AlgoLook

Uzmar Gómez

January 28, 2020

Architectures

The first reference we could find on the subject is in [1], a study on face recognition of pedestrians by analyzing live video streaming. They describe this process as computationally challenging, given that this video process involve different algorithms and steps, such as face detection, shadow removing, background substraction, feature extraction, and so on. They give the example of the face recognition algorithm, which can be executed very quickly and therefore is recommended to put it on more than one nodes in order to get the faster response. On the contrary, some heavy weight algorithms such as Gaussian mixture model for background substraction, need to be put into effect online if a system needs to process streaming video data. Therefore, in order to process video effectively and efficiently, the use of both batch processing capabilities and also real time in-memory processing capabilities is recommended. Apache Kafka deals the problem of live data transfer with high efficiency.

As for Spark, its a fast, generalised cluster-computing system. It offers an abstraction called "resilient distributed datasets" (RDDs) to support multi-pass applications that require low-latency data sharing across multiple parallel operations. RDDs can be stored in memory between queries without requiring replication. Instead, they rebuild lost data on failure using lineage: each RDD remembers how it was built from other datasets to rebuild itself.

They solve the problem of pedestrian face detection first by receiving the video stream data from a cluster of IP cameras with the video stream collector, which uses the OpenCV video-processing library to convert a video stream into frames. It then sends the messages to the Kafka broker using the KafkaProducer client. To process a huge amount of video stream data without loss, it is necessary to store the stream data in temporary storage, so this client works as a buffer queue for the data that the video stream collector produces. Keeping the data in storage before processing ensures its durability and improves the overall performance of the system as processors can process data at different times and at different speeds depending on the load.

The video stream processor is built on Apache Spark, which uses a Structured Streaming API to consume and process messages from Kafka, and uses the OpenCV library to process video stream data. A similar architecture is described on the following link.

In the article [2], the authors talk about the performance on the detection of intruders when using Apache Kafka and Spark Streaming. They mention that, when experimenting on the UNSWNB-15 dataset, this architecture has a good performance in terms of processing time and fault-tolerance on this huge amount of data.

Programming Languages

Although there is little research on the subject, there had been some studies that compare the performance when using OpenCV, either with Python or with C++.

For instance, on the article [3]

[4], [5] [6] [7]

References

[1] Abhinav Pandey and Harendra Singh. Face recognition of pedestrians from live video stream using apache spark streaming and Kafka. *International Journal of Innovative Technology and Exploring Engineering*, 7(5):4–10, 2018.

[2] May Thet Tun, Dim En Nyaung, and Myat Pwint Phyu. Performance Evaluation of Intrusion Detection Streaming Transactions Using Apache Kafka and Spark Streaming. *2019 International Conference on Advanced Information Technologies, ICAIT 2019*, pages 25–30, 2019.

[3] İlkbahar Fatih and Kara Resul. Performance Analysis of Face Recognition Algorithms. 2017.

[4] Alexandru E. Susu, Valeriu Codreanu, Georgios Evangelidis, and Lucian Petrica. Efficient implementation of a video change detection algorithm. *IEEE International Conference on Communications*, 2016-Augus:77–82, 2016.

[5] Weiyan Wang, Yunquan Zhang, Shengen Yan, Ying Zhang, and Haipeng Jia. Parallelization and performance optimization on face detection algorithm with OpenCL: A case study. *Tsinghua Science and Technology*, 17(3):287–295, 2012.

[6] Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Thomas Graves, Mark Holderbaugh, Zhuo Liu, Kyle Nusbaum, Kishorkumar Patil, Boyang Jerry Peng, and Paul Poulosky. Benchmarking streaming computation engines: Storm, flink and spark streaming. *Proceedings - 2016 IEEE 30th International Parallel and Distributed Processing Symposium, IPDPS 2016*, pages 1789–1792, 2016.

[7] Chenggang Yang, Yan Song, Jiang Qian, Hanying Zhao, Wei Jiang, Haibo Tang, and Jie Wu. Apache spark based urban load data analysis and forecasting technology research. *2017 IEEE Conference on Energy Internet and Energy System Integration, EI2 2017 - Proceedings*, 2018-Janua:1–6, 2017.