Data Stream Processing and Analytics

Mentor: Prof. Dr. Vasiliki Kalavri
Autor: Dimitar Dimitrov 18-940-080
Autor: Svilen Stefanov 18-950-907
Autor: Stevan Mihajlovic 18-936-815

Midterm Progress Report

29th April 2019

Your progress so far. Which tasks have you implemented?

So far, we have fully implemented Task #0 Data Preparation. The stream data is successfully stored in Kafka and the simulation of the out-of-order delivery, as well as, the speed-up factor code is completed. We do not anticipate any additional work needed for it at this time. We are currently in the process of implementing the code for Task #1 Active Posts Statistics. We have working Flink timestamps in place and we have just started implementing the time window infrastructure required to maintain the active post statistics. In parallel we are also working on Task #2 Recommendations, where we have managed to create a simple recommender system. Currently the system is solely based on static data, but we plan to extend it using stream data once Task #1's stream infrastructure is completed. We are yet to start with Task #3 Unusual Activity Detection.

Any divergence from your original plan. Did you change your mind about a tool you wanted to use or an algorithm?

Since the last report, a change in the overall structure of our team occurred in terms of Stevan joining our team. As such there were actually two Design documents submitted for our project - one from Stevan's team and one from Svilen and Dimitar's team. During our meetings we agreed to mostly follow the Svilen and Dimitar's design decisions. Taking this into consideration we are going to talk about changes with respect to that design only.

Originally, we planned to simulate the realistic data flow by doing the event delay and message reordering in the Kafka producer itself. We also planned on using Kafka for creating the timestamps used in Flink. In our current implementation we switched to doing the simulation and reordering in a Flink source instead. Additionally, the Flink source generates timestamps based on the maximum delay chosen for the simulation. Main benefit of the new design is that we no longer require to write to Kafka from scratch all the events for all the streams. Instead, the information is written to Kafka once and read multiple times by the Flink source. Additionally, the new timestamping allows for a more realistic streaming system which assumes a maximum waiting period after which messages are dropped.

Another change that we had to do is to our active post statistics design for Task #1. We originally assumed all of the comments are stamped with a PostID to signify which post the comments are part of. Instead, as we saw in the class for comment replies that field is left blank and there is only a ReplyToID supplied signifying the comment to which this reply is written. That requires a broadcasting scheme to be implemented so that a tree of replies is built at each active post partition of the data. We are still implementing the solution to that.

Challenges and issues faced so far and how you solved them or planning to solve them

We faced several challenges implementing Task #0. They arose from our decision to switch away from Kafka producer message reordering and instead to do it in Flink, as described above. We thought such a change will be trivial to implement. We were wrong and we ended up trying and abandoning several designs until we arrived at our current one. Our first attempt was to use KafkaConsumer as a Flink source for our stream and have streaming operators that manipulate the stream to delay randomly the messages. While we managed to implement the delay we failed to see how to properly generate watermarks in that context as the watermark generation in that context had to be decoupled from the reordering of the stream. We were also tripped by the ProcessFunction examples in the Flink documentation and their demonstration of the use of ValueState. While we eventually figured out how ValueStates work and understood they are not needed in our reordering implementation we lost a lot of time on that, as well. We eventually opted out for a KafkaConsumer wrapped inside a Flink source function that takes care of both the timestamps and the message reordering.

Outstanding tasks and a timeline for completion

As already mentioned, we have not yet started with the Task #3 Unusual Activity Detection. Due to an abrupt team change during the team formation phase, we are slightly behind schedule and are in the process of catching up.

We intend to have a working implementation of tasks #1 and #2 by the second week of May. The plan is to have 2 more weeks for task #3, which we expect to be the most difficult task, and for improvements and to still finish everything by 25th of May and then have another week to write the report. If we run into more challenges that would delay our progress, we might cut out some of the initially planned features.