Customer Profiling

January 27, 2021

Streaming Part Proposal Overview

The proposed new streaming solution uses Kafka as the streaming platform and Faust as the stream processing library for this task. Multiple streams are created from different data sources (transfer_history, invoice_history, last_login_history), to be transformed into meaningful indicators for customer profiling (clustering).

Kafka Producer

The kafka_producer.py creates 2 separate streams, one for invoices_histroy and the other for transfers_history. Transfers_histroy gets streamed row-based to 'transfers' topic in JSON format. The JSON format follows a pre-defined schema for a transfer instance.

The producer publishes the invoice history data from the CSV files in the format of a serialized JSON object per row that contains the header and row in key-value pairs. The rows are being streamed at 1 row/2 microseconds frequency i.e. 500 rows per second.

Kafka Consumers

On the consumer side, the stream is consumed to be transformed into meaningful metrics for the clustering process. Kafka Streams is a library that allows us to do per-event processing of the streamed records. Since we are using python as the programming language for our development environment, we chose to use Faust, a stream processing library, porting the ideas from Kafka Streams to Python [1]. Since there is no direct API for KafkaStreams that exists for python [2], Faust is considered the KafkaStreams equivalent in python.

1. transfers_consumer

   The consumer reads data from the 'transfers' topic and utilizes it along with customers' metadata to calculate clustering metrics. In the transfers scope, We chose to cluster the

customers into 3 groups depending on ***customer transfer activity***. The 3 groups are “High”, “Steady”, “Low”, expressing the activity based on thresholding. Customer transfer activity is calculated as a combination of ***customer account life*** and ***transfer rate*** metrics.

Transfer rate of a customer is the ratio between the sum of all transfer amounts to the number of transfers. Depending on the rate value predefined thresholds, customers’ rates are clustered into “High”, “Moderate”, ”Low”. For this purpose, total transfer amount and number of transfers is persisted. For every incoming transfer event, the number of transfers is incremented and the transfer amount is added to the total amount. This is done based on the PARTY_ID filtering. Afterwards, the transfer rate is calculated and classified into a cluster.

Customer account life is the period of time since the customer’s account creation till the present date. For control purposes, we chose the present date to be January 01 2021. Depending on this period, customers’ life are classified into “Mature”, “Moderate”, “New”.

Finally, the customer transfer activity combines both metrics as follows:

- Life Rank (i.e. Moderate) = Transfer Rate Rank (i.e Moderate), then this customer is clustered in “Steady Activity” group.
- Life Rank (i.e. Moderate) > Transfer Rate Rank (i.e Low), then this customer is clustered in “Low Activity” group.
- Life Rank (i.e. Moderate) < Transfer Rate Rank (i.e High), then this customer instance is clustered in “High Activity” group.

A Customer’s cluster keeps getting modified every time a transfer arrives to the system that belongs to this customer.

2. Invoices_consumer

The consumer reads data from ‘invoices’ topic and utilizes it along with the customers’ metadata to calculate clustering metrics. In this process, we decided to cluster the customers in 3 groups based on the following metrics: average invoice payable amount per enterprise, enterprise size, and global coverage. The groups, “Steady Customers”, “Under-performing Customers”, and “Developed Customers”, express the activity based on thresholding the mentioned metrics.

For every incoming invoice event, we filter it out first based on PARTY_IDs in the customer metadata. Then we filter it further to exclude the cancelled invoices (negative amounts). Since our invoices can be in different currencies, we then convert it into HUF using the *forex-python* library if it's a foreign currency and persist the count of different currencies encountered to measure the global coverage of the enterprise. We also persist the sum of the invoice payable amount and the count of invoice events received over time per customer, which we then use to calculate the average invoice payable amount. In order to get a proper metric for the amount, we try to normalize the average amount by dividing it with the enterprise size of the customer. Then, based on some predefined thresholds, we cluster the resulting amount into 3 ranks, 'S' (small transaction amount), 'M' (medium transaction amount), and 'L' (large transaction amount).

We also cluster the customers according to the enterprise sizes into 3 ranks 'S' (small-sized enterprise), 'M' (medium-sized enterprise) and 'L' (large-sized enterprise) according to the definition of enterprise by business sizes [3].

The global coverage of a customer i.e. the count of foreign currencies a customer has received invoices in so far, is also clustered into 3 ranks, 'S' (small coverage), 'M' (moderate coverage) and 'L' (large coverage)

Once we have all the metrics discretized into ranks, we use different combinations to profile the customer into 3 profiles:

- Steady Customers - Customers whose groups are the same across all metrics i.e Average Invoice Payable Amount per Enterprise Rank == Enterprise Size Rank == Global Coverage Rank
- Under-performing Customers - Customers with higher coverage but lower size rank and average invoice amounts i.e. Average Invoice Payable Amount per Enterprise Rank < Global Coverage Rank and Enterprise Size Rank < Global Coverage Rank
- Developed Customers - Customers with higher enterprise size rank and average invoice amounts compared to their global coverage

The profile of the customer keeps on getting updated as we receive invoice events over time.

3. Output

   The output is sent to a Kafka topic as a JSON object that contains PARTY_ID and the profile of the customer in key-value pairs. Depending on the stream, the topics are separate for customer profiles according to transfers, and invoices.

   Output object structure:

   {

   “party_id” : Customer_Party_Id,

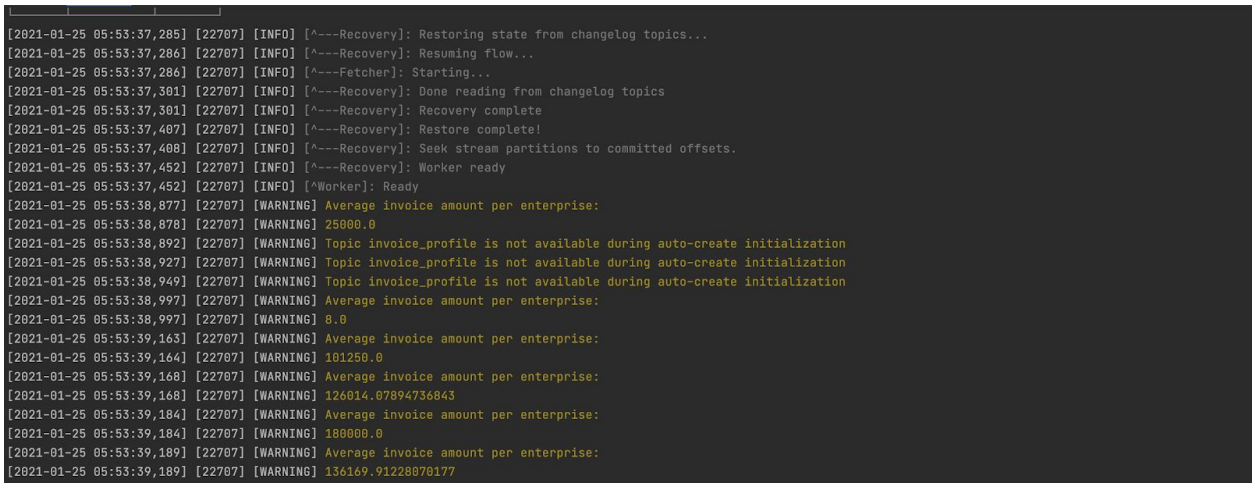
   “profile”: Profile_value

   }

Output Screenshots

Invoice History Processing

Output from Faust worker:



Profiling Output from Kafka:

```
shaikhsarmad@sarmad ~ % ~/Downloads/kafka_2.12-2.7.0/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic invoice_profile
"{\"party_id\": \"233824108\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"325170977\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"257500486\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"233823923\", \"profile\": \"Steady Customers\"}"
"{\"party_id\": \"233824108\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"349706937\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"233824108\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"334245713\", \"profile\": \"Steady Customers\"}"
"{\"party_id\": \"397530463\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"233823923\", \"profile\": \"Steady Customers\"}"
"{\"party_id\": \"353396518\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"345726751\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"334245713\", \"profile\": \"Steady Customers\"}"
"{\"party_id\": \"335008992\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"239904895\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"618230025\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"334245713\", \"profile\": \"Steady Customers\"}"
"{\"party_id\": \"334628087\", \"profile\": \"Developed Customers\"}"
"{\"party_id\": \"356882785\", \"profile\": \"Steady Customers\"}"
```

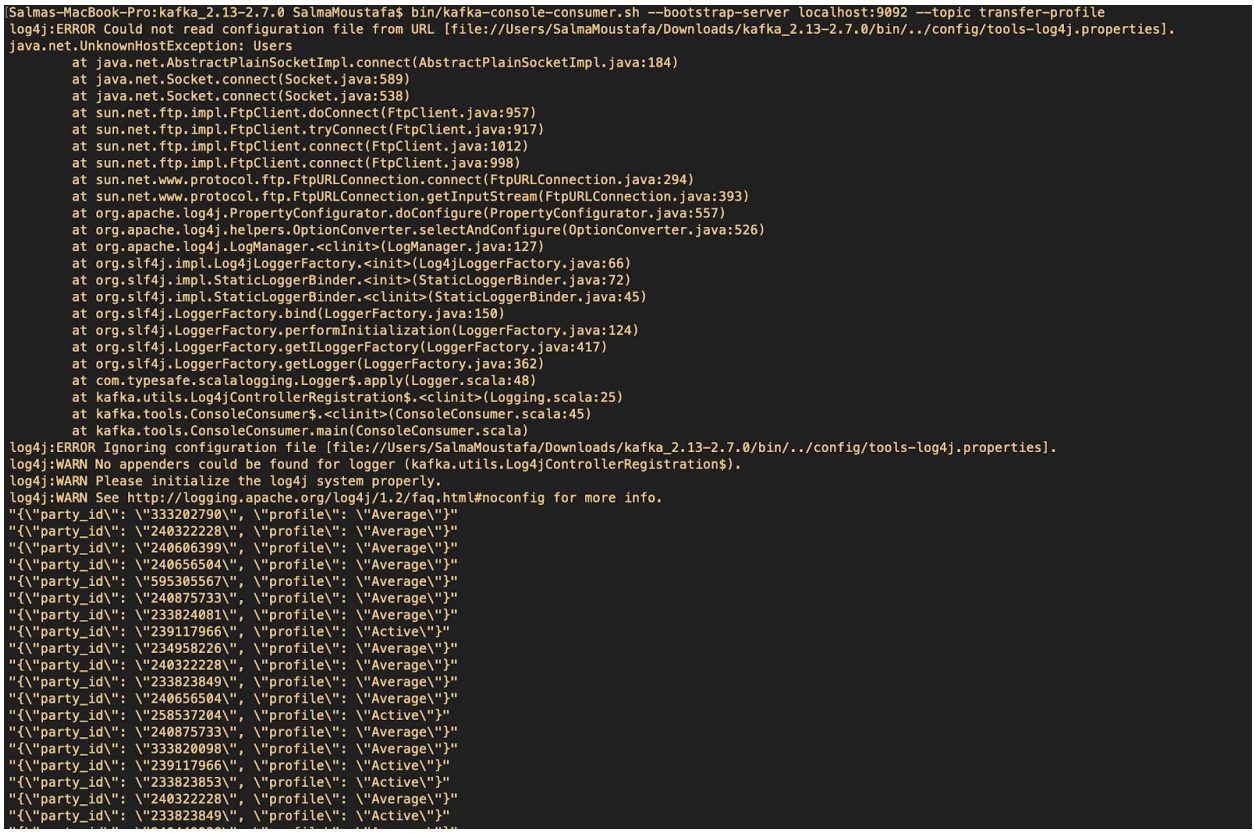
Transfer History Processing

Output from Faust worker:

```
[2021-01-27 18:12:26,353] [17114] [INFO] [^---Recovery]: Still fetching changelog topics for recovery, estimated time remaining 5.18 seconds (total remaining=21914):
Remaining for active recovery
+-----------------------------------------------------+-----------+-------------+-------------+-----------+
| topic                                               | partition | need offset | have offset | remaining |
+-----------------------------------------------------+-----------+-------------+-------------+-----------+
| customer_profiling_app-customer-profiling-changelog | 6         | 46420       | 42316       | 4104      |
| customer_profiling_app-rate-table-changelog         | 6         | 46419       | 36497       | 9922      |
| customer_profiling_app-sum-table-changelog          | 6         | 46419       | 38531       | 7888      |
+-----------------------------------------------------+-----------+-------------+-------------+-----------+

[2021-01-27 18:12:30,763] [17114] [INFO] [^---Recovery]: Done reading from changelog topics
[2021-01-27 18:12:30,763] [17114] [INFO] [^---Recovery]: Recovery complete
[2021-01-27 18:12:30,869] [17114] [INFO] [^---Recovery]: Restore complete!
[2021-01-27 18:12:30,872] [17114] [INFO] [^---Recovery]: Seek stream partitions to committed offsets.
[2021-01-27 18:12:30,934] [17114] [INFO] [^---Recovery]: Worker ready
[2021-01-27 18:12:30,934] [17114] [INFO] [^Worker]: Ready
[2021-01-27 18:12:32,134] [17114] [INFO] Fetch offset 2805 is out of range for partition TopicPartition(topic='transfers', partition=0), resetting offset
[2021-01-27 18:14:41,856] [17114] [INFO] NumExpr defaulting to 4 threads.
[2021-01-27 18:14:42,241] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,242] [17114] [WARNING] 3059000.0
[2021-01-27 18:14:42,347] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,347] [17114] [WARNING] 239436.50064627317
[2021-01-27 18:14:42,369] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,369] [17114] [WARNING] 291472.00783783785
[2021-01-27 18:14:42,378] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,378] [17114] [WARNING] 139295.74160322436
[2021-01-27 18:14:42,386] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,387] [17114] [WARNING] 524519.8139534884
[2021-01-27 18:14:42,397] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,397] [17114] [WARNING] 89974.00440379404
[2021-01-27 18:14:42,457] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,457] [17114] [WARNING] 239336.9392764858
[2021-01-27 18:14:42,571] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,572] [17114] [WARNING] 116734.74971064815
[2021-01-27 18:14:42,633] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,633] [17114] [WARNING] 830290.9393368501
[2021-01-27 18:14:42,730] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,731] [17114] [WARNING] 239243.18510546707
[2021-01-27 18:14:42,892] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:42,892] [17114] [WARNING] 162758.8433862434
[2021-01-27 18:14:43,039] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:43,039] [17114] [WARNING] 162652.82029598308
[2021-01-27 18:14:43,140] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:43,140] [17114] [WARNING] 162545.6747624076
[2021-01-27 18:14:43,185] [17114] [WARNING] Transfer Amount Rate per Customer:
[2021-01-27 18:14:43,185] [17114] [WARNING] 532241.7873100983
```

Profiling Output from Kafka:



```
Salmas-MacBook-Pro:kafka_2.13-2.7.0 SalmaMoustafa$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic transfer-profile
log4j:ERROR Could not read configuration file from URL [file://Users/SalmaMoustafa/Downloads/kafka_2.13-2.7.0/bin/../config/tools-log4j.properties].
java.net.UnknownHostException: Users
        at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:184)
        at java.net.Socket.connect(Socket.java:589)
        at java.net.Socket.connect(Socket.java:538)
        at sun.net.ftp.impl.FtpClient.doConnect(FtpClient.java:957)
        at sun.net.ftp.impl.FtpClient.tryConnect(FtpClient.java:917)
        at sun.net.ftp.impl.FtpClient.connect(FtpClient.java:1012)
        at sun.net.ftp.impl.FtpClient.connect(FtpClient.java:998)
        at sun.net.www.protocol.ftp.FtpURLConnection.connect(FtpURLConnection.java:294)
        at sun.net.www.protocol.ftp.FtpURLConnection.getInputStream(FtpURLConnection.java:393)
        at org.apache.log4j.PropertyConfigurator.doConfigure(PropertyConfigurator.java:557)
        at org.apache.log4j.helpers.OptionConverter.selectAndConfigure(OptionConverter.java:526)
        at org.apache.log4j.LogManager.<clinit>(LogManager.java:127)
        at org.slf4j.impl.Log4jLoggerFactory.<init>(Log4jLoggerFactory.java:66)
        at org.slf4j.impl.StaticLoggerBinder.<init>(StaticLoggerBinder.java:72)
        at org.slf4j.impl.StaticLoggerBinder.<clinit>(StaticLoggerBinder.java:45)
        at org.slf4j.LoggerFactory.bind(LoggerFactory.java:150)
        at org.slf4j.LoggerFactory.performInitialization(LoggerFactory.java:124)
        at org.slf4j.LoggerFactory.getILoggerFactory(LoggerFactory.java:417)
        at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:362)
        at com.typesafe.scalalogging.Logger$.apply(Logger.scala:48)
        at kafka.utils.Log4jControllerRegistration$.<clinit>(Logging.scala:25)
        at kafka.tools.ConsoleConsumer$.<clinit>(ConsoleConsumer.scala:45)
        at kafka.tools.ConsoleConsumer.main(ConsoleConsumer.scala)
log4j:ERROR Ignoring configuration file [file://Users/SalmaMoustafa/Downloads/kafka_2.13-2.7.0/bin/../config/tools-log4j.properties].
log4j:WARN No appenders could be found for logger (kafka.utils.Log4jControllerRegistration$).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
"{\"party_id\": \"333202790\", \"profile\": \"Average\"}"
"{\"party_id\": \"240322228\", \"profile\": \"Average\"}"
"{\"party_id\": \"240606399\", \"profile\": \"Average\"}"
"{\"party_id\": \"240656504\", \"profile\": \"Average\"}"
"{\"party_id\": \"595305567\", \"profile\": \"Average\"}"
"{\"party_id\": \"240875733\", \"profile\": \"Average\"}"
"{\"party_id\": \"233824081\", \"profile\": \"Average\"}"
"{\"party_id\": \"239117966\", \"profile\": \"Active\"}"
"{\"party_id\": \"234958226\", \"profile\": \"Average\"}"
"{\"party_id\": \"240322228\", \"profile\": \"Average\"}"
"{\"party_id\": \"233823849\", \"profile\": \"Average\"}"
"{\"party_id\": \"240656504\", \"profile\": \"Average\"}"
"{\"party_id\": \"258537204\", \"profile\": \"Active\"}"
"{\"party_id\": \"240875733\", \"profile\": \"Average\"}"
"{\"party_id\": \"333820098\", \"profile\": \"Average\"}"
"{\"party_id\": \"239117966\", \"profile\": \"Active\"}"
"{\"party_id\": \"233823853\", \"profile\": \"Active\"}"
"{\"party_id\": \"240322228\", \"profile\": \"Average\"}"
"{\"party_id\": \"233823849\", \"profile\": \"Active\"}"
```

Contributions

Sarmad Shaikh (I1I8G4)

- Contributed to Kafka consumer that does **invoice history processing** in stream using Faust and publishes the profiling output on Kafka
- Contributed to Kafka producer for publishing invoices data to corresponding topic
- Contributed to creating an executable shell script that sets up the environment and runs the invoice processing related code
- Contributed to managing Github branch on repository for the project

Salma Moustafa (G09BEI) - “Github Username: [smoustaf23]”

- Contributed to Kafka producer for publishing transfers data to corresponding topic (transfers-topic)

- Created Kafka consumer that does **transfer history processing** in streams using Faust (KafkaStreams python equivalent) and publishes profiling results to transfers-profiling.
- Contributed to creating an executable shell script that runs the code (for transfers)
- Integrated Forex-Python library for exchange rate conversions in invoices faust pipeline.
- Contributed to managing Github branch on repository for the project

Github Repository

https://github.com/TameemElshoumari/explainable-customer-profiling-in-financial-systems/tree/new_work_with_faust

References

[1] https://pypi.org/project/faust-streaming/

[2]https://towardsdatascience.com/3-libraries-you-should-know-to-master-apache-kafka-in-python-c95fdf8700f2

[3] https://data.oecd.org/entrepreneur/enterprises-by-business-size.htm