"$eq": ["$link_size", 1]},
 "then": {
 "$cond": {
 "if": {"$eq": ["$pos_size", 1]},
 "then": 0,
 "else": {
 "$subtract": [{"$arrayElemAt": ["$position", -1]},
 {"$arrayElemAt": ["$position", 0]}]
 }
 }
 },
 "else": {
 "$add": [
 {"$subtract": [500, {"$arrayElemAt": ["$position", 0]}]},
 {"$multiply": [{"$subtract": ["$link_size", 2]}, 500]},
 {"$arrayElemAt": ["$position", -1]}
]
 }
 }
 }
 }
 },
 {
 "$project": {
 "_id": 0,
 "name": "$_id",
 "route_distance": 1
 }
 },
 {
 "$group": {
 "_id": None,
 "max_distance": {"$max": "$route_distance"},
 "route_car": {"$push": {"name": "$name", "route_distance": "$route_distance"}}
 }
 },
 {"$unwind": "$route_car"},
 {
 "$match": {
 "$expr": {"$eq": ["$max_distance", "$route_car.route_distance"]}
 }
 },
 {
 "$project": {
 "_id": 0,
 "name": "$route_car.name",
 "route_distance": "$route_car.route_distance"
 }
 }
]
```

## Εγκατάσταση MongoDB

Για να εγκαταστήσουμε σωστά τη MongoDB ακολουθήσαμε τις οδηγίες που παρέχονται από την ιστοσελίδα της MongoDB:

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/>

Εγκαθιστούμε την έκδοση 7.0.11 της MongoDB.

Η έκδοση Ubuntu (WSL2) που έχουμε είναι 22.04 και το init system μας είναι το systemd.

Το GUI που χρησιμοποιούμε είναι το mongodb-compass, το οποίο το εγκαθιστούμε με την βοήθεια των οδηγιών που παρέχονται από τον ιστότοπο:

<https://www.mongodb.com/docs/compass/current/install/>

Απαραίτητο να αναφερθεί, και πάλι, πως και στα packages του kafka προσθέτουμε τον mongo-spark-connector έκδοσης 10.3.0.

## Σχολιασμός αποτελεσμάτων

Από αυτό το project, κατανοήσαμε την σημασία ύπαρξης των Kafka servers, σε συνδυασμό με τη διαχείριση εισερχομένων δεδομένων μέσω της Spark και αποθήκευση των δεδομένων σε ένα NoSQL σύστημα όπως είναι η MongoDB.

Πιο συγκεκριμένα, τα δεδομένα που παράγονται από την εξομίωση της UXSIM μπορούμε με ευκολία να τα στείλουμε με τη χρήση του Kafka Server. Έπειτα, διαχειριζόμαστε τα δεδομένα που έχουμε στείλει, από τον Kafka, με τη χρήση της Spark. Η Spark είναι πολύ χρήσιμη στη περίπτωσή μας, καθώς είναι αναγκαίο να διαχειριστούμε μεγάλο όγκο δεδομένων με μεγάλη ταχύτητα. Τέλος, τα δεδομένα στέλνουμε σε μία βάση δεδομένων, τη MongoDB, η οποία αποθηκεύει σε batches τα δεδομένα τα οποία διαχειριστήκαμε όσο και τα ωμά.

Έτσι, δημιουργήσαμε ένα πλήρως λειτουργικό σύστημα διαχείρισης πραγματικών δεδομένων (DSMS, Data Streaming Management System) και κατανοήσαμε την αξία ύπαρξής του, σε περιβάλλοντα ανάκτησης μεγάλου πλήθους δεδομένων στο πραγματικό κόσμο καθώς και real-time δεδομένων.

## Βιβλιογραφία

- Example Spark 3.0.1 Data Transformations in Python, Sandeep Kattepogu, <https://sandeepkattepogu.medium.com/python-spark-transformations-on-kafka-data-8a19b498b32c>
- Single Node Spark/PySpark Cluster on Windows Subsystem for Linux (WSL2), Shafiqul Islam, <https://medium.com/@aitmsi/single-node-spark-pyspark-cluster-on-windows-subsystem-for-linux-wsl2-22860888a98d>
- Apache Kafka Quickstart, <https://kafka.apache.org/quickstart>



- MongoDB aggregation pipeline,  
<https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/>
- Spark Structured Streaming in general,  
<https://spark.apache.org/docs/3.1.1/api/python/reference/pyspark.ss.html>
- MongoDB-Spark connector, <https://www.mongodb.com/docs/spark-connector/current/>
- Spark-Kafka connection, <https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>
- MongoDB-compass installation,  
<https://www.mongodb.com/docs/compass/current/install/>
- MongoDB installation guide on Ubuntu,  
<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/>