Video Processing using Apache Flink

Converting normal colour video to Anaglyphed colour video

Project Objective

This project explores the conversion of Video Frames to a 3D Anaglyphed Video Frame through the integration of Apache Kafka and Apache Flink in a streaming data processing pipeline for video data.

We read video frames and load them to kafka. Using the Kafka frames as a source, we manipulate the stream of data, to transform each frame to a anaglyphed image, this operation is the processing workload that Flink performs.

Through the project, we aim to explore, understand and demonstrate the distributed system features provided by Flink - namely - Communication between Processes, Scalability and Fault Tolerance.

Technologies Used

The integration of Apache Kafka and Apache Flink is key to the success of our streaming data processing pipeline for video data.

Kafka is a distributed publish-subscribe messaging system, enables reliable and scalable data streaming between different components of the pipeline.

Flink is an open-source stream processing and batch processing framework, provides the necessary tools to process and analyze the streaming data in real-time. Flink applications are fault-tolerant, support exactly-once semantics, and can be written in Java, Scala, or Python, making it a versatile choice for our pipeline.

Technologies Used

Technology	Description
Apache Kafka	Distributed publish-subscribe messaging system for reliable and scalable data streaming.
Apache Flink	Open-source stream processing and batch processing framework for real-time data processing and analysis. Supports fault-tolerance and exactly-once semantics. Can be written in Java, Scala, or Python.

Implementation Overview

Video Processing System

The implementation consists of a video processing system that utilizes Apache Kafka and Apache Flink.

Apache Kafka

Apache Kafka is used to produce a stream of frames from a video.

The frames are read and produced to a Kafka topic, which serves as the input stream for Apache Flink.

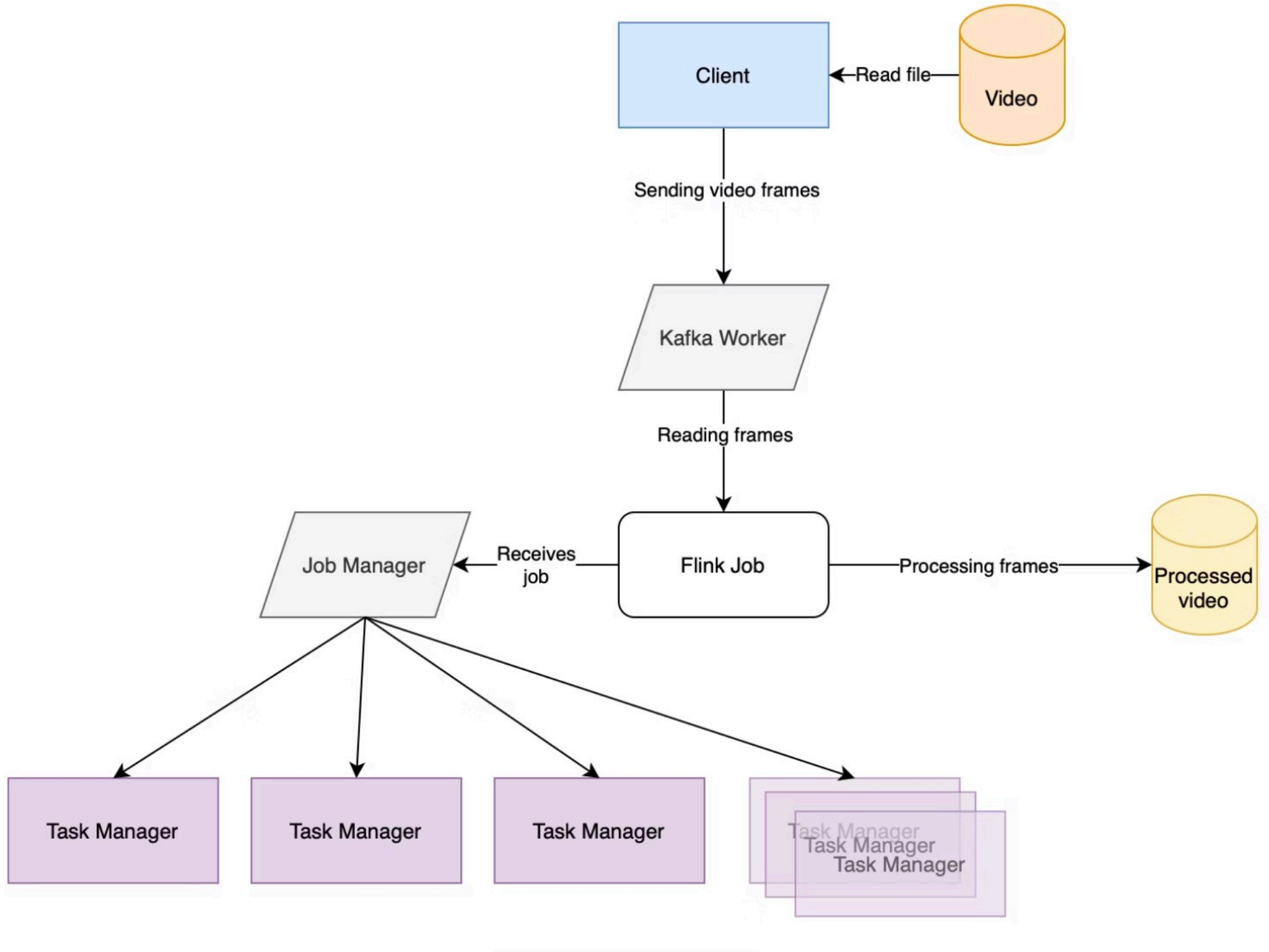
Apache Flink

Apache Flink is used to process the stream of frames from Kafka.

Flink performs an anaglyph transform on the frames and aggregates them.

The processed frames are then saved as a video.

Workflow

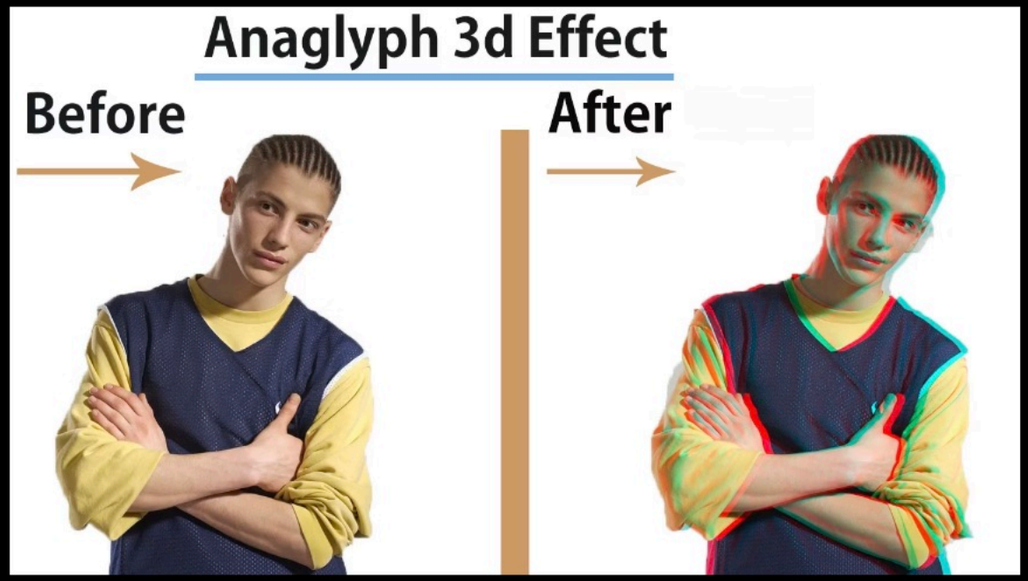

Client
Read file
Video
Sending video frames
Kafka Worker
Reading frames
Job Manager
Receives job
Flink Job
Processing frames
Processed video
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager

Anaglyph Processing

Anaglyph processing is a technique used to create a stereoscopic 3D effect by encoding the image each eye sees with filters of different colours, typically cyan and red.

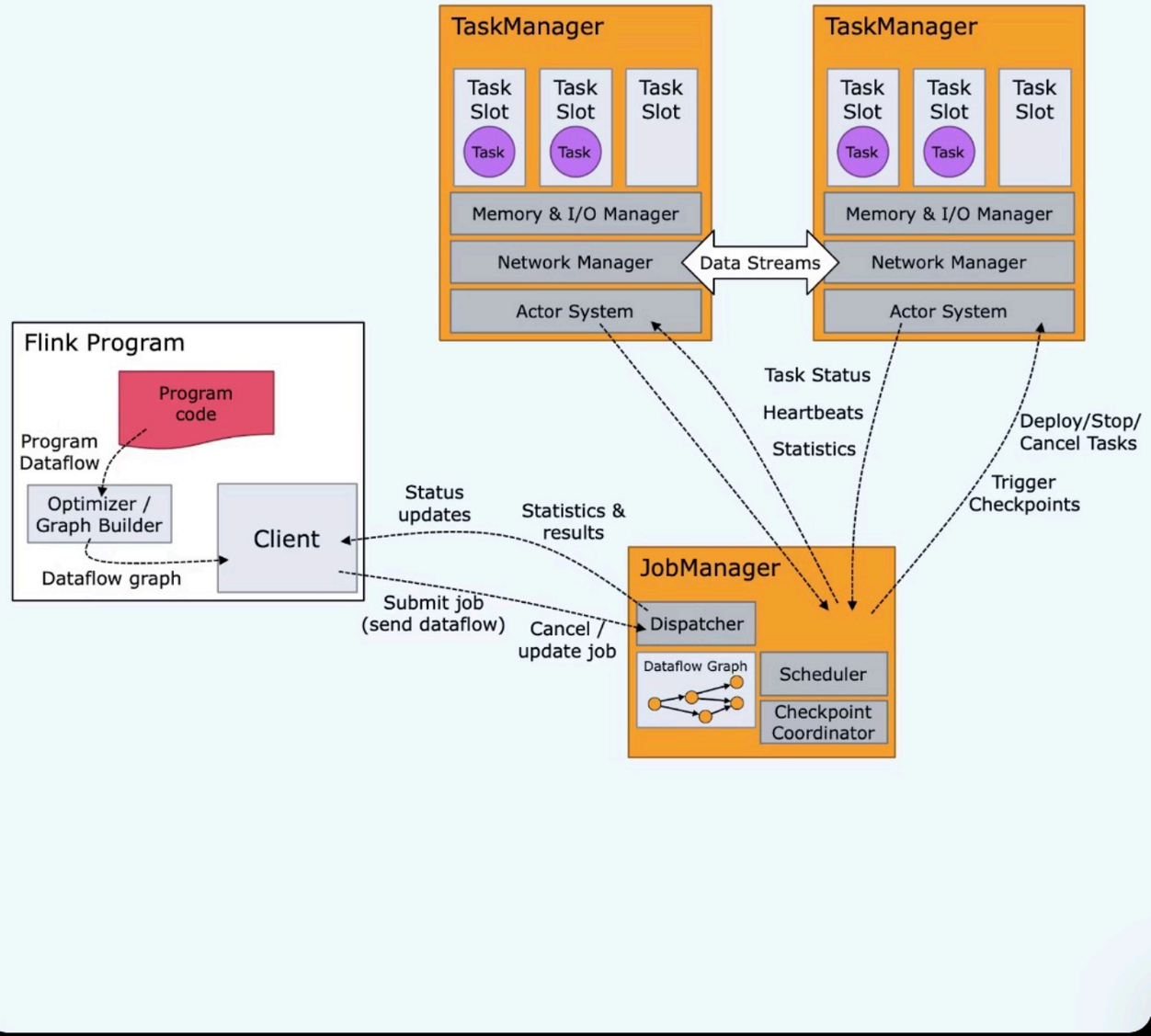
This affect is usually done with the help of a depth map or stereo images. But can also be mimicked from a mono image through horizontal offset.

Each pixel for the left and right view undergoes a mathematical transformation, in our use case, we decided to go with the populate COLOUR ANAGLYPH transform.



Apache Flink

- Distributed processing engine for Stream(Bounded and Unbounded streams) and Batch Processing
- Performs computations at in-memory speed, at any scale, boasting of event-time processing, exactly once state consistency, and scalability, with high-availability setups, low latency and high throughput
- Architecture mainly consists of JobManager and TaskManager
- JobManager distributes tasks over multiple TaskManagers
- High-availability setup has multiple job managers incase of job manager failure
- Can integrate well with kubernetes and yarn to provide reactive scaling



Discussions (Challenges)

- During the integration of Apache Kafka and Apache Flink in our streaming data processing pipeline for video data, we encountered several challenges.
- One major challenge was the unreliable documentation for Apache Flink. The documentation lacked clear and comprehensive explanations, making it difficult to troubleshoot issues and find solutions.
- Steep learning curve: Flink documentation is not very reliable for setting up the project environment. The documentation is either outdated or requires a lot of manual configurations which is undocumented.
- Limited support even for supported languages: Pyflink only provides limited support for certain functionalities which are possible in Java or Scala. But since image processing is best done using OpenCV, python is a better choice since OpenCV has limited support in Java. This led to the project being rewritten in the two languages since restricted documentation made it difficult to pick one over the other.Limited community support: Even though Flink has been around since 2010, it is not as widely used or discussed, especially for niche use cases. Which makes debugging extremely challenging owing to the poor documentation.

Conclusion

The integration of Apache Kafka and Apache Flink in a streaming data processing pipeline for video data has proven to be highly effective and efficient.

Key Findings:

- The combination of Kafka's distributed messaging system and Flink's powerful stream processing capabilities enables real-time analysis and insights on video data.
- The scalability and fault-tolerance of Kafka and Flink ensure the reliability and availability of the streaming data processing pipeline.
- The integration of Kafka and Flink allows for seamless data ingestion, processing, and delivery, facilitating real-time decision-making and action.

Outcomes:

- Improved efficiency and speed in processing and analyzing video data, leading to faster insights and actionable intelligence.
- Enhanced scalability and fault-tolerance, ensuring the reliability and availability of the streaming data processing pipeline even under high data volumes and demanding workloads.
- Real-time decision-making and action based on up-to-date insights from video data, enabling timely responses and optimizations.