Η εργασία αποτελείται από 3 ζητούμενα. Πραγματοποιείται επεξεργασία ενός αρχείου με την χρήση του Apache Spark και συγκεκριμένα το pyspark. Το παρόν έγγραφο παρουσιάζει τα προβλήματα που εμφανίστηκαν κατά τη λύση αυτών καθώς και οδηγίες εκτέλεσης της εφαρμογής.

Προβλήματα που αντιμετώπισα

Κατά την εισαγωγή της εικονικής μηχανής στον προσωπικό υπολογιστή (λειτουργικό σύστημα Fedora Linux έκδοση 41) και στην προσπάθεια να εκκινηθεί, εμφανίστηκε πρόβλημα με το VBoxGuesAdditions.iso. Μετακινήθηκα στις *Ρυθμίσεις>Αποθήκευση>Συσκευές>Ελεγκτής:IDE* όπου απενεργοποίησα το ISO. Στην επόμενη προσπάθειά μου για εκκίνηση, μου εμφανίστηκε το παρακάτω σφάλμα:

```
NAT#0: configuration error: failed to set up redirection of 80 to 80. Probably a conflict with existing services or other rules (VERR_NAT_REDIR_SETUP).
Failed to attach the network LUN (VERR_NAT_REDIR_SETUP).
Result Code:
NS_ERROR_FAILURE (0x80004005)
Component:
ConsoleWrap
Interface:
IConsole {6ac83d89-6ee7-4e33-8ae6-b257b2e81be8}
```

Μετακινήθηκα *Ρυθμίσεις>Δίκτυο* και άλλαξα από NAT σε Γεφυρωμένη Κάρτα

Στην επόμενή μου προσπάθεια, εμφανίστηκε το παρακάτω σφάλμα:

```
VirtualBox can't operate in VMX root mode. Please disable the KVM kernel extension, recompile your kernel and reboot (VERR_VMX_IN_VMX_ROOT_MODE).
Result Code:
NS_ERROR_FAILURE (0x80004005)
Component:
ConsoleWrap
Interface:
IConsole {6ac83d89-6ee7-4e33-8ae6-b257b2e81be8}
```

Υπήρχε σφάλμα στο KVM. Από το Stackoverflow βρήκα την λύση να απενεργοποιήσω προσωρινά το KVM με τις εντολές:

sudo rmmod kvm_intel
sudo rmmod kvm

Ενώ εάν θέλω να γίνει μόνιμα, μπορώ να το κάνω με τις εντολές:

echo "blacklist kvm" | sudo tee -a /etc/modprobe.d/blacklist.conf
echo "blacklist kvm_intel" | sudo tee -a /etc/modprobe.d/blacklist.conf

Επίσης κάτι άλλο που έπρεπε να αλλάξω ήταν στις *Ρυθμίσεις>Οθόνη>Ελεγκτής γραφικών* να το γυρίσω σε **VMSVGA**.

Τελικά κατάφερε να εκκινηθεί η εικονική μηχανή. Με την προσπάθειά μου να εισέλθω στο openeclass για να κατεβάσω τα αρχεία, δεν είχα δίκτυο.

Λύση με Docker

Αφού δεν μπόρεσα να χρησιμοποιήσω την εικονική μηχανή, αναζήτησα την λύση του Docker. Εάν δεν είχα επιτυχία, θα έπρεπε να εγκαταστήσω τα λογισμικά στον υπολογιστή μου.

Μετά από πολλές δοκιμές, κατέληξα στο image https://hub.docker.com/r/apache/spark-py

Για λήψη του image, εκτέλεσα την εντολή:
docker pull apache/spark-py

Με την παρακάτω εντολή, είχα πρόσβαση στο Apache Spark
docker run -d -it --name spark apache/spark-py /opt/spark/bin/pyspark

Αντί για το όνομα **spark** μπορούμε, στις παρακάτω εντολές μπορούμε να χρησιμοποιήσουμε και το CONTAINER ID (το βλέπουμε με την εντολή **docker ps**).

Καλύτερα να έχουμε πρόσβαση στο bash με την εντολή:
docker exec -it spark bash

Στην αρχή εκτέλεσα το docker στην διεπαφή με το pyspark
docker exec -it spark /opt/spark/bin/pyspark

Εκεί δοκίμασα μια προς μια τις εντολές που χρειαζόμουν για να λύσω τα 3 ζητήματα. Οπότε στην συνέχεια δοκίμασα να αποθηκεύσω τις εντολές σε ένα αρχείο file.py και να δοκιμάσω να το εκτελέσω ως script στο bash.

Δεν χρησιμοποίησα volume, και για την αντιγραφή των αρχείων χρησιμοποίησα τις εντολές:
docker cp customer_shopping_data.csv spark:/opt/spark/work-dir/customer_shopping_data.csv

Για την αντογραφή του αρχείου από την εργασία του κου Κασκάλη
docker cp output_frontend.csv spark:/opt/spark/work-dir/output_frontend.csv

Με τον ίδιο τρόπο, αντέγραψα τα αρχεία που είχα ονομάσει ανάλογα:
docker cp first.py spark:/opt/spark/work-dir/first.py
docker cp second.py spark:/opt/spark/work-dir/second.py
docker cp third-1.py spark:/opt/spark/work-dir/third-1.py
docker cp third-2.py spark:/opt/spark/work-dir/third-2.py

Η εντολή εκτέλεσης μέσα από το bash του container γίνεται με τις εντολές:
/opt/spark/bin/spark-submit --master local[*] first.py
/opt/spark/bin/spark-submit --master local[*] second.py

Πρώτα εκτελούμε την πρώτη εντολή και περιμένουμε μέχρι δείχνει πολλές γραμμές:
INFO InMemoryFileIndex: It took 0 ms to list leaf files for 1 paths.
/opt/spark/bin/spark-submit --master local[*] third-1.py
Μετά εκτελούμε τον κώδικα (βγάζει τα αποτελέσματα από το αρχείο του κου Κασκάλη)
/opt/spark/bin/spark-submit --master local[*] third-2.py

Και για την λήψη των αποτελεσμάτων στο σύστημά μου (κάθε σειρά είναι μια γραμμή):
ΖΗΤΗΜΑ 1
docker cp spark:/opt/spark/work-dir/transactions_by_item_count ./transactions_by_item_count

docker cp spark:/opt/spark/work-dir/transactions_with_total ./transactions_with_total

ΖΗΤΗΜΑ 2
docker cp spark:/opt/spark/work-dir/shopping_mall_statistics ./shopping_mall_statistics

ΖΗΤΗΜΑ 3
docker cp spark:/opt/spark/work-dir/updated_stats ./updated_stats

Για διακοπή του container δίνουμε την εντολή:
docker stop spark

Για εκκίνηση ενός σταματημένου container δίνουμε την εντολή:
docker start spark

Ενώ για διαγραφή από το σύστημά μας δίνουμε την εντολή:
docker rm spark

ΣΗΜΕΙΩΣΗ: Όπου εμφανίζονται μόνο τα αρχεία (χωρίς διαδρομή), αυτό σημαίνει ότι η εντολή εκτελείται στον κατάλογο που βρίσκονται τα αρχεία. Για λόγους ευκολίας, τα έχω χωρίσει ανάλογα με τα ζητήματα. Οι ονομασίες των αρχείων ακολουθούν στην ανάλυση.

ΖΗΤΗΜΑ 1

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import  col, regexp_replace, upper
# from pyspark.sql.functions import round

# Δημιουργία ενός SparkSession
spark = SparkSession.builder.appName("RetailTransactionProcessing").getOrCreate()

# Φόρτωση του αρχείου CSV σε DataFrame. Ο κατάλογος είναι αυτός μέσα από το docker
df = spark.read.option("header",
"true").csv("file:////opt/spark/work-dir/customer_shopping_data.csv")

# 1. Αντικατάσταση του κενό με κάτω παύλα στις στήλες payment_method και shopping_mall
df = df.withColumn("payment_method", regexp_replace(col("payment_method"), " ", "_"))
df= df.withColumn("shopping_mall", regexp_replace(col("shopping_mall"), " ", "_"))

# 2. Μετατροπή των ονομάτων του shopping_mall σε κεφαλαία
df = df.withColumn("shopping_mall", upper(col("shopping_mall")))

# 3. Υπολογισμός του πλήθους των συναλλαγών
transaction_count = df.count()
print(f"Πλήθος συναλλαγών: {transaction_count}")

# 4. Μετατροπή της τιμής από TL σε ευρώ (1 TL = 0.1 EUR)
df = df.withColumn("price_eur", col("price") * 0.1)
```

```
# 5. Υπολογισμός του πλήθους των συναλλαγών ανά πλήθος αντικειμένων και αποθήκευση σε CSV
transaction_items_count = df.groupBy("quantity").count()
transaction_items_count.write.format('csv').option("header", True).mode('overwrite').save("transactions_by_item_count")

# 6. Υπολογισμός του συνολικού ποσού που δαπανήθηκε και αποθήκευση σε αρχείο CSV
df = df.withColumn("total", col("price") * col("quantity"))
df.write.format('csv').option("header", True).mode('overwrite').save("transactions_with_total")

# Σταματάμε το SparkSession μετά την ολοκλήρωση
spark.stop()
```

##

ZHTHMA 2

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum, count

# Δημιουργία ενός SparkSession
spark = SparkSession.builder.appName("StatisticsTable").getOrCreate()

# Φόρτωση του αρχείου CSV σε DataFrame. Ο κατάλογος είναι αυτός μέσα από το docker
df = spark.read.option("header", "true").csv("file:////opt/spark/work-dir/customer_shopping_data.csv")

# Υπολογισμός του συνολικού ποσού για κάθε εγγραφή
df = df.withColumn("quantity", col("quantity").cast("int"))
df = df.withColumn("price", col("price").cast("double"))
df = df.withColumn("total", round(col("price") * col("quantity"), 2))

# Υπολογισμός των στατιστικών ανά εμπορικό κέντρο
total_spent = df.groupBy("shopping_mall").sum("total")
total_quantity = df.groupBy("shopping_mall").sum("quantity")
total_transactions = df.groupBy("shopping_mall").count()

# Ενοποίηση των αποτελεσμάτων σε ένα DataFrame (μια γραμμή)
statistics_df = total_spent.join(total_quantity, "shopping_mall").join(total_transactions, "shopping_mall")

# Αποθήκευση του στατιστικού πίνακα σε αρχείο CSV (μια γραμμή)
statistics_df.write.format('csv').option("header", True).mode('overwrite').save("shopping_mall_statistics")

# Σταματάμε το SparkSession μετά την ολοκλήρωση
spark.stop()
```

ΣΗΜΕΙΩΣΗ: Στα αποτελέσματα, κάποιες φορές, η στήλη sum(total), μου βγάζει αποτελέσματα πχ 50554231,1000006 αλλά στο πρόγραμμα λογιστικού φύλλου το δείχνει 505542E+07. Όμως αν αλλάξω την μορφή του κελιού σε αριθμό το εμφανίζει κανονικά.

##

ΖΗΤΗΜΑ 3

Στο ζήτημα 3, ανοίγω 2 τερματικά. Στο πρώτο εκτελώ το script third-1.py και στο άλλο το third-2.py. Στο πρώτο τερματικό περιμένω να μου βγάλει συνεχόμενες γραμμές
INFO InMemoryFileIndex: It took 0 ms to list leaf files for 1 paths.

third-1.py

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType
from pyspark.sql.functions import col, sum, when

# Δημιουργία ενός SparkSession
spark = SparkSession.builder.appName("Streaming").getOrCreate()

# Ορισμός του σχήματος για το αρχείο με τα υφιστάμενα στατιστικά
existing_stats_schema = (
    StructType()
    .add('shopping_mall', 'string')
    .add('sum_total', 'double')
    .add('sum_quantity', 'integer')
    .add('transaction_count', 'integer')
)

# Φόρτωση του αρχείου με τα υφιστάμενα στατιστικά (ΠΡΟΗΓΟΥΜΕΝΟ ΖΗΤΗΜΑ)
existing_stats_df = (
    spark.read
    .option("header", "true")
    .schema(existing_stats_schema)
    .csv("/opt/spark/work-dir/shopping_mall_statistics/*.csv")
)

# Ορισμός του σχήματος για το streaming DataFrame
streaming_schema = (
    StructType()
    .add('shopping_mall', 'string')
    .add('total_spent', 'double')
    .add('total_quantity', 'integer')
    .add('transactions', 'integer')
)

# Διαβάζουμε τα νέα δεδομένα ως streaming
streaming_df = (
    spark.readStream
```

```
    .format("csv")
    .option('sep', ',')
    .option("header", "true")
    .schema(streaming_schema)
     .option("path", "/opt/spark/work-dir/input_frontend/*.csv") # Τον φάκελο input_frontend το
είχα δημιουργήσει ήδη.
    .load()
)

def process_batch(df_batch, batch_id):
    # Σύνδεση (join) των δύο πινάκων με βάση τη στήλη shopping_mall
    combined_df = existing_stats_df.join(
        df_batch,
        on="shopping_mall",
        how="outer"
    )

      # Υπολογισμός των συνολικών στατιστικών (βάζω isNotNull για περιπτώσεις που δεν
υπάρχει αριθμός, όπως και είχα περίπτωση)
    updated_stats_df = combined_df.groupBy("shopping_mall").agg(
        (
            sum(when(col("sum_total").isNotNull(), col("sum_total")).otherwise(0)) +
            sum(when(col("total_spent").isNotNull(), col("total_spent")).otherwise(0))
        ).alias("total_spent"),

        (
            sum(when(col("sum_quantity").isNotNull(), col("sum_quantity")).otherwise(0)) +
            sum(when(col("total_quantity").isNotNull(), col("total_quantity")).otherwise(0))
        ).alias("total_quantity"),

        (
                                                sum(when(col("transaction_count").isNotNull(),
col("transaction_count")).otherwise(0)) +
            sum(when(col("transactions").isNotNull(), col("transactions")).otherwise(0))
        ).alias("transactions")
    )

    # Αποθήκευση των νέων στατιστικών σε CSV
          updated_stats_df.write.mode("overwrite").format('csv').option("header",
True).save("/opt/spark/work-dir/updated_stats")

# Εκκίνηση της streaming εφαρμογής
query = (
    streaming_df.writeStream
    .outputMode("update")
    .foreachBatch(process_batch)
    .option("checkpointLocation", "/opt/spark/work-dir/checkpoint")
    .start()
```

```
)

# Αναμονή για την ολοκλήρωση της ροής
query.awaitTermination()
```

third-2.py

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType
from pyspark.sql.functions import col, sum, count

# Δημιουργία ενός SparkSession
spark = SparkSession.builder.appName("FrontendStatistics").getOrCreate()

# Φόρτωση του αρχείου CSV σε DataFrame. Ο κατάλογος είναι αυτός μέσα από το docker
df = spark.read.option("header", "true").csv("output_frontend.csv")

# Υπολογισμός του συνολικού ποσού για κάθε εγγραφή
df = df.withColumn("quantity", col("quantity").cast("int"))
df = df.withColumn("total", col("price") * col("quantity"))

# Υπολογισμός των στατιστικών ανά εμπορικό κέντρο
total_spent = df.groupBy("shopping_mall").sum("total")
total_quantity = df.groupBy("shopping_mall").sum("quantity")
total_transactions = df.groupBy("shopping_mall").count()

# Ενοποίηση των αποτελεσμάτων σε ένα DataFrame
statistics_df = total_spent.join(total_quantity, "shopping_mall").join(total_transactions,
"shopping_mall")

# Αποθήκευση του στατιστικού πίνακα σε αρχείο CSV
statistics_df.write.format('csv').option("header",
True).mode('overwrite').save("input_frontend")

# Σταματάμε το SparkSession μετά την ολοκλήρωση
spark.stop()
```

Μετά το πέρας της δημιουργίας του αρχείου των στατιστικών του αρχείου του Κασκάλη, περιμένουμε λίγο και σταματάμε το τερματικό που εκτελείται το αρχείο third-1.py.