

Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου

Εργαστηριακή Άσκηση 2023/24

Όνομα	Επώνυμο	ΑΜ
ΒΑΣΙΛΕΙΟΣ	ΑΛΕΞΟΠΟΥΛΟΣ	1084625
ΝΙΚΟΛΑΟΣ	ΒΟΥΛΓΑΡΗΣ	1084626

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



19 / 9 / 2024

Υπογραφή



19 / 9 / 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	Περιέχει τον κώδικα για παραγωγή δεδομένων με εξομοιωτή <i>uxsim</i> .
<i>kafka_producer.py</i>	1	Περιέχει τον κώδικα για τον <i>Kafka producer</i> .
<i>kafka_consumer.py</i>	1	Περιέχει τον κώδικα για τον <i>Kafka producer</i> .
<i>spark_mongo_pipeline.py</i>	2,3	Περιέχει τον κώδικα για <i>real time</i> επεξεργασία των δεδομένων με <i>Spark</i> και αποθήκευση των επεξεργασμένων και μη

		δεδομένων σε μια <i>MongoDB database</i> .
<i>mongodb_queries.py</i>	3	Περιέχει τον κώδικα για υλοποίηση <i>queries</i> στη <i>MongoDB</i> .

Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

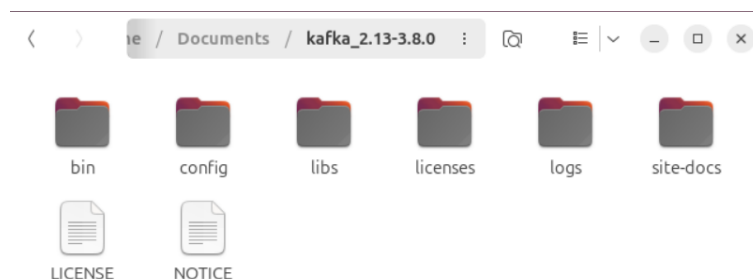
Για την εργασία χρησιμοποιήθηκε ένα Ubuntu 24.04.1 LTS (x86_64) Virtual Machine μέσω του προγράμματος Oracle VM VirtualBox σε λειτουργικό Windows 10. Τα χαρακτηριστικά της CPU, του αριθμού φυσικών πυρήνων και της δευτερεύουσας μνήμης αφορούν το φυσικό Η/Υ, ενώ τα υπόλοιπα αφορούν τους πόρους που διατίθενται στο VM.

Χαρακτηριστικό	Τιμή
CPU model	Intel(R) Core(TM) i7-1065G7
CPU clock speed	1.50 GHz
Physical CPU cores	4
Logical CPU cores	4
RAM	8
Secondary Storage Type	SSD

Ερώτημα 1: Παραγωγή δεδομένων

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε Python 3.12.3, Java 22.0.2 και για την ανάπτυξη κώδικα χρησιμοποιήθηκε Pycharm Professional Edition. Στο τοπικό της περιβάλλον εγκαταστάθηκαν όλα τα πακέτα της Python που χρειάστηκαν για την εργαστηριακή άσκηση.

Για την εγκατάσταση του Apache Kafka χρησιμοποιήθηκαν οι οδηγίες που παρέχονται στην επίσημη τους [ιστοσελίδα](#). Φαίνονται παρακάτω ο φάκελος που είναι αποθηκευμένα τα αρχεία του Kafka, καθώς και τα αποτελέσματα από την επιτυχή διαδικασία ενεργοποίησης του Kafka server με χρήση Zookeeper. Επίσης φαίνεται η επιτυχής δημιουργία topic ονόματι vehicle_positions με bootstrap-server localhost:9092. Το συγκεκριμένο topic θα χρησιμοποιηθεί και στο υπόλοιπο της εργασίας.



```
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0$ bin/zookeeper-server-start.sh config/zookeeper.properties
[2024-09-19 16:17:44,708] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,716] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,721] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,721] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,721] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,721] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-09-19 16:17:44,731] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-09-19 16:17:44,731] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-09-19 16:17:44,731] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
```

```
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0$ bin/kafka-server-start.sh config/server.properties
[2024-09-19 16:17:54,207] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.tools.Log4jControllerRegistration$)
[2024-09-19 16:17:54,871] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-09-19 16:17:54,879] 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
```

```
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0$ bin/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.
nikos@nikos-VirtualBox: ~/Documents/kafka_2.13-3.8.0$
```

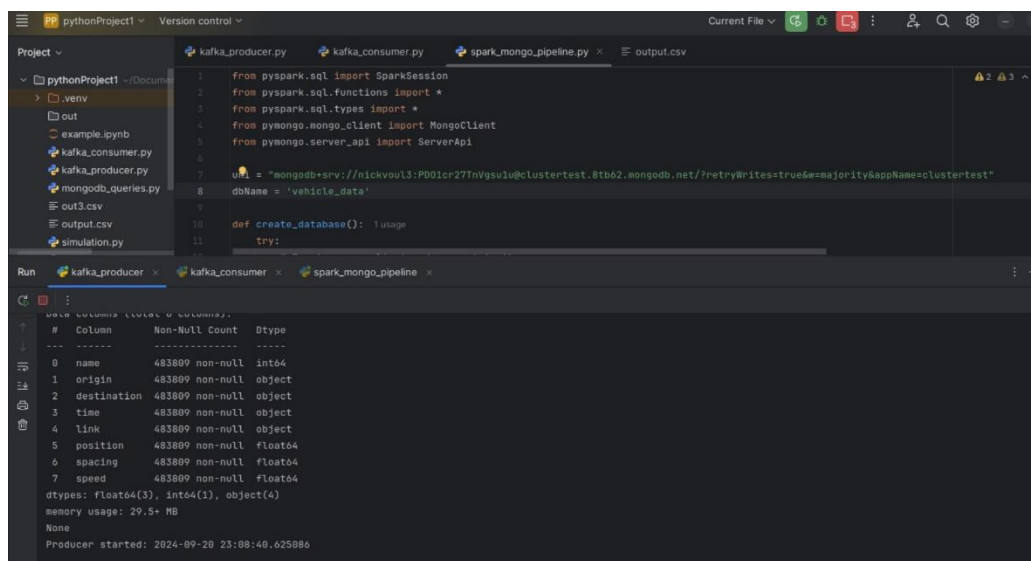
Για την παραγωγή δεδομένων χρησιμοποιείται απόσπασμα από την πρότυπη υλοποίηση σε Jupyter Notebook, που δίνεται μαζί με την εργασία, δημιουργώντας μια προσομοίωση διάρκειας μίας ώρας ενός απλού σεναρίου (αρχείο `simulation.py`). Τα παραγόμενα δεδομένα εξάγονται στο αρχείο `output.csv`, προκειμένου να χρησιμοποιηθούν από τους Kafka clients.

Εξετάζουμε το αρχείο `kafka_producer.py`. Λόγω ασυμβατότητας του πακέτου `kafka-python` με την έκδοση της Python που έχουμε εγκατεστημένη, προτείνεται η χρήση του πακέτου `kafka-python-ng`, το οποίο και εγκαταστήσαμε. Χρησιμοποιήσαμε επίσης `json`, `pandas` και `datetime` πακέτα για σωστή μορφοποίηση των δεδομένων που θα στείλουμε. Θέτουμε τις σωστές παραμέτρους για τη `KafkaProducer()` και διαβάζουμε τα δεδομένα από το `output.csv` σε ένα `df pandas DataFrame`. Μετονομάζουμε τις στήλες με βάση το παράδειγμα της αναφοράς. Υπολογίζουμε την ώρα εκκίνησης του producer μέσω της `datetime.now()` και την αποθηκεύουμε στη μεταβλητή `start..` Για το πεδίο "time", αλλάζουμε την τιμή σε `start + t` δευτερόλεπτα, με βάση την τιμή `t` που υπάρχει αρχικά στο

πεδίο. Το αποτέλεσμα μορφοποιείται σε %d/%m/%Y %H:%M:%S, για να ταιριάζει με το παράδειγμα. Αφήνουμε τα πεδία “dn” και “Unnamed” του output.csv εκτός του τελικού dataframe.

Έχοντας διαμορφώσει τα δεδομένα, τα στέλνουμε σε μηνύματα κάθε N=5 δευτερόλεπτα με χρήση ενός βρόχου. Σε κάθε επανάληψη στέλνονται τα δεδομένα για τα αυτοκίνητο εφόσον βρίσκεται ήδη σε κίνηση, εφόσον δηλαδή έχει στο πεδίο “link” τιμή διάφορη του “waiting_at_origin_node” και στο πεδίο “time” ίση με τον χρόνο που έχει περάσει. Τα δεδομένα για κάθε όχημα σε μια θέση στέλνονται σειριακά σε json. Στο αρχείο kafka_consumer.py αρχικοποιείται ένα KafkaConsumer() αντικείμενο, με ίδιες παραμέτρους με τον Producer, που εκτυπώνει τα δεδομένα που παραλαμβάνει στη κονσόλα.

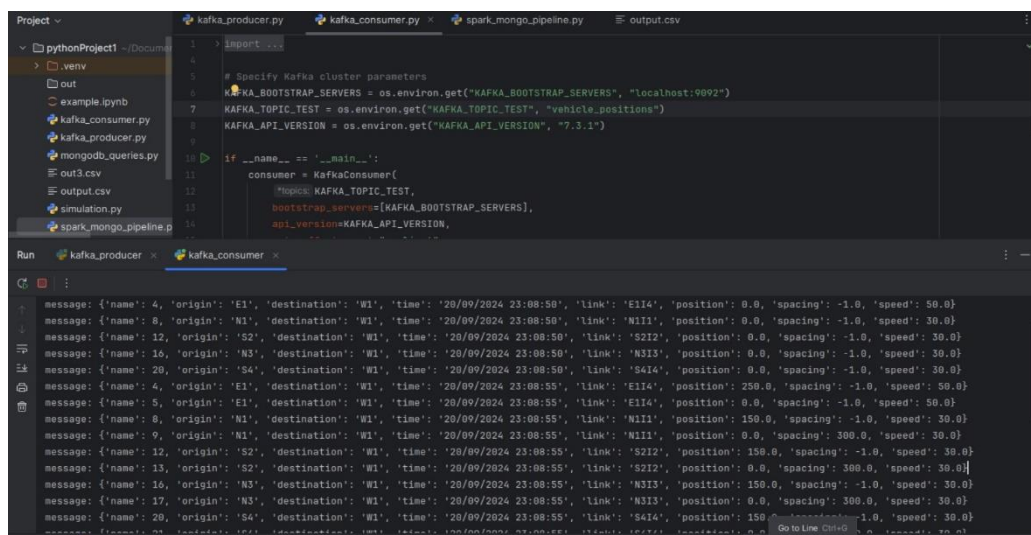
Παρακάτω φαίνονται τα αποτελέσματα από την επιτυχή εκτέλεση των 2 αρχείων.



```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.types import *
4 from pymongo.mongo_client import MongoClient
5 from pymongo.server_api import ServerApi
6
7 url = "mongodb+srv://nickvoul3:PD01cr27TnVgsul@cluster0-test.8tb62.mongodb.net/?retryWrites=true&majority=&appName=cluster0-test"
8 dbName = 'vehicle_data'
9
10 def create_database():
11     try:
```

#	Column	Non-Null Count	DType
0	name	483809 non-null	int64
1	origin	483809 non-null	object
2	destination	483809 non-null	object
3	time	483809 non-null	object
4	link	483809 non-null	object
5	position	483809 non-null	float64
6	spacing	483809 non-null	float64
7	speed	483809 non-null	float64

dtypes: float64(3), int64(1), object(4)
memory usage: 29.5+ MB
None
Producer started: 2024-09-20 23:08:48.625986



```
1 import json
2
3 # Specify Kafka cluster parameters
4 KAFKA_BOOTSTRAP_SERVERS = os.environ.get("KAFKA_BOOTSTRAP_SERVERS", "localhost:9092")
5 KAFKA_TOPIC_TEST = os.environ.get("KAFKA_TOPIC_TEST", "vehicle_positions")
6 KAFKA_API_VERSION = os.environ.get("KAFKA_API_VERSION", "7.3.1")
7
8 if __name__ == '__main__':
9     consumer = KafkaConsumer(
10         KAFKA_TOPIC_TEST,
11         bootstrap_servers=KAFKA_BOOTSTRAP_SERVERS,
12         api_version=KAFKA_API_VERSION,
```

```
message: {'name': 4, 'origin': 'E1', 'destination': 'W1', 'time': '20/09/2024 23:08:50', 'link': 'E114', 'position': 0.0, 'spacing': -1.0, 'speed': 50.0}
message: {'name': 8, 'origin': 'N1', 'destination': 'W1', 'time': '20/09/2024 23:08:50', 'link': 'N111', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 12, 'origin': 'S2', 'destination': 'W1', 'time': '20/09/2024 23:08:50', 'link': 'S212', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 16, 'origin': 'N3', 'destination': 'W1', 'time': '20/09/2024 23:08:50', 'link': 'N313', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 20, 'origin': 'S4', 'destination': 'W1', 'time': '20/09/2024 23:08:50', 'link': 'S414', 'position': 0.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 4, 'origin': 'E1', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'E114', 'position': 250.0, 'spacing': -1.0, 'speed': 50.0}
message: {'name': 8, 'origin': 'E1', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'E114', 'position': 0.0, 'spacing': -1.0, 'speed': 50.0}
message: {'name': 5, 'origin': 'N1', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'N111', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 9, 'origin': 'N1', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'N111', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
message: {'name': 12, 'origin': 'S2', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'S212', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 13, 'origin': 'S2', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'S212', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
message: {'name': 16, 'origin': 'N3', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'N313', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
message: {'name': 17, 'origin': 'N3', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'N313', 'position': 0.0, 'spacing': 300.0, 'speed': 30.0}
message: {'name': 20, 'origin': 'S4', 'destination': 'W1', 'time': '20/09/2024 23:08:55', 'link': 'S414', 'position': 150.0, 'spacing': -1.0, 'speed': 30.0}
```

Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

Για το συγκεκριμένο ερώτημα χρησιμοποιήθηκε Spark 3.5.2 με Hadoop3 και η έκδοση 3.5.2 του πακέτου rgsark. Για εγκατάσταση του Spark ακολουθήσαμε τις οδηγίες που παρατίθενται [εδώ](#).

Εξετάζουμε το αρχείο `spark_mongo_pipeline.py`, παραλείποντας τα κομμάτια που αφορούν το MongoDB, τα οποία θα εξηγηθούν στο επόμενο ερώτημα. Αρχικοποιούμε μια Spark session και για κάθε εξάρτηση χρησιμοποιούμε το αντίστοιχο πακέτο: `org.apache.spark:spark-streaming-kafka-0-10_2.12:3.5.2`, `org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.2` και `org.mongodb.spark: mongo-spark-connector_2.12:3.0.1`. Επίσης για χρήση στο επόμενο ερώτημα παραθέτουμε πληροφορίες για το URI και το όνομα της Mongo Database. Για να ελαττώσουμε την σύγχυση στη κονσόλα από τα πολλαπλά μηνύματα INFO καλούμε την `spark_session.sparkContext.setLogLevel("ERROR")`.

Θέλουμε να μεταδώσουμε τα δεδομένα από το kafka topic σε μορφή dataframe για να μπορέσουμε να τα επεξεργαστούμε. Για την σύνδεση με το topic χρησιμοποιούμε τις ίδιες παραμέτρους με το προηγούμενο ερώτημα. Υλοποιούμε ένα schema, το οποίο αντιπροσωπεύει για κάθε γραμμή τον τύπο δεδομένων που περιμένουμε να παραλάβουμε και αν μπορούν τα πεδία να έχουν τιμή null. Κάθε γραμμή διαβάζεται σαν JSON string, στο οποίο εφαρμόζεται το schema που φτιάξαμε πριν, προκειμένου να δημιουργήσουμε στήλες για τα αναγνωρισμένα πεδία. Τα δεδομένα αποθηκεύονται στο `raw_df` Dataframe. Θέλουμε για κάθε δέσμη δεδομένων να υπάρξει επεξεργασία, προκειμένου να δημιουργηθεί ένα νέο dataframe με πεδία που αναγράφονται στην εκφώνηση. Επομένως γράφουμε το stream των δεδομένων του `raw_df` και χρησιμοποιούμε το `foreachBatch` output sink του Spark για να επεξεργαστούμε την κάθε δέσμη(batch dataframe) που γράφεται.

Καλείται η `send_batch` για κάθε δέσμη και μέσα της καλείται για την επεξεργασία των δεδομένων η `processed_df()`. Θέλουμε για κάθε link να βρούμε το πλήθος οχημάτων και την μέση ταχύτητα των οχημάτων μέσα του για την συγκεκριμένη χρονική στιγμή. Επομένως, ομαδοποιούμε τα δεδομένα με βάση τα πεδία "time", "link" και τους κάνουμε aggregation με βάση το πλήθος των οχημάτων ("`count(*) as vcount`") και τη μέση τιμή της ταχύτητας τους ("`avg(speed) as vspeed`").

Παρακάτω φαίνονται τα αποτελέσματα από την εκτέλεση της `show()` σε κάθε παρτίδα δεδομένων, πριν και μετά την επεξεργασία που τους κάνουμε.

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.types import *
4 from pymongo.mongo_client import MongoClient
5 from pymongo.server_api import ServerApi
6
7 uri = "mongodb+srv://nickvoul3:PD01cr27TnVgsulu@clustertest.8tb62.mongodb.net/?retryWrites=true&majorityAppName=clustertest"
8 dbName = 'vehicle_data'
9
10 def create_database():
11     try:
12         # Create database and collection
13         client = MongoClient(uri)
14         db = client[dbName]
15         collection = db.create_collection('raw_data')
```

id	origin	destination	time	link	position	spacing	speed
4	E1	W1	20/09/2024 23:09:45	I11W1	250.0	-1.0	0.0
5	E1	W1	20/09/2024 23:09:45	I211	250.0	-1.0	0.0
6	E1	W1	20/09/2024 23:09:45	I312	391.66666	-1.0	50.0
7	E1	W1	20/09/2024 23:09:45	I312	216.66667	-1.0	6.666665
11	N1	W1	20/09/2024 23:09:45	N111	500.0	-1.0	0.0
15	S2	W1	20/09/2024 23:09:45	S212	500.0	-1.0	0.0
18	N3	W1	20/09/2024 23:09:45	I11W1	41.666668	-1.0	41.666668
19	N3	W1	20/09/2024 23:09:45	N313	500.0	-1.0	0.0
20	S4	W1	20/09/2024 23:09:45	I11W1	250.0	-1.0	0.0
21	S4	W1	20/09/2024 23:09:45	I211	216.66667	-1.0	6.666665
22	S4	W1	20/09/2024 23:09:45	I211	41.666668	-1.0	41.666668

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.types import *
4 from pymongo.mongo_client import MongoClient
5 from pymongo.server_api import ServerApi
6
7 uri = "mongodb+srv://nickvoul3:PD01cr27TnVgsulu@clustertest.8tb62.mongodb.net/?retryWrites=true&majorityAppName=clustertest"
8 dbName = 'vehicle_data'
9
10 def create_database():
11     try:
12         # Create database and collection
13         client = MongoClient(uri)
14         db = client[dbName]
15         collection = db.create_collection('raw_data')
```

time	link	vcount	vspeed
20/09/2024 23:09:50	I211	4	42.9166658772095
20/09/2024 23:09:50	I413	2	50.0
20/09/2024 23:10:00	N313	7	12.857142857142858
20/09/2024 23:09:45	S14	1	50.0
20/09/2024 23:09:55	S212	7	9.285714285714286
20/09/2024 23:10:00	I11W1	1	0.0
20/09/2024 23:09:55	I11W1	1	0.0
20/09/2024 23:09:55	trip_end	1	-1.0
20/09/2024 23:09:55	S414	7	9.285714285714286
20/09/2024 23:09:55	S212	2	15.0

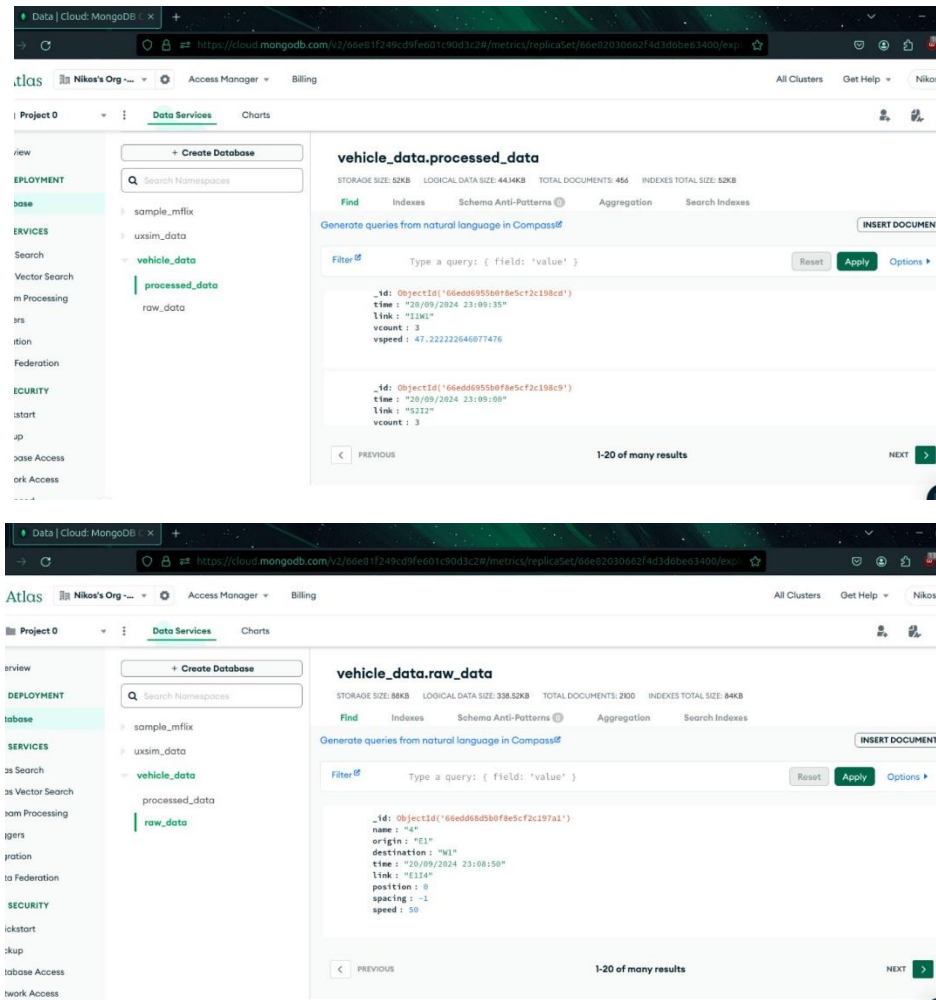
Ερώτημα 3: Αποθήκευση σε MongoDB

Για το συγκεκριμένο ερώτημα δε εγκαταστάθηκε το MongoDB τοπικά, φτιάξαμε λογαριασμό και είχαμε πρόσβαση σε ένα M0 Sandbox cluster ονόματι clustertest, με το οποίο συνδεόμαστε μέσω ενός URI που μας παρείχαν. Χρησιμοποιούμε επίσης τα πακέτα pymongo[srv] και pyspark-mongodb. Θα φτιαχτεί ένα database ονόματι “vehicle_data” με 2 συλλογές, “raw_data” και “processed_data”. Σε αυτά αποθηκεύονται ως documents οι δέσμες δεδομένων όπως έρχονται στο Spark από τον Kafka και όπως είναι μετά την επεξεργασία αντίστοιχα.

Στο αρχείο spark_mongo_pipeline.py, προτού ακολουθήσουμε την διαδικασία που περιγράφεται στο προηγούμενο ερώτημα καλείται αρχικά η create_database(). Μέσα στην συνάρτηση δημιουργείται ένας client για εγκαθίδρυση σύνδεσης με το server. Ελέγχουμε αν το όνομα της καινούργιας database υπάρχει ήδη στην λίστα με τα ονόματα του server. Αν δεν υπάρχει, η MongoDB δημιουργεί την database και τις συλλογές της μόνο αφού έχουν εισαχθεί document σε αυτή, επομένως θα εισάγουμε dummy documents για αρχή.

Ακολουθώντας την διαδικασία του προηγούμενου ερωτήματος, έχουμε πλέον μια δεσμίδα δεδομένων πριν και μετά την επεξεργασία τους. Στη πρώτη περίπτωση, μέσω της *raw_mongo_pipeline()*, αποθηκεύουμε τις δεσμίδες στη “raw_data” συλλογή, δίνοντας τις λεπτομέρειες για το URI του server και το όνομα της database. Αντίστοιχα στην δεύτερη περίπτωση μέσω της *processed_df()* στη “processed_data” συλλογή.

Παρακάτω φαίνονται μερικά από τα documents που είναι αποθηκευμένα στη βάση.



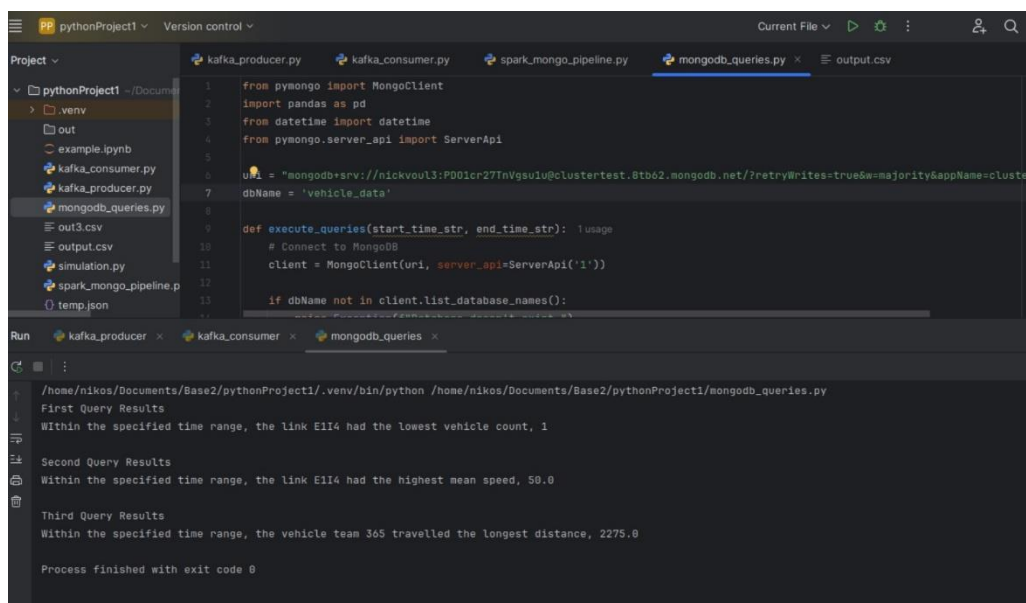
Εξετάζουμε το αρχείο *mongodb_queries*, στο οποίο υλοποιούνται τα 3 queries που ζητούνται στην MongoDB. Παρέχονται αρχικά 2 ημερομηνίες που διαμορφώνουν την χρονική περίοδο της αναζήτησης. Καλείται η *execute_queries()* με όρισμα τις ημερομηνίες. Ελέγχεται αν υπάρχει το όνομα της βάσης στη λίστα του server και εγκαθιδρύεται σύνδεση με αυτόν. Στη συνέχεια μετατρέπονται οι ημερομηνίες σε μορφή *%d/%m/%Y %H:%M:%S*, προκειμένου να αναγνωρίζονται μες στα query. Για κάθε query ακολουθείται η εξής διαδικασία: τα αποτελέσματα του query μετατρέπονται σε list object, ελέγχεται αν το list είναι κενό, μετατρέπεται σε pandas Dataframe για ευκολότερη επεξεργασία και εκτυπώνονται οι τιμές των κατάλληλων πεδίων του πρώτου στοιχείου αυτού του dataframe.

Για το πρώτο query, ψάχνουμε στη “processed_data” συλλογή, όλα τα στοιχεία με τιμή “time” μέσα στο διάστημα που ορίζουν οι *start_time* και *end_time* μεταβλητές. Τα αποτελέσματα ταξινομούνται κατά αύξουσα σειρά με βάση την τιμή του πεδίου “vcount”. Έτσι βρίσκουμε το link με το μικρότερο πλήθος οχημάτων μέσα στη προκαθορισμένη χρονική περίοδο. Για το δεύτερο query, κάνουμε παρόμοια αναζήτηση με το πρώτο, αλλά

ταξινομούμε κατά φθίνουσα σειρά με βάση την τιμή του πεδίου “vspeed”. Έτσι βρίσκουμε το link με τη μεγαλύτερη μέση ταχύτητα μέσα στη προκαθορισμένη χρονική περίοδο.

Για το τρίτο query ζητάμε την μεγαλύτερη διαδρομή στη προκαθορισμένη χρονική περίοδο. Για να το βρούμε αυτό πρέπει να ελέγξουμε για κάθε αυτοκίνητο ξεχωριστά από ποια link πέρασε μέχρι να φτάσει στον προορισμό του και πόσα μέτρα διένυσε εντός τους μέχρι να φτάσει στο τέλος του. Το άθροισμα των αποστάσεων που διένυσε εντός του κάθε link θα είναι η τελική διαδρομή του. Φιλτράρουμε αρχικά τα αποτελέσματα του query για να κρατήσουμε τα δεδομένα που αφορούν τη προκαθορισμένη χρονική περίοδο. Έπειτα κάνουμε aggregation στην “raw_data” συλλογή, με ταξινόμηση αρχικά της συλλογής με βάση τα “name”, “link” και εύρεση της μέγιστης τιμής “position” (απόστασης δηλαδή από την αρχή του link) για το κάθε όχημα. Στη συνέχεια ξανακάνουμε ταξινόμηση με βάση το “name” και υπολογίζουμε το άθροισμα όλων των τιμών “maxPosition” για το συγκεκριμένο όχημα. Τέλος ταξινομούμε τα αποτελέσματα κατά φθίνουσα σειρά με βάση την τιμή “totalDistance”. Επιστρέφεται το όχημα που διένυσε την μεγαλύτερη απόσταση. Παρατηρούμε εδώ ότι υπάρχει μεγάλη περίπτωση να υπάρχουν πολλαπλά οχήματα που έχουν ίδια τιμή “totalDistance”, άρα κάθε εκτέλεση του κώδικα θα επιστρέψει ένα από αυτά.

Παρακάτω φαίνονται τα αποτελέσματα από την εκτέλεση των 3 queries.



```
from pymongo import MongoClient
import pandas as pd
from datetime import datetime
from pymongo.server_api import ServerApi

uri = "mongodb+srv://niksvoul3:P001cr27TnVgsulu@cluster0.mongodb.net/?retryWrites=true&w=majority&appName=cluster0"
dbName = 'vehicle_data'

def execute_queries(start_time_str, end_time_str):
    # Connect to MongoDB
    client = MongoClient(uri, server_api=ServerApi('1'))

    if dbName not in client.list_database_names():
        raise ValueError(f"Database {dbName} does not exist")

    # Query 1: Find the link with the lowest vehicle count
    query1 = {'time': {'$gt': start_time_str, '$lt': end_time_str}}
    result1 = client[dbName].aggregate([{'$match': query1}, {'$group': {'_id': '$link', 'count': {'$sum': 1}}}, {'$sort': {'count': 1}}])

    # Query 2: Find the link with the highest mean speed
    query2 = {'time': {'$gt': start_time_str, '$lt': end_time_str}}
    result2 = client[dbName].aggregate([{'$match': query2}, {'$group': {'_id': '$link', 'mean_speed': {'$avg': '$vspeed'}}}, {'$sort': {'mean_speed': -1}}])

    # Query 3: Find the vehicle with the longest distance
    query3 = {'time': {'$gt': start_time_str, '$lt': end_time_str}}
    result3 = client[dbName].aggregate([{'$match': query3}, {'$group': {'_id': '$name', 'total_distance': {'$sum': '$distance'}}}, {'$sort': {'total_distance': -1}}])

    return result1, result2, result3
```

Run /home/nikos/Documents/Base2/pythonProject1/.venv/bin/python /home/nikos/Documents/Base2/pythonProject1/mongodb_queries.py

First Query Results
Within the specified time range, the link E114 had the lowest vehicle count, 1

Second Query Results
Within the specified time range, the link E114 had the highest mean speed, 50.0

Third Query Results
Within the specified time range, the vehicle team 365 travelled the longest distance, 2275.0

Process finished with exit code 0

Σχολιασμός αποτελεσμάτων

Συνοψίζοντας τα αποτελέσματα της εμπειρίας μας από την εργαστηριακή άσκηση, έχουμε σίγουρα εξοικειωθεί με τα σύγχρονα εργαλεία ανάλυσης δεδομένων και τις διαφορετικές λειτουργίες τους καθώς και την χρησιμότητά τους για την παραγωγή, επεξεργασία και αποθήκευση δεδομένων σε πραγματικό χρόνο. Το γεγονός ότι πρόκειται για εργαλεία open source μας έδειξε τη δύναμή τους στην ανάπτυξη έργων μεγάλης κλίμακας, χωρίς την ανάγκη ακριβών και κλειστών πλατφορμών. Η εργασία μας έδωσε επίσης μια ιδέα για την διαχείριση μεγάλων όγκων δεδομένων, και για τα προβλήματα κλιμάκωσης που προκύπτουν, μέσω της χρήσης κατακευκασμένων συστημάτων αποθήκευσης και επεξεργασίας. Ήταν ενδιαφέρον, ύστερα και από την απασχόληση μας με παραδοσιακές σχεσιακές βάσεις δεδομένων να παρατηρούμε την αποθήκευση τέτοιου όγκου δεδομένων σε μερικά λεπτά και εκτέλεση queries μέσα σε δευτερόλεπτα.

Βιβλιογραφία

- **Apache Kafka Quickstart** [\[1\]](#)
- **Apache Kafka with Python** [\[2\]](#)
- **Example Spark 3.0.1 Data Transformations in Python** [\[3\]](#)
- **Working with JSON in Apache Spark** [\[4\]](#)
- **What and How to use from_json and explode functions a json column in csv files with PySpark** [\[5\]](#)
- **PySpark MongoDB Pipeline | Setup Tutorial** [\[6\]](#)
- **Getting started with MongoDB, PySpark, and Jupyter Notebook** [\[7\]](#)
- **MongoDB Manual Operators** [\[8\]](#)
- **MongoDB Manual Window Fields** [\[9\]](#)