Application Features

This application achieves the function of simple real-time chat platform. Users can chat with other users by their unique generated UUID. Also, the message content is censored, there are three words will be censored and replaced in “***”. Besides, the application implemented an additional feature, which is the message rate folder. The user cannot send with more than specific x messages in y timeframe length.

Additional Assumption

- Assume that all users have the stable and fast internet connection
- Kafka cluster and Flink cluster is setup properly, and have the enough resources to handle the message stream.
- Users know how to use the terminal, and have the basic knowledge of start and run the system.
- The cluster resources is limited, which means the cluster may cannot hold the big-flow of data in case there are spam message senders.

Configuration

Producer and Consumer

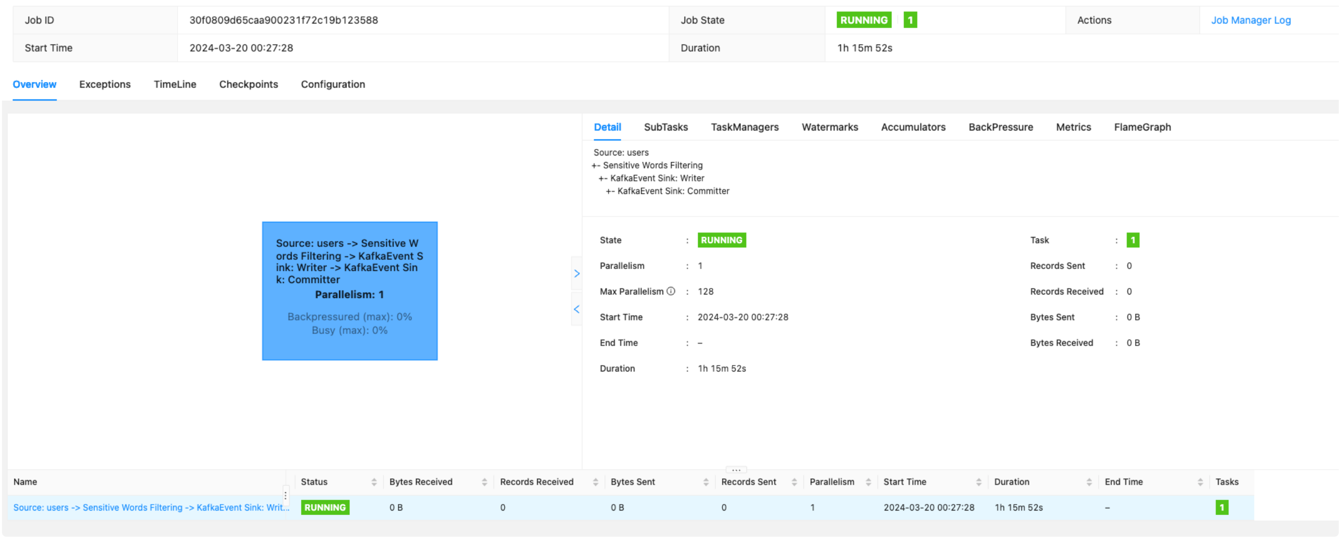
- The application uses `UUIDSearializer` and `UUIDDeserializer` to handle the UUID of users.
- The application use the `ChatMessageSerializer` and `ChatMessageDeserializer` to process the `ChatMessage` object to ensure the message is serialized and deserialized properly.

Topics

- Source topic: the source topic `users` is used to receive the raw chat message.
- Sink topic: the sink topic `message-adjusted` is used to store the adjusted message after censorship.

Apache Flink task manager

- The Flink task manager has scale of `1` , and the number of task slot is `2` . The configuration choose is based on the consideration of throughput and delay, also ensure the high effective and robustness of the system. The Flink can scale up painlessly, so it is more convenient to add more nodes.



Job ID	30f0809d65caa900231f72c19b123588	Job State	RUNNING 1	Actions	Job Manager Log
Start Time	2024-03-20 00:27:28	Duration	1h 15m 52s		

```
Source: users
+- Sensitive Words Filtering
  +- KafkaEvent Sink: Writer
    +- KafkaEvent Sink: Committer
```

State	:	RUNNING	Task	:	1
Parallelism	:	1	Records Sent	:	0
Max Parallelism	:	128	Records Received	:	0
Start Time	:	2024-03-20 00:27:28	Bytes Sent	:	0 B
End Time	:	–	Bytes Received	:	0 B
Duration	:	1h 15m 52s			

Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Duration	End Time	Tasks
Source: users -> Sensitive Words Filtering -> KafkaEvent Sink: Writ...	RUNNING	0 B	0	0 B	0	1	2024-03-20 00:27:28	1h 15m 52s	–	1

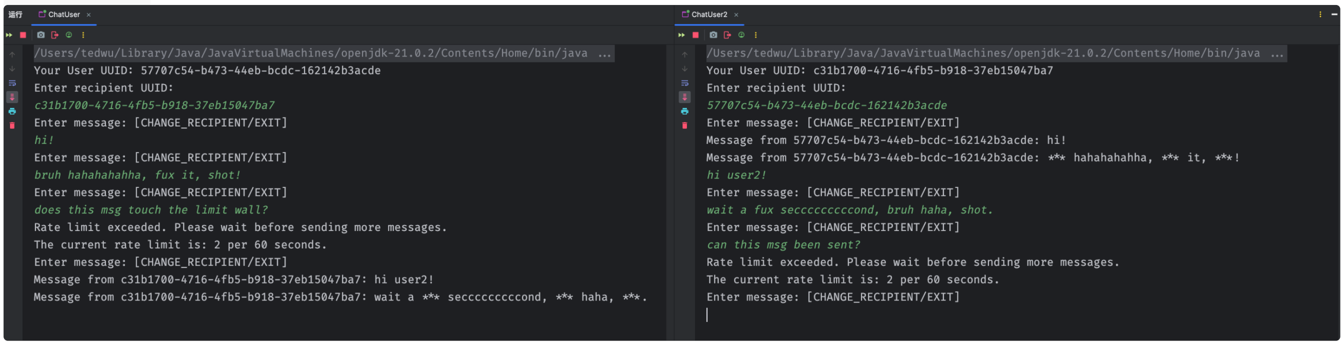
Build

1. Run Kafka Cluster, create topic with name `users` and `message-adjusted` . The bootstrap server is set default as `kafka0:9092`
2. Deploy Flink, map ports on 8081, and it is automatically connected to Kafka cluster.
3. Locate into sensitive message processing project folder, and use `mvn package` command to generate `.jar` file. After that, upload `.jar` file into Flink dashboard, finally start the job with `parallelism=1` .
    - In case of this application, the message go through the Flink and Kafka can be considered low volume, since we have the message rate limiter. The parallelism can be set bigger based on the current machine performance, Flink nodes, and the volume of the message. While it is worth to add more parallelism in case the user is increasing, and adding more Flink nodes.
4. Locate into Kafka chatting application folder, run two `ChatUser` program (`ChatUser.java` and `ChatUser2.java`)
5. Enter another user's UUID to terminal, start chatting.

Other information

- It is worth to mention that, the message rate limiter is set on the app-side, where the over-ratted message will not pass through the Kafka pipeline and processed by the Flink.
    - There is also a way to set this limiter in Flink side or somewhere else, while I think for this app's environment, is that the spam message should not go into the pipeline since it is a *ready-to-drop message* and the *cluster resources is limited* . Based on this assumption, I kill this message from the client side. This approach can reduce the stress of the cluster, especially in case that there are many spam messages.
- After run the `ChatUser.java` and start chatting, user can enter `CHANGE_RECIPIENT` to change the recipient ID.
    - User can also enter `EXIT` to quit the program.

- The sensitive word replacement is working, the default sensitive word list is: `{"bruh", "shot", "fux"}` , these word will appear as `***` if user sent message in the app. The user can change sensitive word list from the project's `Main.java` .
- The message rate limiter is working. The default message rate is set as 2 messages per minute (60000ms). The user can change the configuration on `ChatUser.java` by changing `RATE_LIMIT_WINDOW_MS=60000` and `MAX_MESSAGES_PER_SECOND=2`
- The detailed log is stored in Kafka cluster and Flink dashboard. User can enter `kafka-topics.sh` or Flink dashboard to check the log.



```
/Users/tedwu/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java ...
Your User UUID: 57707c54-b473-44eb-bcdc-162142b3acde
Enter recipient UUID:
c31b1700-4716-4fb5-b918-37eb15047ba7
Enter message: [CHANGE_RECIPIENT/EXIT]
hi!
Enter message: [CHANGE_RECIPIENT/EXIT]
bruh hahahahahha, fux it, shot!
Enter message: [CHANGE_RECIPIENT/EXIT]
does this msg touch the limit wall?
Rate limit exceeded. Please wait before sending more messages.
The current rate limit is: 2 per 60 seconds.
Enter message: [CHANGE_RECIPIENT/EXIT]
Message from c31b1700-4716-4fb5-b918-37eb15047ba7: hi user2!
Message from c31b1700-4716-4fb5-b918-37eb15047ba7: wait a *** secccccccccond, *** haha, ***.
```

```
/Users/tedwu/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java ...
Your User UUID: c31b1700-4716-4fb5-b918-37eb15047ba7
Enter recipient UUID:
57707c54-b473-44eb-bcdc-162142b3acde
Enter message: [CHANGE_RECIPIENT/EXIT]
Message from 57707c54-b473-44eb-bcdc-162142b3acde: hi!
Message from 57707c54-b473-44eb-bcdc-162142b3acde: *** hahahahahha, *** it, ***!
hi user2!
Enter message: [CHANGE_RECIPIENT/EXIT]
wait a fux secccccccccond, bruh haha, shot.
Enter message: [CHANGE_RECIPIENT/EXIT]
can this msg been sent?
Rate limit exceeded. Please wait before sending more messages.
The current rate limit is: 2 per 60 seconds.
Enter message: [CHANGE_RECIPIENT/EXIT]
```