

Πρότυπο αναφοράς άσκησης
Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου
Εργαστηριακή Άσκηση 2023/24

Όνομα	Επώνυμο	ΑΜ
Μαρία	Παπαδοπούλου	1072494

Email: up1072494@upnet.gr

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμψει 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 φαλκίδευσης της εξεταστικής και εκπαιδευτικής διαδικασίας από οιονδήποτε εξεταζόμενο, πέραν του μηδενισμού, συνιστά βαρύ πειθαρχικό παράπτωμα.

Υπογραφή

Υπογραφή

20 / 9 / 2024

___ / ___ / 2024

Συνημμένα αρχεία κώδικα

Μαζί με την παρούσα αναφορά υποβάλλουμε τα παρακάτω αρχεία κώδικα

Αρχείο	Αφορά το ερώτημα	Περιγραφή/Σχόλιο
<i>simulation.py</i>	1.1	Κώδικας της προσομοίωσης
<i>kafka_producer.py</i>	1.4	Κώδικας KafkaProducer
<i>ksm_pipeline_batched</i>	2, 3	Κώδικας Spark (επεξεργασία και αποθήκευση δεδομένων)
<i>queries.py</i>	3	Ερωτήματα στη βάση

Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

[Τεχνικά χαρακτηριστικά φυσικού Η/Υ που χρησιμοποιήθηκε για την εργασία, αν χρησιμοποιήθηκε hosted υπηρεσία μπορείτε απλά να αναφέρετε αυτό αντί για τον πίνακα]

Χαρακτηριστικό	Τιμή
CPU model	AMD Ryzen 5 5600G
CPU clock speed	3.9GHz
Physical CPU cores	4
Logical CPU cores	8
RAM	8
Secondary Storage Type	HDD/SSD

Για την εργασία χρησιμοποιήθηκε VM Linux Ubuntu Desktop 22.04.2

1. Ερώτημα 1: Παραγωγή δεδομένων

1.1 ΚΩΔΙΚΑΣ ΕΞΟΜΟΙΩΣΗΣ (simulation.py)

Ο κώδικας παραγωγής δεδομένων ξεκινά με τον ορισμό του seed για την παραγωγή τυχαιότητας και την αρχικοποίηση του κόσμου, όπως στο παράδειγμα που μας δόθηκε. Το οδικό δίκτυο που προσομοιώνουμε είναι στην ουσία μία πλατεία, με 4 εισόδους-εξόδους.

Η προσομοίωση ενός οδικού δικτύου με τον συγκεκριμένο εξομοιωτή μπορεί να χωριστεί σε 3 βήματα:

- 1) ΟΡΙΣΜΟΣ ΣΗΜΑΤΩΝ ΚΑΙ ΚΟΜΒΩΝ
- 2) ΟΡΙΣΜΟΣ ΑΚΜΩΝ
- 3) ΟΡΙΣΜΟΣ ΑΠΑΙΤΗΣΕΩΝ

Όπου, σήματα είναι τα φανάρια των διασταυρώσεων, κόμβοι τα σημεία έναρξης, προορισμού και οι διασταυρώσεις, ακμές οι δρόμοι και απαιτήσεις η κυκλοφορία στους δρόμους αυτούς.

Οι ορισμοί στον κώδικα μας έγιναν με την ίδια σειρά, με τα σήματα EWsignal, NSsignal να συμβολίζουν τα δευτερόλεπτα που το φανάρι είναι πράσινο την κατεύθυνση East-West και North-South αντίστοιχα. Οι κόμβοι τοποθετήθηκαν συμμετρικά και όλες οι διασταυρώσεις χρησιμοποιούν τα ίδια φανάρια. Οι δρόμοι παρουσιάζουν επίσης συμμετρία σε όλες τις ρυθμίσεις τους (μήκος, ταχύτητα κυκλοφορίας, πυκνότητα μποτιλιαρίσματος, πλήθος λωρίδων), εκτός από το signal group (0 = EWsignal και 1= NSsignal). Για την κυκλοφορία ορίστηκαν τυχαίες απαιτήσεις κάθε 30 δευτερόλεπτα, από όλες τις εισόδους προς όλες τις εξόδους. Οι βασικές ρυθμίσεις παρέμειναν ίδιες με αυτές του παραδείγματος.

Εκτελούμε την προσομοίωση, παίρνουμε τα πρώτα αποτελέσματα και το animation του οδικού δικτύου και αποθηκεύουμε τα δεδομένα που λάβαμε σε json αρχείο για να το χρησιμοποιήσουμε στα επόμενα ερωτήματα.

1.2 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΦΚΑ

Για την εγκατάσταση του Apache Kafka ακολουθήσαμε τα links και τις οδηγίες της εκφώνησης. Κατεβάσαμε την έκδοση 2.13-3.8.0 και σετάρουμε τον cluster με τις παρακάτω εντολές:

- 1) `cd ./kafka_2.13-3.8.0` (μεταφορά στο φάκελο εγκατάστασης του kafka)
- 2) `bin/zookeeper-server-start.sh config/zookeeper.properties` (εκκίνηση zookeeper)

```
marlagmaria@virtual-machine: ~/BigDataProject/kafka_2.13-3.8.0/
marlagmaria@virtual-machine:~/BigDataProject/kafka_2.13-3.8.0$ bin/zookeeper-server-start.sh config/zookeeper.properties
[2024-09-19 12:10:06,021] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,023] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,029] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,029] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,029] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,029] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,033] INFO autopurge.snapshotCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-09-19 12:10:06,033] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-09-19 12:10:06,033] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-09-19 12:10:06,033] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2024-09-19 12:10:06,037] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-09-19 12:10:06,037] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,038] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,038] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 12:10:06,038] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-09-19 12:10:06,038] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-09-19 12:10:06,038] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@174d2ba (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,043] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-09-19 12:10:06,043] INFO ZooKeeper.digestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-09-19 12:10:06,075] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-09-19 12:10:06,101] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,102] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,105] INFO Server environment:zookeeper.version=3.8.4-9316c2a7a97e166dd8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,105] INFO Server environment:host.name=maria-virtual-machine (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,105] INFO Server environment:java.version=1.8.0_422 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,106] INFO Server environment:java.vendor=Private Build (org.apache.zookeeper.server.ZooKeeperServer)
[2024-09-19 12:10:06,106] INFO Server environment:java.home=/usr/lib/jvm/java-8-openjdk-and64/ (org.apache.zookeeper.server.ZooKeeperServer)
```

Στη συνέχεια το terminal μας ενημερώνει για το server environment, τις διάφορες ρυθμίσεις του zookeeper και τα ports που χρησιμοποιεί.

- 3) `bin/kafka-server-start.sh config/server.properties` (εκκίνηση kafka)

```
marlagmaria@virtual-machine: ~/BigDataProject/kafka_2.13-3.8.0/
marlagmaria@virtual-machine:~/BigDataProject/kafka_2.13-3.8.0$ bin/kafka-server-start.sh config/server.properties
[2024-09-19 12:10:13,042] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2024-09-19 12:10:13,514] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-09-19 12:10:13,517] INFO RemoteLogManagerConfig values:
  log.local.retention.bytes = -2
  log.local.retention.ms = -2
  remote.fetch.max.wait.ms = 500
  remote.log.index.file.cache.total.size.bytes = 1073741824
  remote.log.manager.copier.thread.pool.size = 10
  remote.log.manager.copy.max.bytes.per.second = 9223372036854775807
  remote.log.manager.copy.quota.window.num = 11
  remote.log.manager.copy.quota.window.size.seconds = 1
  remote.log.manager.expiration.thread.pool.size = 10
  remote.log.manager.fetch.max.bytes.per.second = 9223372036854775807
  remote.log.manager.fetch.quota.window.num = 11
  remote.log.manager.fetch.quota.window.size.seconds = 1
  remote.log.manager.task.interval.ms = 30000
  remote.log.manager.task.retry.backoff.max.ms = 30000
  remote.log.manager.task.retry.backoff.ms = 500
  remote.log.manager.task.retry.jitter = 0.2
  remote.log.manager.thread.pool.size = 10
  remote.log.metadata.custom.metadata.max.bytes = 128
  remote.log.metadata.manager.class.name = org.apache.kafka.server.log.remote.metadata.storage.TopicBasedRemoteLogMetadataManager
  remote.log.metadata.manager.class.path = null
  remote.log.metadata.manager.impl.prefix = rlm.config
  remote.log.metadata.manager.listener.name = null
  remote.log.reader.max.pending.tasks = 100
  remote.log.reader.threads = 10
  remote.log.storage.manager.class.name = null
  remote.log.storage.manager.class.path = null
  remote.log.storage.manager.impl.prefix = rsn.config
  remote.log.storage.system.enable = false
(org.apache.kafka.server.log.remote.storage.RemoteLogManagerConfig)
[2024-09-19 12:10:13,668] INFO Registered signal handlers for [TERM, INT, HUP] (org.apache.kafka.common.utils.LoggingSignalHandler)
[2024-09-19 12:10:13,670] INFO Starting (kafka.server.KafkaServer)
[2024-09-19 12:10:13,670] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2024-09-19 12:10:13,687] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2024-09-19 12:10:13,692] INFO Client environment:zookeeper.version=3.8.4-9316c2a7a97e166dd8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.ZooKeeper)
[2024-09-19 12:10:13,693] INFO Client environment:host.name=maria-virtual-machine (org.apache.zookeeper.ZooKeeper)
[2024-09-19 12:10:13,693] INFO Client environment:java.version=1.8.0_422 (org.apache.zookeeper.ZooKeeper)
[2024-09-19 12:10:13,693] INFO Client environment:java.vendor=Private Build (org.apache.zookeeper.ZooKeeper)
[2024-09-19 12:10:13,693] INFO Client environment:java.home=/usr/lib/jvm/java-8-openjdk-and64/ (org.apache.zookeeper.ZooKeeper)
```

Στη συνέχεια το terminal μας ενημερώνει για το client environment, και διάφορες ρυθμίσεις του kafka και ξεκινάει τον server με τον kafka broker. Ενδιαφέρον παρουσιάζει η περίπτωση recovery από session expired, στην οποία εμφανίζονται και μηνύματα επαναφόρτωσης των αποθηκευμένων offsets (την κατάσταση αυτή πετύχαμε τρέχοντας kafka producer και consumer και βάζοντας το PC σε sleep mode για ώρες).

- 4) `bin/kafka-topics.sh --create --topic vehicles-positions --bootstrap-server localhost:9092 --partitions 4` (δημιουργία topic “vehicle-positions”, με 4 partitions)

```
marlagmaria@virtual-machine: ~/BigDataProject/kafka_2.13-3.8.0$ bin/kafka-topics.sh --create --topic vehicle-positions --bootstrap-server localhost:9092 --partitions 4
Created topic vehicle-positions.
```

- 5) Για την επαφή μας με το περιβάλλον kafka μέσω Python, εγκαταστάθηκε η βιβλιοθήκη KafkaPython με την εντολή `pip install kafka-python`, η οποία χρησιμοποιείται στον κώδικα.

1.3 ΚΩΔΙΚΑΣ ΠΑΡΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ ΚΑΦΚΑ (kafka_producer.py)

Αρχικά ορίζουμε τον παραγωγό και τις και τις ρυθμίσεις του, οι οποίες είναι:

- `bootstrap-servers`: ο server στον οποίο θα συνδεθεί ο παραγωγός είναι ο `localhost:9092`
- `linger_ms`: μέγιστο πλήθος `millisecond` που ο παραγωγός θα περιμένει, πριν στείλει ένα batch δεδομένων. Αν το batch γεμίσει νωρίτερα, τότε το στέλνει απευθείας. Η μεταβλητή αυτή ορίστηκε στα 10 δευτερόλεπτα.
- `batch_size`: πλήθος εγγραφών στο batch, ορίστηκε στις 5000 εγγραφές.
- `value_serializer`: encoding των εγγραφών προς αποστολή, επιλέχθηκε `json` μορφή με `utf-8 encoding`.

Αποθηκεύουμε τη στιγμή έναρξης της αποστολής και αρχικοποιούμε τον μετρητή απεσταλμένων εγγραφών. Ανοίγουμε το `json file` που προέκυψε από την εξομοίωση του προηγούμενου ερωτήματος και για κάθε εγγραφή μέσα σε αυτό εκτελούμε τις παρακάτω ενέργειες, προκειμένου να φτάσουμε στην επιθυμητή μορφή της εκφώνησης.

- Διαγραφή του πεδίου `'dn'`
- Φιλτράρισμα μόνο των οχημάτων που βρίσκονται πάνω σε ακμή
- Μετονομασία κλειδιών των αντίστοιχων εγγραφών
- Μετατροπή του πεδίου `t=δευτερόλεπτα` από την έναρξη της προσομοίωσης σε `T+t`, όπου `T` η χρονική στιγμή έναρξης της αποστολής δεδομένων από τον παραγωγό.
- Έλεγχος για γεμάτο batch. Στην περίπτωση που το Batch είναι γεμάτο, ο κώδικας θα σταματήσει να εκτελείται για 2 δευτερόλεπτα, επιτυγχάνοντας έτσι την περιοδικότητα που ζητείται από την εκφώνηση.
- Αποστολή της εγγραφής και αύξηση του μετρητή απεσταλμένων εγγραφών.

1.4 ΚΑΤΑΝΑΛΩΣΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΜΟΝΤΕΛΟΥ ΠΑΡΑΓΩΓΟΥ-ΚΑΤΑΝΑΛΩΤΗ

Η υλοποίηση Kafka-consumer από τον Kafka broker είναι απλή και γίνεται με την εντολή:

```
bin/kafka-console-consumer.sh --topic vehicle-positions --from-beginning --bootstrap-server localhost:9092
```

Ξεκινάμε πρώτα τον καταναλωτή και παρατηρούμε ότι στην αρχή, το `topic` είναι άδειο:

```
maria@maria-virtual-machine:~/BigDataProject/kafka_2.13-3.8.0$ bin/kafka-console-consumer.sh --topic vehicle-positions --from-beginning --bootstrap-server localhost:9092
```

Στη συνέχεια, εκκινούμε τον παραγωγός εκτελώντας το script *kafka_producer.py* και παρατηρούμε την κονσόλα του καταναλωτή:

```
maria@maria-virtual-machine:~$ cd $bigdataProject
maria@maria-virtual-machine:~/bigdataProject$ python3 final/kafka_producer.py
maria@maria-virtual-machine:~/bigdataProject$ python3 final/kafka_producer.py

{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:18", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:38", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:48", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:53", "link": "S1I4", "position": 150.0, "spacing": 25.0, "speed": 10.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:03", "link": "S1I4", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:48", "link": "S1I4", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:08", "link": "I4I2", "position": 50.0, "spacing": 1.0, "speed": 0.0}
{"name": "145", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:13", "link": "I2E1", "position": 100.0, "spacing": 1.0, "speed": 30.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:33", "link": "S1I4", "position": 75.0, "spacing": 25.0, "speed": 0.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:38", "link": "S1I4", "position": 75.0, "spacing": 25.0, "speed": 0.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:53", "link": "S1I4", "position": 75.0, "spacing": 25.0, "speed": 0.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:58", "link": "S1I4", "position": 125.0, "spacing": 25.0, "speed": 10.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:13", "link": "S1I4", "position": 125.0, "spacing": 25.0, "speed": 0.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:33", "link": "S1I4", "position": 125.0, "spacing": 25.0, "speed": 0.0}
{"name": "146", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:18", "link": "I4I2", "position": 200.0, "spacing": 1.0, "speed": 35.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:13", "link": "S1I4", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:58:23", "link": "S1I4", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:13", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:28", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 19:59:58", "link": "S1I4", "position": 100.0, "spacing": 25.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:03", "link": "S1I4", "position": 200.0, "spacing": 1.0, "speed": 20.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:08", "link": "S1I4", "position": 200.0, "spacing": 1.0, "speed": 0.0}
{"name": "147", "origin": "S1", "destination": "E1", "time": "2024-09-20 20:00:08", "link": "I4I2", "position": 75.0, "spacing": 25.0, "speed": 0.0}
```

Όπως φαίνεται, όσο ο παραγωγός ακόμα δουλεύει, ο καταναλωτής έχει εμφανίσει μερικά δεδομένα, αυτά που περιέχονται στα batches που έχει στείλει ο παραγωγός στον kafka broker. Μόλις ο παραγωγός ολοκληρώσει την αποστολή, ο καταναλωτής διαβάζει το τελευταίο batch και αναμένει για νέο μήνυμα από τον kafka broker.

```
maria@maria-virtual-machine:~$ cd $bigdataProject
maria@maria-virtual-machine:~/bigdataProject$ python3 final/kafka_producer.py
maria@maria-virtual-machine:~/bigdataProject$

{"name": "891", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:34", "link": "W1I3", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "891", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:44", "link": "W1I3", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "891", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:49", "link": "W1I3", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "891", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:14", "link": "I3I4", "position": 50.0, "spacing": 25.0, "speed": 5.0}
{"name": "891", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:29", "link": "I3I4", "position": 25.0, "spacing": 20.0, "speed": 10.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:44", "link": "W1I3", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:54", "link": "W1I3", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:59", "link": "W1I3", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:04", "link": "W1I3", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:19", "link": "W1I3", "position": 50.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:39", "link": "W1I3", "position": 150.0, "spacing": 25.0, "speed": 0.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:43:24", "link": "I3I4", "position": 25.0, "spacing": 50.0, "speed": 10.0}
{"name": "892", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:43:29", "link": "I3I4", "position": 50.0, "spacing": 25.0, "speed": 5.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:41:59", "link": "W1I3", "position": 25.0, "spacing": 25.0, "speed": 0.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:09", "link": "W1I3", "position": 25.0, "spacing": 25.0, "speed": 0.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:14", "link": "W1I3", "position": 25.0, "spacing": 25.0, "speed": 0.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:44", "link": "W1I3", "position": 125.0, "spacing": 25.0, "speed": 0.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:59", "link": "W1I3", "position": 125.0, "spacing": 25.0, "speed": 0.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:43:14", "link": "W1I3", "position": 200.0, "spacing": 1.0, "speed": 35.0}
{"name": "893", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:43:59", "link": "I3I4", "position": 200.0, "spacing": 1.0, "speed": 0.0}
{"name": "894", "origin": "W1", "destination": "S1", "time": "2024-09-20 20:42:54", "link": "W1I3", "position": 0.0, "spacing": 50.0, "speed": 50.0}
```


2. Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

2.1 ΕΓΚΑΤΑΣΤΑΣΗ SPARK

Εγκαταστήσαμε την έκδοση 3.5.2 του Apache Spark με υποστήριξη Hadoop3 από την [επίσημη ιστοσελίδα](#) του εργαλείου. Στη συνέχεια, εκτελέσαμε τις παρακάτω εντολές για να το ενεργοποιήσουμε:

- `cd ./spark-3.5.2-bin-hadoop3` (μεταφορά στο φάκελο εγκατάστασης του spark)
- `sbin/start-master.sh` (εκκίνηση master node)
- `sbin/start-worker.sh spark://localhost:7077` (εκκίνηση worker nodes)

Εδώ πρέπει να αναφέρουμε ότι αλλάξαμε τις ρυθμίσεις του περιβάλλοντος spark από το προεπιλεγμένο. Συγκεκριμένα, ψάχνοντας στο φάκελο `./conf` εντός του φακέλου εγκατάστασης του spark, παρατηρήσαμε ότι υπάρχει μόνο το αρχείο `spark-env.sh.template`, ένα αρχείο γεμάτο σχόλια. Το αντιγράψαμε, σβήσαμε την κατάληξη `.template` και προσθέσαμε μετά τα σχόλια τις εξής γραμμές:

`SPARK_MASTER_HOST='localhost'` : ο host του master node (by default ήταν ο `local[*]`, οπότε τώρα αλλάξαμε το default και δεν χρειάζεται να το γράφουμε κάθε φορά)

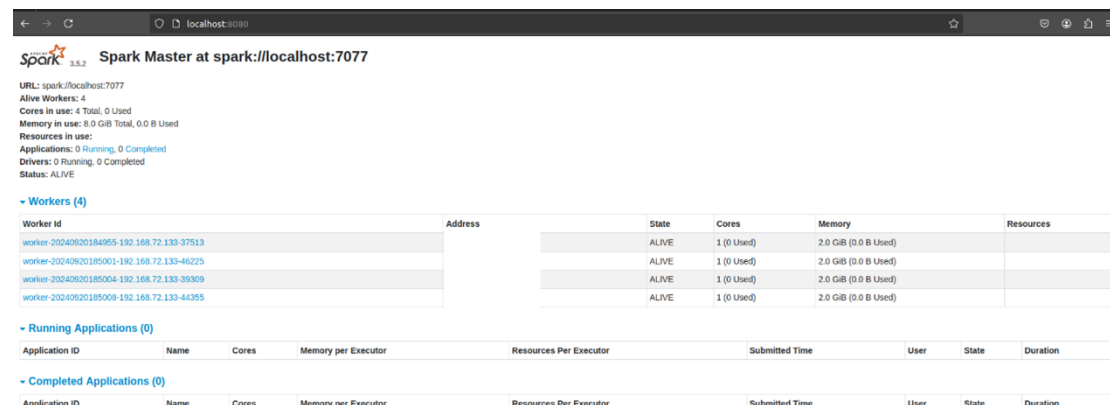
`SPARK_WORKER_CORES=1` : πλήθος cores ανά worker

`SPARK_WORKER_INSTANCES=4` : πλήθος workers

`SPARK_WORKER_MEMORY=2g` : πλήθος μνήμης ανά worker

Οι ρυθμίσεις αυτές ορίστηκαν ώστε να αξιοποιούμε στο έπακρο τους πόρους της εικονικής μας μηχανής και αν ξεπεραστούν με τον οποιοδήποτε τρόπο η spark διαδικασία δεν θα τρέξει, αλλά θα λάβουμε μήνυμα λάθους ότι οι πόροι που απαιτεί ξεπερνούν τους διαθέσιμους.

Επιβεβαιώνουμε ότι ο master και οι workers δουλεύουν, αν πληκτρολογήσουμε στη γραμμή διευθύνσεων του browser τη διεύθυνση `localhost:8088` (βλ. επόμενη εικόνα)



Spark Master at spark://localhost:7077

URL: spark://localhost:7077
Alive Workers: 4
Cores in use: 4 Total, 0 Used
Memory in use: 8.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (4)

Worker ID	Address	State	Cores	Memory	Resources
worker-20240920184955-192.168.72.133-37513	192.168.72.133:37513	ALIVE	1 (0 Used)	2.0 GiB (0.0 B Used)	
worker-20240920185001-192.168.72.133-40225	192.168.72.133:40225	ALIVE	1 (0 Used)	2.0 GiB (0.0 B Used)	
worker-20240920185004-192.168.72.133-39309	192.168.72.133:39309	ALIVE	1 (0 Used)	2.0 GiB (0.0 B Used)	
worker-20240920185009-192.168.72.133-44355	192.168.72.133:44355	ALIVE	1 (0 Used)	2.0 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 (0)

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

Spark Web UI: λειτουργεί και μας δείχνει 4 workers, με τις ρυθμίσεις που έχουμε κάνει. Οι διευθύνσεις έχουν αποκρυφθεί.

2.2 CONNECTORS – RUN SPARK APPLICATION

Δεδομένου ότι η εφαρμογή μας απαιτεί kafka-spark-mongoDB pipeline σε python, το spark χρειάζεται 2 connectors για να συνδεθεί με τα άλλα 2 εργαλεία και 1 βιβλιοθήκη για να μπορούμε να το προγραμματίσουμε μέσω python. Εμείς χρησιμοποιήσαμε:

+ spark-MongoDB connector

org.mongodb.spark:mongo-spark-connector_2.12:10.4.0

+ spark-kafka connector

org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.2

+ Python Library for Spark: pyspark (με pip install pyspark)

Για να τρέξει η εφαρμογή μας με τους connectors, πρέπει να κάνουμε 2 πράγματα:

- 1) Να ορίσουμε τις εξαρτήσεις εντός της εφαρμογής, στο SparkSession configuration (βλ. επόμενη παράγραφο)
- 2) Να προσθέσουμε τις εξαρτήσεις στην εντολή εκτέλεσης του κώδικα ως εξής:
spark-submit --packages org.mongodb.spark:mongo-spark-connector_2.12:10.4.0, org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.2 ../example_write_mongo.py

2.3 ΚΩΔΙΚΑΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΔΕΔΟΜΕΝΩΝ ΚΑΦΚΑ ΚΑΙ ΑΠΟΘΗΚΕΥΣΗΣ ΣΤΗ MONGODB (ksm_pipeline_batched.py)

Αρχικά ορίζεται το spark Session και οι ρυθμίσεις του:

- appName: όνομα της spark διαδικασίας
- config: στο πεδίο αυτό μπορούμε να ορίσουμε πολλές ρυθμίσεις, εμείς βάλαμε μόνο τις εξαρτήσεις που αναφέραμε.

Ολόκληρή η διαδικασία αποτελείται από 3 τμήματα:

- 1) Ανάγνωση stream από Kafka:
 - a) Το session συνδέεται στον kafka server localhost:9092 και με earliest offset, επιχειρεί ανάγνωση streaming data από το topic vehicle-positions, στο Spark Dataframe json_stream.
 - b) Ορίζεται το σχήμα των δεδομένων εισόδου, που δεν είναι άλλο από αυτό που έστειλε ο παραγωγός της προηγούμενης ενότητας στον kafka server και
 - c) Απομονώνονται τα επιθυμητά πεδία του kafka stream, σύμφωνα με αυτό το σχήμα: Το kafka stream περιέχει πεδία όπως key, value (binaries), topic (string), partition (int), offset (long), timestamp, timestampType (int) και πιθανόν headers array. Από αυτά, τα δεδομένα μας βρίσκονται στο πεδίο value, το οποίο μετατρέπεται σε αλφαριθμητικό για να μπορούμε να το διαβάσουμε και απομονώνεται ως json με σχήμα, αυτό που ορίσαμε στο βήμα 1c. Τέλος, μετονομάζεται σε data και επιλέγουμε όλα τα πεδία του (ζεύγη κλειδιών-τιμών)
 - d) Μετατροπή του πεδίου “time” σε timestamp, για να αποθηκευτεί με αυτή τη μορφή στη βάση δεδομένων (έτσι διαχειρίζεται η MongoDB τις χρονικές τιμές).

2) Επεξεργασία των δεδομένων εισόδου:

- a) Προετοιμασία των δεδομένων προς επεξεργασία, με απομόνωση των χρήσιμων για την επεξεργασία πεδίων σε νέο spark dataframe.
- b) Προσθήκη watermark: για να εξάγουμε τα δεδομένα εξόδου με τα πεδία όπως, “αριθμός αυτοκινήτων” και “μέση ταχύτητα” των ακμών κάθε χρονική στιγμή, πρέπει να κάνουμε ορισμένα aggregations, ομαδοποιώντας με βάση το πεδίο του χρόνου (και της ακμής, αλλά αυτή δε σχετίζεται με το watermark). Το Spark πρέπει να γνωρίζει το χρονικό παράθυρο μέσα στο οποίο μπορεί να δέχεται δεδομένα για τα aggregations καθώς πίσω από κάποιο όριο τα δεδομένα θεωρούνται παρωχημένα και δεν συμπεριλαμβάνονται στις πράξεις. Εμείς ορίζουμε τη διάρκεια του χρονικού παραθύρου και το όριο προκύπτει ως max event time – διάρκεια παραθύρου. Ως events εννοούνται οι εγγραφές μας, άρα event time είναι το πεδίο time σε κάποια εγγραφή. Έτσι, για διάρκεια παραθύρου 1 ώρας, όσο και η διάρκεια της εξομοίωσης που παρήγαγε τα δεδομένα, αν η μέγιστη τιμή του πεδίου time των δεδομένων είναι 12:00, τότε το Spark θα συμπεριλάβει στο aggregation δεδομένα με πεδίο time μέχρι και 11:00 αλλά όχι παλιότερο (δηλαδή 10:59 και πίσω).
- c) Πράξεις στα watermarked δεδομένα: η λογική πίσω από τα πεδία vcount = αριθμός αυτοκινήτων μέσα στην ακμή και vspeed = μέση ταχύτητα οχημάτων πάνω στην ακμή είναι η εξής: ομαδοποίησε τα δεδομένα ως προς το χρόνο και τις ακμές και:
 - i) vcount: μέτρα πόσες εγγραφές υπάρχουν σε κάθε group (time, link)
 - ii) vspeed: Υπολόγισε τον μέσο όρο του “speed” για τα groups με (time, link)

Έτσι, έχουμε τις ζητούμενες ποσότητες για κάθε ακμή, κάθε χρονική στιγμή

3) Αποθήκευση δεδομένων στη MongoDB: αναλύεται στο κεφάλαιο 3.4.

Την επιτυχία εκτέλεσης καταλαβαίνουμε και από τον έλεγχο των collections που δημιουργήσαμε στη MongoDB, αλλά και από το terminal. Θα εκτελέσουμε 2 queries στη βάση για την αποθήκευση των επεξεργασμένων και των ωμών δεδομένων, για καθένα από τα οποία λαμβάνουμε στο terminal τις στήλες των επόμενων εικόνων.

```
24/09/20 20:08:07 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "69cee38a-053d-47a7-a55e-366ac512e9e4",
  "runId" : "5404fb84-c337-4b06-b192-833e44d5d39a",
  "name" : null,
  "timestamp" : "2024-09-20T17:08:02.067Z",
  "batchId" : 1,
  "numInputRows" : 0,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 0.0,
  "durationMs" : {
    "addBatch" : 5200,
    "commitOffsets" : 31,
    "getBatch" : 0,
    "latestOffset" : 7,
    "queryPlanning" : 22,
    "triggerExecution" : 5300,
    "walCommit" : 36
  },
  "eventTime" : {
    "watermark" : "2024-09-20T16:49:43.000Z"
  },
  "stateOperators" : [ {
    "operatorName" : "stateStoreSave",
    "numRowsTotal" : 6458
  } ],
  "watermark" : "1970-01-01T00:00:00.000Z"
},
24/09/20 20:08:02 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "69cee38a-053d-47a7-a55e-366ac512e9e4",
  "runId" : "5404fb84-c337-4b06-b192-833e44d5d39a",
  "name" : null,
  "timestamp" : "2024-09-20T17:07:47.249Z",
  "batchId" : 0,
  "numInputRows" : 27281,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 1841.6931074056572,
  "durationMs" : {
    "addBatch" : 13074,
    "commitOffsets" : 38,
    "getBatch" : 23,
    "latestOffset" : 958,
    "queryPlanning" : 633,
    "triggerExecution" : 14813,
    "walCommit" : 54
  },
  "eventTime" : {
    "avg" : "2024-09-20T17:20:05.020Z",
    "max" : "2024-09-20T17:49:43.000Z",
    "min" : "2024-09-20T16:49:53.000Z",
    "watermark" : "1970-01-01T00:00:00.000Z"
  },
  "stateOperators" : [ {
    "operatorName" : "stateStoreSave",
    "numRowsTotal" : 6458
  } ],
  "watermark" : "1970-01-01T00:00:00.000Z"
}
```

Αυτό που ενδιαφέρει περισσότερο από αυτές είναι τα επόμενα σημεία που περιγράφουν τα queries μας:

```

    },
    "sources" : [ {
      "description" : "KafkaV2[Subscribe[vehicle-positions]]",
      "startOffset" : null,
      "endOffset" : {
        "vehicle-positions" : {
          "2" : 6832,
          "1" : 6829,
          "3" : 6812,
          "0" : 6808
        }
      },
      "latestOffset" : {
        "vehicle-positions" : {
          "2" : 6832,
          "1" : 6829,
          "3" : 6812,
          "0" : 6808
        }
      }
    },
    "numInputRows" : 27281,
    "inputRowsPerSecond" : 0.0,
    "processedRowsPerSecond" : 1841.6931074056572,
    "metrics" : {
      "avgOffsetsBehindLatest" : "0.0",
      "maxOffsetsBehindLatest" : "0",
      "minOffsetsBehindLatest" : "0"
    }
  } ],
  "sink" : {
    "description" : "ForeachBatchSink",
    "numOutputRows" : -1
  }
}

```

3. Ερώτημα 3: Αποθήκευση σε MongoDB

3.1 ΕΓΚΑΤΑΣΤΑΣΗ MONGODB

Εγκαταστήσαμε MongoDB 7.0 Community Edition για τα Ubuntu, από την επίσημη ιστοσελίδα του εργαλείου. Στη συνέχεια, την ενεργοποιήσαμε ακολουθώντας τις παρακάτω εντολές:

- `sudo systemctl start mongod`
- `mongosh`

```
maria-virtual-machine:~/bigdataproject/kafka_2.13-3.0.0$ sudo systemctl start mongod
[sudo] password for maria:
maria-virtual-machine:~/bigdataproject/kafka_2.13-3.0.0$ mongosh
Current Mongosh Log ID: 66eda966d3d38f83c964032
Connecting to:
  mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.1
Using MongoDB:
  7.0.14
Using Mongosh:
  2.3.1

For mongosh info see: https://www.mongodb.com/docs/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-09-20T19:57:07.402+03:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-09-20T19:57:07.825+03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-09-20T19:57:07.827+03:00: vm.max_map_count is too low
-----

test> show dbs
admin      48.00 KiB
config     84.00 KiB
local      72.00 KiB
project    1.21 MiB
test       72.00 KiB
test10     1.53 MiB
test2      48.00 KiB
test3      124.00 KiB
test4      156.00 KiB
test5      128.00 KiB
test> use project
switched to db project
project>
```

3.2 ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΗΣ

Το σχήμα της βάση διατηρήθηκε απλό. Υπάρχουν μόνο 2 collections, μία για τα ωμά δεδομένα και μία για τα επεξεργασμένα. Κάθε document μέσα στις collections έχει τα πεδία του, όπως διαμορφώθηκαν από την αρχή της διαδικασίας μέχρι αυτό το σημείο και ένα πεδίο πρωτεύοντος κλειδιού `_id`.

Το εν λόγω σχήμα εξυπηρετούσε τα ερωτήματα που σκοπεύουμε να κάνουμε στη βάση, συνεπώς δεν υπήρχε ανάγκη για κάτι πιο πολύπλοκο. Δημιουργήσαμε όμως indexes για τα πεδία εκείνα που θα χρησιμοποιήσουμε για την ομαδοποίηση των δεδομένων στα ερωτήματα που θα κάνουμε στη βάση παρακάτω. Συγκεκριμένα, φτιάξαμε ευρετήρια για τα πεδία `time`, `link`, `origin` και `destination`, όλα με την προεπιλεγμένη αύξουσα σειρά.

3.3 ΔΗΜΙΟΥΡΓΙΑ ΣΥΛΛΟΓΩΝ (ΑΠΟ ΤΟ `ksm_pipeline.py`)

Οι συλλογές της βάσης δημιουργήθηκαν αυτόματα κατά την εγγραφή των δεδομένων από το spark στη MongoDB. Τα statements που οδήγησαν στην δημιουργία τους είναι συνεπώς τα `query1` και `query2`, που αναλύονται στο επόμενο κεφάλαιο:

```

61 # STORE DATA IN MONGODB: AGGREGATED DATA
62 def write_to_mongodb(batch_df, batch_id):
63     batch_df.write \
64         .format("mongodb") \
65         .option("spark.mongodb.connection.uri", "mongodb://localhost:27017/project.spark_aggregated") \
66         .option("spark.mongodb.database", "project") \
67         .option("spark.mongodb.collection", "spark_aggregated") \
68         .mode("append") \
69         .save()
70
71 query1 = (
72     aggregated_df.writeStream
73     .foreachBatch(write_to_mongodb) \
74     .outputMode("update") \
75     .trigger(processingTime="10 seconds") \
76     .start()
77 )
78
79 # STORE DATA IN MONGODB: RAW DATA
80 def write_raw_to_mongodb(batch_df, batch_id):
81     batch_df.write \
82         .format("mongodb") \
83         .option("spark.mongodb.connection.uri", "mongodb://localhost:27017/project.spark_raw") \
84         .option("spark.mongodb.database", "project") \
85         .option("spark.mongodb.collection", "spark_raw") \
86         .mode("append") \
87         .save()
88
89 query2 = (
90     json_stream_tmstamp.writeStream
91     .foreachBatch(write_raw_to_mongodb) \
92     .outputMode("append") \
93     .trigger(processingTime="10 seconds") \
94     .start()
95 )
96
97 query1.awaitTermination()
98 query2.awaitTermination()
99

```

3.4 ΑΠΟΘΗΚΕΥΣΗ ΔΕΔΟΜΕΝΩΝ (ΩΜΑ ΚΑΙ ΕΠΕΞΕΡΓΑΣΜΕΝΑ - ΑΠΟ ΤΟ ksm_pipeline.py)

Τόσο τα επεξεργασμένα δεδομένα, όσο και τα ωμά, αποθηκεύονται σε ξεχωριστά collections της ίδιας βάσης. Τα queries αποθήκευσης είναι παρόμοια και είναι ουσιαστικά καλούν τις συναρτήσεις εγγραφής σε MongoDB για κάθε batch.

Παρατηρούμε ότι, ενώ τα queries χρησιμοποιούν τη συνάρτηση writeStream(), επειδή ζητούν εγγραφή streaming data, οι συναρτήσεις χρησιμοποιούν τη write() επειδή τα batches έχουν συγκεκριμένο πλήθος εγγραφών πριν αποσταλούν. Οι ρυθμίσεις που χρησιμοποιούνται είναι:

α) Εντός των συναρτήσεων:

- spark.mongodb.connection.uri: σύνδεσμος για τον εντοπισμό της βάσης και της collection στην οποία θα αποθηκευτούν τα δεδομένα, η δική μας είναι πάνω στον localhost:27017
- spark.mongodb.database: Η βάση στην οποία θα εγγραφούν τα δεδομένα ("project").
- spark.mongodb.collection: Η συλλογή στην οποία θα εγγραφούν τα δεδομένα (spark_aggregated και spark_raw για τα επεξεργασμένα και τα ωμά δεδομένα αντίστοιχα).
- mode: καθορίζει το πώς θα εγγραφούν τα δεδομένα. Στην περίπτωσή μας ("append"), μόνο τα νέα documents από κάθε batch.

b) Για τα queries εγγραφής στη βάση:

- trigger (processingTime): κάθε πότε υπολογίζονται τα aggregations με τα νέα δεδομένα που έχουν ληφθεί. Ορίζεται στα 10 δευτερόλεπτα.
- outputMode: ίδιο με το mode, με την πρόσθετη διαγραφή των ενδιάμεσων aggregation states.

Επιβεβαιώνουμε ότι τα δεδομένα έχουν καταχωρηθεί στη βάση από το mongoDB shell:

```
test10> use project
switched to db project
project> db.spark_aggregated.find({})
[
  {
    _id: ObjectId('66ec46858e43bd2c2cd39203'),
    time: ISODate('2024-09-19T15:40:08.000Z'),
    link: 'I2I1',
    vcount: Long('1'),
    vspeed: 0
  },
  {
    _id: ObjectId('66ec46858e43bd2c2cd39204'),
    time: ISODate('2024-09-19T16:15:23.000Z'),
    link: 'I3I4',
    vcount: Long('5'),
    vspeed: 0
  },
  {
    _id: ObjectId('66ec46858e43bd2c2cd39205'),
    time: ISODate('2024-09-19T15:56:28.000Z'),
    link: 'N1I1',
    vcount: Long('5'),
    vspeed: 3
  },
  {
    _id: ObjectId('66ec46858e43bd2c2cd39206'),
    time: ISODate('2024-09-19T16:22:18.000Z'),
    link: 'N1I1',
    vcount: Long('5'),
    vspeed: 3
  }
]
```

Τα επεξεργασμένα δεδομένα από το spark στη συλλογή project.spark_aggregated της mongoDB

```
project> db.spark_raw.find({})
[
  {
    _id: ObjectId('66ec467c8e43bd2c2cd3277c'),
    name: '0',
    origin: 'N1',
    destination: 'S1',
    time: ISODate('2024-09-19T15:35:53.000Z'),
    link: 'N1I1',
    position: 0,
    spacing: -1,
    speed: 50
  },
  {
    _id: ObjectId('66ec467c8e43bd2c2cd32785'),
    name: '0',
    origin: 'N1',
    destination: 'S1',
    time: ISODate('2024-09-19T15:36:13.000Z'),
    link: 'N1I1',
    position: 200,
    spacing: -1,
    speed: 0
  },
  {
    _id: ObjectId('66ec467c8e43bd2c2cd32788'),
    name: '0',
    origin: 'N1',
    destination: 'S1',
    time: ISODate('2024-09-19T15:36:13.000Z'),
    link: 'N1I1',
    position: 200,
    spacing: -1,
    speed: 0
  }
]
```

Τα ωμά δεδομένα από το spark, στη συλλογή project.spark_raw της mongoDB

3.5 ΚΩΔΙΚΑΣ QUERIES (queries.py)

Για τον κώδικα των ερωτημάτων χρησιμοποιήσαμε τη βιβλιοθήκη pymongo, την οποία εγκαταστήσαμε με `pip install pymongo`.

Στο script, αρχικά συνδεόμαστε στη βάση, παίρνουμε τις επιθυμητές συλλογές και δημιουργούμε τα ευρετήρια. Για την αρχικοποίηση της χρονικής περιόδου που μας ενδιαφέρει, οι χρόνοι έναρξης και λήξης είναι τύπου `datetime` για να μπορούν να γίνουν οι απαραίτητες συγκρίσεις στη MongoDB. Τα ερωτήματα είναι αλφαριθμητικά που συντάσσονται όπως τα `aggregation pipelines`. Κάθε στάδιο του pipeline κάνει διαφορετική επεξεργασία στα αποθηκευμένα documents και η έξοδος του αποτελεί είσοδο για το επόμενο στάδιο. Στο τέλος, τα ερωτήματα εκτελούνται και τα αποτελέσματά τους εμφανίζονται στην οθόνη.

Εξηγούμε τη λογική για κάθε query:

Σε όλα τα ερωτήματα το 1^ο στάδιο είναι το φιλτράρισμα των εγγράφων με πεδίο “time” εντός της χρονικής περιόδου που έχουμε ορίσει. Στη συνέχεια:

- 1) Για το ερώτημα 1 - ακμή με το μικρότερο πλήθος οχημάτων μεταξύ μιας προκαθορισμένης χρονικής περιόδου: εφαρμόζεται στα επεξεργασμένα δεδομένα από το spark. Οι εγγραφές ομαδοποιούνται με βάση το πεδίο “link”, δηλαδή ανά ακμή και υπολογίζεται το άθροισμα των “ncount” (=πλήθος οχημάτων) πεδίων τους. Το αποτέλεσμα είναι το πλήθος οχημάτων της κάθε ακμής σε ολόκληρη τη χρονική περίοδο.
- 2) Για το ερώτημα 2 – ακμή με τη μέγιστη μέση ταχύτητα μεταξύ μιας προκαθορισμένης χρονικής περιόδου: εφαρμόζεται στα επεξεργασμένα δεδομένα από το spark. Γίνεται η ίδια ομαδοποίηση, απλά αντί για πρόσθεση των ncount, υπολογίζεται ο μέσος όρος των “vspeed” (= μέση ταχύτητα) πεδίων τους. Το αποτέλεσμα είναι η μέση ταχύτητα όλων των οχημάτων που πέρασαν εντός ολόκληρης της χρονικής περιόδου για κάθε ακμή.
- 3) Για το ερώτημα 3 - μέγιστη διαδρομή σε μια προκαθορισμένη χρονική περίοδο: εφαρμόζεται στα ωμά δεδομένα από το spark. Οι εγγραφές ομαδοποιούνται με βάση το ζεύγος πεδίων (origin, destination) που συμβολίζει μια διαδρομή. Για κάθε ομάδα καταμετρείται το πλήθος των διαφορετικών ακμών που διασχίστηκαν. Η καταμέτρηση μόνο των διαφορετικών εξασφαλίζεται με την εισαγωγή τους σε σύνολο, συνεπώς αν μια ακμή συναντηθεί 2^η φορά, δεν θα ξανατοποθετηθεί στο σύνολο, αφού θα έχει τοποθετηθεί την πρώτη φορά που συναντήθηκε. Το αποτέλεσμα είναι το πλήθος ακμών ανά διαδρομή.

Όλα τα queries συνεχίζουν στα επόμενα στάδια με αύξουσα (το 1^ο)/φθίνουσα ταξινόμηση (τα 2 και 3) των αποτελεσμάτων και απομόνωση μόνο της πρώτης εγγραφής, για να πάρουμε το μέγιστο.

Στο επόμενο screenshot φαίνονται τα επιστρεφόμενα αποτελέσματα. Ο κώδικας για τα ερωτήματα υπάρχει στο αρχείο queries.py .

```
maria@maria-virtual-machine:~/BigDataProject/spark-3.5.2-bin-hadoop3$ python3 ../final/queries.py
Query 1 returned: [{'_id': 'I1N1', 'total_vcount': 155}]
Query 2 returned: [{'_id': 'I4S1', 'average_vspeed': 14.294117647058824}]
Query 3 returned: [{'_id': {'origin': 'W1', 'destination': 'N1'}, 'uniqueLinkCount': 5}]
maria@maria-virtual-machine:~/BigDataProject/spark-3.5.2-bin-hadoop3$
```

Εκτέλεση και αποτελέσματα του queries.py

4. Σχολιασμός αποτελεσμάτων

Το συγκεκριμένο project ήταν μια πολύ ενδιαφέρουσα προσέγγιση στο πρόβλημα της συλλογής, επεξεργασίας και αποθήκευσης μεγάλων δεδομένων. Το χτίσιμο του kafka-spark-mongodb pipeline είχε ξεκάθαρη λογική, ωστόσο, παρόλη την πληθώρα βιβλιογραφίας και οδηγιών για επιτυχή εγκατάσταση, η χρήση των σωστών connectors μεταξύ των εργαλείων αποδείχθηκε χρονοβόρα και πολύπλοκη διαδικασία.

5. Βιβλιογραφία

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

<https://spark.apache.org/downloads.html>

<https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>

<https://www.mongodb.com/docs/manual/core/indexes/create-index/#std-label-manual-create-an-index>

<https://www.mongodb.com/docs/manual/core/aggregation-pipeline/>