
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issues 3, pp. 479–496, DOI 10.12694:/scpe.v21i3.1764

PERFORMANCE ANALYSIS OF VIDEO ON-DEMAND AND LIVE VIDEO STREAMING USING CLOUD BASED SERVICES

UJASH PATEL*, SUDEEP TANWAR†, AND ANUJA NAIR‡

Abstract. The advent of Cyber-Physical Systems (CPS) has brought a revolutionary change coined as a mixture of information, communication, computation, and control. With applications in smart grid, health monitoring, automatic avionics, distributed robotics, etc., CPS is currently an area of attention among the academia and industry. The advancement of mobile communications and embedded technology has made it possible to build large scale CPS consisting of the interconnection of mobile phones. These devices collect information about the surrounding environment at any time anywhere basis through real-time video capture. Video streaming has proven to be a massive industry that is growing rapidly playing an important role in everyday life. Customer-driven approach wanting best experience with quality has to be the core offering of contemporary scenario. Video streaming is categorized into Video-On-Demand Streaming (VoDS) and Live Video Streaming (LVS) showing the current state-of-art opportunities. Many diverse applications of video streaming are military video surveillance using drones, live sports match player face recognition, on-demand video characters recognition, movie summarization like identifying parts of the movie which are viewed many times by different users, movie and series recognition, motion detection, gesture recognition, image segmentation, etc. This paper introduces an approach to develop video analysis on VoD and LVS using cloud-based services and analyzes the impact of Quality of Experience (QoE), cost, and bandwidth on the cloud. To achieve the best user experience for video streaming and video analysis, Content Delivery Network (CDN) offers the best QoE at various analyzed locations using various cloud providers like Amazon Web Services (AWS) CloudFront, Google Cloud CDN, Azure CDN, Akamai CDN, etc.

Key words: Cyber-Physical Systems, Live video streaming, Video on-demand streaming, Kafka, RabbitMQ, OpenCV, Spark, AWS Services

AMS subject classifications. 68M11

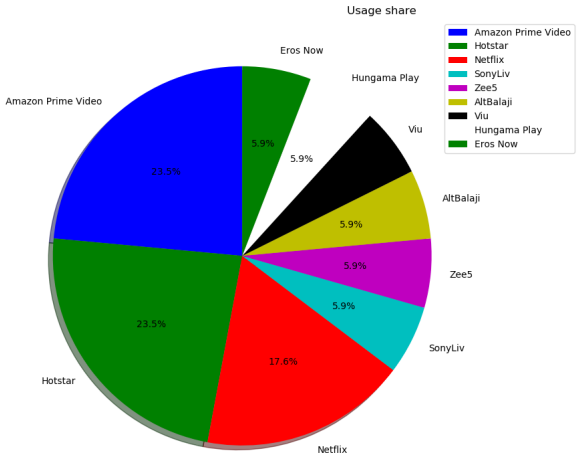
1. Introduction.

Cyber-Physical Systems (CPS) [1] has arisen as a novel generation of engineered systems seamlessly assimilating the capability of computing, control, communication, and information with physical systems. Information sharing, organization & coordination among physical systems, human beings, and the Internet are often required by CPS. The advancement of mobile communications and embedded technology has made it possible to build large scale CPS consisting of the interconnection of mobile phones. These devices collect information about the surrounding environment at any time anywhere basis through real-time video capture. Sensing applications that are video-based equipped with camera-based mobile phones required to capture, send, and receive real-time videos for CPS is envisioned. Internet is playing a major role in the communication of the captured videos and live or broadcasted videos from mobile phones about the events taking place in the proximity of a user in real-time. In comparison with applications on mobile devices like for downloading and viewing videos, several challenges are faced by video-based CPS. These challenges include abundant broadband network access, relevant content delivery as per the substantial need of the hour, secure communication, unnecessary downloading of software required for inspecting these videos, etc. The adoption of VoD & LVS has advanced implementation methods to overcome these challenges. Video content providers like YouTube, Netflix, Hotstar, Facebook, Instagram, Snapchat, TikTok boasting 300 hours of video content uploaded to their space every minute have adopted different services to achieve the best user experience with

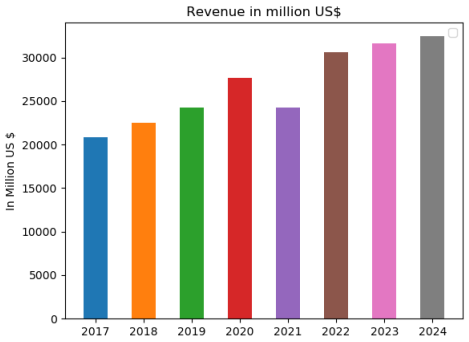
*Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad, Gujarat, India 382481 (17bce161@nirmauni.ac.in).

†Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad, Gujarat, India 382481 (sudeep.tanwar@nirmauni.ac.in).

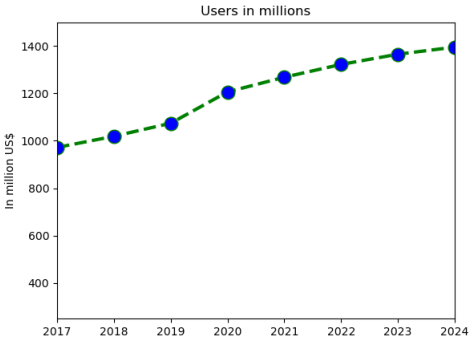
‡Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad, Gujarat, India 382481 (anuja.nair@nirmauni.ac.in).



(a) Video streaming application usage share in India



(b) Revenue in a million US$ world wide



(c) Users in millions world wide

FIG. 1.1. *Video streaming market analysts reports with detailed comparisons [2]*

video delivery. LVS refers to having no time delay, ability to chat, ask, and respond to questions, etc. For example, live cricket match streaming like Hotstar streaming match where each frame serves to users immediately after it is captured.

On-demand streaming refers to high quality (HD) video and audio playing smoothly at any Internet speed where the video is already framed at different edge locations of the cache server with different resolutions. Considering the current scenario, due to the COVID-19 virus, there will be a direct impact on the video streaming market because almost all educational universities are moving towards online lecture video streaming. In the entertainment industry also, streaming services revenue will grow rapidly as shown in Figure 1.1a [2]. Figure 1.1b [2] indicates how revenue will increase in 2020 as total segment amount will be US$ 27,628M. Along with revenue increase, the number of users will also increase as shown in Figure 1.1c [2] where the number of users is expected to be 1,395.7M by 2024. The services like bandwidth analysis, capturing, uploading a video, encode frame, analysis video frame, decode the frame, convert the frame into different resolutions, distributed storage, user analysis data, and most important continuous delivery of frame in sequence frames to the end-user are required. All of that can be achieved by dividing services in microservices. In this paper, we will discuss how to capture a frame from a large scale user and analyze each frame and serve them to end-user using cloud-oriented services.

1.1. Need of Video Analysis. Video analysis has numerous advantages in today's streaming applications. Video analysis is used to analyze objects in a continuous frame using machine learning and deep learning algorithms. For example, in cricket LVS, we can identify player face reactions or predict ball direction, etc. To achieve the best video analysis system, we need to take care of how fast we can stream data to multiple nodes. Multiple nodes are required because video analysis needs a lot of computational power to analyze which is not reasonable using single thread or single CPU power. Instead, distributed computational power is the need of the hour. That would be a reason why we need Kafka as messaging services. An open-source stream-processing software platform written in Scala and Java coined as Apache Kafka provides high throughput, unified, and low latency platforms for real-time video feeds.

Till now, many surveys, studies, and methodologies are conducted by several authors for VoDS & LVS. For example, Panda *et al.* [3] analyzed capturing video frames from different sources like IP cameras, web cameras, mobile cameras performed using wireless ad hoc networks. Ichinose *et al.* [4] investigates how video analysis can be done using Kafka and Spark. The video frame was converted into the JSON object and the paper defined different configurations for the Kafka producer side and consumer side. Bouge *et al.* [5] analyzed how Kafka is helpful in big data analysis and Liu *et al.* [6] worked on the Internet of Things (IoT) devices for Kafka which helps in analysis video. Big data needs a lot of computational power to analyze fast, at the same time, video and image should not drop their quality. Hence, all Kafka and Spark configuration must be done at the client-side as well as the server-side. Ma *et al.* [7] showed performance evaluation for video and image services provided by different cloud service providers. Huang *et al.* [8] used NGINX for load balancing and Red5 media servers or Amazon web service (AWS) services like media elemental services. Red5 is an open-source media server that is designed in Java and provides different services like Wowza Streaming Engine, Adobe Flash Media, AWS Streaming and Wowza Streaming Support, HTTP Live Streaming (HLS), Flash, WebSockets, and RTSP so that video frames gets delivered in various devices. In mobile devices, there are several parameters like error handling, frame loss, and buffering which was handled by Muthuswamy *et al.* [9] using sliding window protocol and standard H.264 encoding technique.

To send a frame to the user's mobile, first, the server needs to know about user bandwidth, accordingly, the server sends relevant frame resolution frames to end-user mobile. Users can send requests from anywhere worldwide and hence, that frame requires time to reach an endpoint. In the VoDS, a delay in the frame is accepted but LVS requires each frame to be delivered to the user without any delay. Kim *et al.* [10] penned down that based on the usage of different CDNs, user experience i.e. QoE and video streaming cost may vary. All services for LVS & VoDS are on the cloud and hence, there are various techniques and user behaviors to decrease cost and increase QoE. Lee *et al.* [11] investigates video streaming based on different factors like user age, user gender, user watch timing, etc.

1.2. Motivation. CPS is a combination of systems with diverse nature with the purpose to control a physical process. It takes feedback and adapts to new conditions based on real-time input. It is a combination of physical processes, networking, and computation. The advancement of mobile communications and embedded technology has made it possible to build large scale CPS consisting of the interconnection of mobile phones. Sensing applications that are video-based equipped with camera-based mobile phones required to capture, send, and receive real-time videos for CPS is envisioned. Video-based CPS also faces a lot of challenges like abundant broadband network access, relevant content delivery, etc. Streaming of video is possible through either LVS or VoDS. Analyzing a video is required in CPS because if the system needs to be changed as per conditions being met in real-time, what is happening in the video after analyzing it. Since we need real-time services, video analysis should be very fast and Apache Kafka messaging services are need of the hour. It provides high throughput, unified, and low latency platforms for real-time video feeds. Hence, this would be beneficial for CPS.

1.3. Contributions. Through this paper, we present a review of video analysis over the LVS & VoDS video application. We compared different messaging brokers which help to deliver each frame in a distributed pipeline. Our primary focus is to analyze the impact on two message brokers for video analysis and how we can achieve LVS & VoDS using AWS elemental services. In this paper, we also analyzed the Kafka configuration parameter for reliability on full-service-Mode. Based on the above discussion, the following are the significant contributions of this paper:

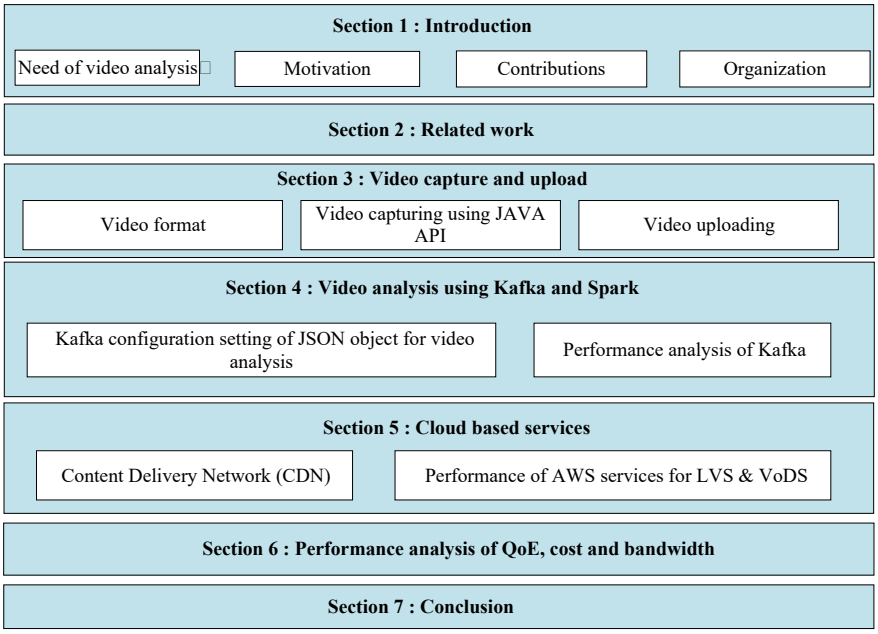


FIG. 1.2. *Organization of the paper.*

- We present a scheme that helps to analyze video from the LVS & VoDS. It obtains how the frame can be divided into multiple OpenCV MAT objects and video analysis through Apache Kafka streaming and Spark.
- Video frame stored in the cloud that gets delivered to the end-user by using a content delivery network.
- The proposed scheme also finds out how to increase QoE and manage cost as well as bandwidth.
- The proposed scheme uses different cloud vendors or implementation like AWS, Google Cloud, Azure, and analyze which cloud vendor is best suited for what services and in which region.

1.4. Organization. The structure of the survey is as shown in Fig. 1.2. Table 1.1 lists all abbreviations used in the paper. The rest of the paper is organized as follows. Section 1 gives an introduction to CPS and its connection with video analysis. Section 2 examines related works conducted by several authors for LVS & VoDS. Section 3 presents insight into video capturing and uploading. Section 4 presents the video analysis using Kafka and Spark. Section 5 describes the implementation of different cloud services and corresponding results. Section 6 presents performance analysis concerning QoE, cost, and bandwidth.

2. Related Work. Several surveys and research work has been done in the area of LVS & VoDS. Ichinose *et al.* [4] analyzed real-time video data from multiple cameras and analyzed them using streaming engines and different machine learning and deep learning libraries. The authors used Kafka for streaming and Spark for streaming and analyze video frames. Real-time Transfer Control Protocol (RTCP) can be used to unicast and multicast which is described by Wang *et al.* [16]. The system is divided into two modules, namely, collection system and play system. Video data collection, H.264 encoding, RTP package, RTCP control, cache and preview, and real-time transmission is a part of the collection system. On the other side, the play system consists of data receiving, H.264 decoding, and RTP decoding. The authors used socket technology caching after video capture. The proposed system has strength over bandwidth for smooth video streaming for a stable environment whereas, in case of the fluctuating environment, the system can not predict actual bandwidth of end-user for smooth video streaming. Patel *et al.* [18] introduced how smooth video streaming can be achieved with fluctuating bandwidth. First, Available Bandwidth (ABW) is analyzed, then the average of that ABW is taken and at the end, according to the average ABW, streaming engine sends particular video frame resolutions.

Eduardo *et al.* [12] proposed a distributed architecture for LVS & VoDS which helps to stream large

Table 1.1
List of abbreviations

Abbreviation	**Description**
CPS	Cyber-Physical Systems
VoDS	Video On-Demand Streaming
LVS	Live Video Streaming
QoE	Quality of Experience
QoS	Quality of Service
CDN	Content Delivery Network
AWS	Amazon Web Services
HD	High quality
IoT	Internet of Things
HLS	HTTP Live Streaming
LUT	Look-up-table
AVC	H.264 Advanced Video Coding
HEVC	H.265 High Efficiency Video Coding
MP3	MPEG Audio Layer-3
AAC	Advanced Audio Coding
D-VoD	Distributed video on-demand
JD-VoD	Java distributed video on-demand
AC-3	Audio Coding 3
CDC	Change data capture
RTCP	Real-time Transfer Control Protocol
ABW	Available Bandwidth
AMQP	Advanced Message Queuing Protocol
IaaS	Infrastructure as a Service
ARQ	Automatic Repeat Request
FEC	Forward Error Correction
EC2	AWS Elastic Compute Cloud
S3	AWS Simple Storage Service
HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Dataset
SSM	Spring SpringMVC Mybatis
MSK	Amazon Manages Streaming Kafka
RTMP	Real-time messaging protocol
AAC-LC	Advanced Audio Aoding - Low Complexity
HE-ACC	High Efficiency Advanced Audio Coding
RTT	Round Trip Time
RDBMS	Relational Database Management System
U_{rl}	Camera URL
C_{obj}	Camera object created by OpenCV
M_{rw}	MAT object number of rows
M_{cl}	MAT object number of columns
M_{ty}	MAT object type
J_{obj}	JSON object
T	Topic name
P_r	Kafka producer object
S_c	JSON schema structure for Spark
D_s	Create dataset from stream messages from Kafka
P_{data}	Process data
S_t	Group states
E_x	Existing states
P_p	Event processed data
E_{data}	MAT data from JSON
E_{key}	Camera key from JSON

videos on small networks also. In the video frame, duplicate frames could also be transferred. To overcome this problem, Wu *et al.* [21] used different delivery semantic and compared with other messaging systems like RabbitMQ, Advanced Message Queuing Protocol (AMQP), and Apache Flink. Apache Kafka works in a distributed environment and there are many cloud vendors available that provide Kafka as Infrastructure as a Service (IaaS). Wu *et al.* [22] compares Kafka performance on different cloud vendors. In Spark, data

TABLE 2.1
A relative comparison of state-of-the-art video analysis approaches

Author	Approach	Objective	Year	Aplication	Pros	Cons
Batista *et al.* [12]	D-VoD server and JD-VoD server	Distributed architecture for LVS & VoDS distribution	2005	Distributed architecture for LVS & VoDS distribution	Processing of large videos on small networks	Not scalable.
Yi *et al.* [13]	Look-up Table (LUT) technique.	Decoding technique by means of converting input video stream into bit stream.	2015	Format stream in Little-Endian Systems	3.49% faster decoding and 11.45% reduction in bit extraction time	Can only be applied to H.264 video application
Machida *et al.* [14]	Socio-ICT model	Timely detect abnormal events and swiftly deliver alerts to security agencies	2015	Resilient video surveillance service for the purpose of ensuring safety	System is more resilient	Costly and requires accurate gadget
Li *et al.* [15]	AWS services for transcoding	Transcode same video into different formats and decrease QoS violation of video streams	2016	Deliver multiple output videos via progressive download or adaptive bit rate	Major computational part resides at cloud and cost-efficient and QoS	Does not work on LVS
Wang *et al.* [16]	Mobile captured, media streaming engine	LVS system that is mobile based on streaming media technology	2016	Used for smaller sized organization	High transmission efficiency	Socket technology used for capturing and caching
Su *et al.* [17]	Femtocaching, VoDS, QoE.	Wireless traffic demand driven by VoD streaming and LVS	2017	Single user architecture	Playback duratio affected	Need a large scale edge location
Patel *et al.* [18]	Available bandwidth prediction, adaptive video streaming	Video streaming over fluctuating bandwidth	2017	Can be suitable for VoD	Good for wireless networks	Very complicated to decide which packet belongs to which base at client side
Kanrar *et al.* [19]	Viewer driven session based multi-user model	Bandwidth utilization	2017	Peer-to-peer network where bandwidth is consumed at every session	Services like pause, move slow, rewind, skip some number of frames	Works only on mesh networks
Bhole *et al.* [20]	Stream data, Apache Kafka, Cryptocurrency	Big data streaming for Cryptocurrency	2018	Tracking of cryptocurrency	Web based tracking modules	Locally tested
Liu *et al.* [6]	Spring SpringMVC Mybatis (SSM) + KAFKA	IoT middleware message service	2019	Used in IoT tracking devices	High throughput, distributed, fault tolerant	Lack of pace, issue with message tweaking
Wu *et al.* [21]	Streaming processing, Apache Kafka	Impact of all kinds of configuration parameter on the reliability of Kafka	2019	One-to -many communication streaming	Management on duplicate message control	Cluster-Service mode
Wu *et al.* [22]	Input configuration parameter	Queueing based packet flow model to predict performance metrics of Kafka	2019	D-stream data configuration parameter	Performance-based configuration	Configuration only for cloud vendors not for cluster-service mode
Tu *et al.* [23]	Map reducer, Spark streaming, big health data	Medical streaming data architecture for big data	2019	Hospital data analysis	Real-time data processing for daily health	MySQL vertical scaling or scale-up
Uddin*et al.* [24]	MMLSpark, OpenCV, HDFS	Distributed video analytic for intelligent video surveillance	2019	Several offline and online surveillance videos	Ensures scalability and fault tolerance	Security and privacy of cloud is not addressed
Mahapatra *et al.* [25]	Spark pipelines, IoT mashap tools, graphical tools, and streaming analysis flow base	Reduce complexity for pipeline data stream	2020	Graphics sorting	Spark pipeline D-stream flow	Re-usable model for persisting external file system
Shabrina *et al.* [26]	Using different cloud vendors	Improving of QoS	2020	Can be used at different edge locations with different cloud vendors	Throughput averaged to 3990.4 KB/S	Geo location may not be close to all target locations, adds complexity to your website for deployment procedures
Dongen *et al.* [27]	Apache Spark, structured streaming, Apache Flink, Apache Kafka, Kafka streams, distributed computing	Single pipeline for data streaming	2020	Large user scale	High throughput	Single Kafka broker
Anveshrithaa *et al.* [28]	Apache Spark, Long Short-Term Memory, Kafka	To predict traffic flow information	2020	Real-Time Vehicle Traffic Analysis	Reduces travel time, energy and cost	Hyperparameter optimization not been carried out

comes from across different platforms like MySQL, Kafka, Flink, RabbitMQ, etc. Tu *et al.* [20] analyzed how Spark architecture will perform with varying input stream data, for example, they have used health monitoring data for analyzing each input stream engine. Yi *et al.* [13] presented a decoding method for compressed video stream which is 3.59% faster than conventional method. Shabrina *et al.* [26] had used AWS service to analyze Quality of Service (QoS) which was proven cost-efficient for HLS. To improve the QoS of a video stream, CDN is the main factor for analysis. The authors in [26] analyzed the mechanism of HLS streaming related to CDN implementation. Su *et al.* [17] investigated the use of caching policy for VoD service demand. Caching policy helps in playback duration but it also requires a large scale edge location for world wild edge location. Kanrar *et al.* [19] presented a session-based multi-user model for bandwidth utilization that helps in a peer-to-peer network where bandwidth is required at any session but it works on only mesh network.

Liu *et al.* [6] proposed Kafka role in IoT based application. IoT services interconnect through many telecommunication networks and the Internet which connects IoT services point to point. There are many to many and one to one relationships between IoT services and hence, middleware plays a major role here which helps to send data to different services according to their relationship. The middleware services must be efficient and reliable message services and hence, Kafka is used as a middleware as a message broker. All IoT services are tested in multiple nodes Kafka. The Spring MVC has been used for Kafka middleware as Spring cloud provides microservices which makes each middleware service as independent service, so that, flexibility using different technology and scalability can be leveraged. Bouge *et al.* [5] analyzed Kafka in different computational parameters for big data streams. They prescribed four phases namely data collection, ingestion, processing, and storage. The main contribution of this paper was that ingestion performance can impact the overall stream processing. Video analysis is a major factor when it comes to smart city surveillance systems. Machida *et al.* [14] introduces a socio-ICT model which consists of a dynamic and queuing model. The platform enables dynamic load changes offloading video analysis. Social simulation to analyze the causal relationship between different parameters like arrival rate increases while the exit rate remains unchanged and the congestion level increases due to increased face drop rate. Concerning one video frame to be served in different resolutions, Li *et al.* [15] introduced transcoding for LVS & VoDS. The authors stored numerous versions of the same video in advance called pre-transcoding. Transcoding services also factorize for cost-efficient video transcoding using heterogeneous cloud services. Table 2.1 shows a comparison between existing approaches in video analysis techniques.

3. Video capture and upload. This section explores various video formats. We have considered the heterogeneity/complexity of different codec techniques used to perform video analysis. Table — depicts the comparative study of such techniques along with their pros and cons. Also, video capturing using JAVA API and video uploading using OpenCV is discussed in brief.

3.1. Video format. A video format consists of two parts. First is the format and codec being the second one. A format is a standard set of rules for storing containers, codecs, metadata, and folder structure. In the market, different format exists like MP4, AVI, WMV, etc. Each different format factor is arrangements of information within a container like a video stream, audio stream, metadata (bit rate, device resolution, subtitle, time of creation). Codec, which is part of video metadata, is a combination of CODER and DECODER. It encodes the video and audio stream making it smaller and easier to manage. At the end-user, the device decodes the video using a video player which also suggests codec. There are several video codec available in the market but mainly there are 3 codecs that almost all devices support like H.264 Advanced Video Coding (AVC), H.265 High-Efficiency Video Coding(HEVC) and VP9. The main audio codec is MPEG Audio Layer-3 (MP3), Advanced Audio Coding (AAC), and Audio Coding 3 (AC-3).

3.2. Video capturing using JAVA API. According to Panda *et al.* [3] Java API helps to capture videos from different sources like IP camera, WebCam, mobile video camera, etc. The API helps in encoding and compressing video and later using a multipart (It is a container that holds the whole body part of the capture video frame) streaming technique to transport the video to the Kafka streaming engine. The whole operation is divided into independent microservices.

Ad-hoc networks are used for capturing large scale video from different users. While the user capturing a video, the wireless link gets repeatedly broken and re-established again and again resulting in the loss of some

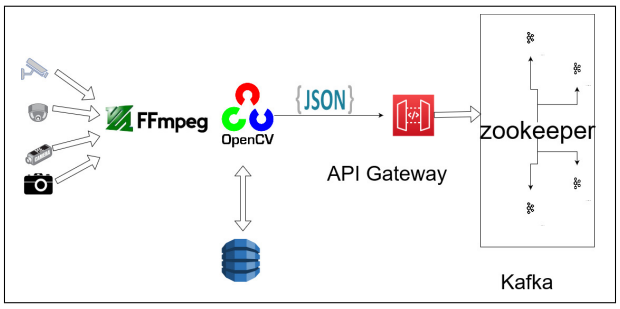


FIG. 3.1. *Video Capturing using OpenCV*

of the frames. The authors have used Automatic Repeat Request (ARQ) and Forward Error Correction (FEC) which is modified to reduce high transmission error. Sender side video is encoded by Java's encoder which is done at the client-side. The streaming is transferred to a particular route by different routing protocols. At the receiver side, the video streams are put in sequence after the video is decoded for different platforms. But, if we have a mobile device or any browser for display, it requires a lot of computational power for decoding, and hence, decoding work is done at the cloud. AWS Elastic Compute Cloud (EC2) helps to create a server that takes an encoded frame. AWS cloud helps us to decode the frame into different formats. Many different cloud services will help to compress the video to reduce the size of the video.

3.3. Video uploading. Video uploading tasks are done by the user or video streaming provider. For example, on Facebook, user uploads video whereas in Netflix, admin uploads video, and different subscribed users are endpoints, hence, it is termed as VoDS. The Video stream collector works with different user's video uploading at large scale OpenCV video processing libraries to help convert a video stream into frames. OpenCV stores all frames and images into MAT objects that the object is serialized to and can transfer to Kafka producer to deliver it to different consumers for Spark machine learning library for video analysis. The MAT object is transferred in JSON format with a user ID and camera ID as shown in Figure 3.1. After Spark operation is performed, all analyzed video frames are transferred to secure storage like AWS Simple Storage Service (S3) or Hadoop Distributed File System (HDFS).

4. Video Analysis using Kafka and Spark. It is very difficult to perform large-scale real-time data processing in the cloud because of the computational complexity while performing data analytics. Data within a distributed network transfers at a very fast rate. For video analysis, we have selected Kafka because it provides a fast, fault-tolerant, scalable messaging system and is often used in real-time data streaming architectures providing real-time analytics. Following are the merit reasons for using Kafka to perform the video analysis:

- Distributed environment with publisher and subscriber architecture.
- Provide up to seven-day storage facility.
- Receive messages in order with different partitions with the same topic (support strict order).
- Support multiple thread nodes for multiple consumer data deliveries.
- Commit log for retrieve data after a failure.
- High throughput.
- Massive scale data support.
- Wider use case of event-driven microservices, log store, streaming, event sourcing, change data capture (CDC), enterprise data pipeline.

Ichinose *et al.* [4] proposed a framework that measures the data analysis throughput which varies depending on the number of brokers and the number of nodes for consumers. Here, video analysis needs continuous transfer of large amounts of data for computation in the cloud. There are many deep learning frameworks like Cafe, TensorFlow, Chainer, etc. A sequential process is followed wherein the client receives data from Kafka with Spark, executes its machine learning library and python program, and later Chainer is called upon. Data is stored in the form of a Resilient Distributed Dataset (RDD) in Spark. Kafka configuration is divided into two subsections, first being single node and second being multiple node configuration as stated below:

- *Single Node:* In this, one thread is assigned to one customer. The number of machine cores used is 8 cores setting several consumers. Thus, the number of machine cores enables efficient processing.
- *Multiple Node:* Configuration of 1 to 2 consumer nodes means that throughput increases by 1.8 times but configuration of 2 to 4 consumer nodes, the improvement is by 1.4 times.

The requirements of Kafka's reliability relies on different use cases. In the case of video streaming, no duplicate messages should be delivered to the endpoint. For this use case, Kafka provides three delivery semantics. The first approach is at-most-once semantics where the producer sends messages continuously without waiting for any acknowledgment from consumers. Thus, in this case, the message delivered might be duplicate or message may be lost. The second semantics is at-least-once where the producer waits until an acknowledgment from the servers. Sometimes this approach might lead to duplicated messages. The third approach is exactly-once semantics which guarantees where all messages will be delivered without duplication. For our approach, we have used exactly-once semantics.

In this paper, Spark SQL is used for grouping the same camera ID frame for analysis. Algorithm 1 shows steps of video capturing and converting frames into JSON objects. In the algorithm, an input is a Camera URL and output is a JSON object. Prerequisite is one must have configured parameters for Kafka producer which is mentioned below and the external OpenCV library must be added in the Java Spring project. If one's source of the video is not a device camera, then one can add a video's actual path instead of URL. According to one's need, one can change URL parameters in the algorithms. These algorithms are designed for both LVS & VoDS.

4.1. Kafka configuration setting of JSON object for video analysis. Maintaining the order of messages in a single partition is mandatory while video analysis is performed using JSON objects. Kafka helps to maintain order by putting key value in the JSON object while producing data. To store large messages, some configuration needs to be changed on the server-side. Firstly, configure the Properties file message.max.bytes and secondly replica.fetch.max.bytes at server-side and for consumer side configuration is max.partion.fetch.byte and max.poll.records. These all configurations help to transfer JSON objects to the Spark for video analysis continuously. Some parameters should be taken care of while performing the configuration of Kafka for computing a large amount of data. The following parameters need to be set for the consumer side and producer side configuration.

1. Kafka configuration parameter for stream collector
   (a) kafka.asks = all
   (b) kafka.retries = 1
   (c) kafka.batch.size = 20971520
   (d) kafka.linger.ms = 5
   (e) kafka.compression.type = gzip
   (f) kafka.max.request.size = 2097152
2. Kafka configuration parameter for stream processor
   (a) kafka.max.partition.fetch.bytes = 2097152
   (b) kafka.max.poll.records = 500

At stream collector side, kafka.ask = all denotes when the producer gets acknowledgment from all in-sync replicas to the leader. kafka.retries = 1 indicates the number of retries if any broker goes down. kafka.batch.size means the total bytes of a message to collect before sending it to the producer. kafka.linger.ms signifies letting the producer know about the waiting time to particular ms in the expectation of more records for arrival. GZIP is a type of compress file that compresses the file to a smaller size and is best for faster network transfer. kafka.max.request.size denotes the total size of the record. A stream processor, kafka.max.partition.fetch.byte indicates the maximum amount of data fetch from per partition. kafka.max.poll.records signify letting the consumer know about the maximum number of records returned in a single consumer call for poll() function.

Figure 4.1 describes how to frame data to JSON object flows through Kafka and Spark with leveraged use of OpenCV for analysis. Some of the sources of video can be IoT devices also. Subscription system middleware is used to store IoT captured data into the database. Most IoT devices are required to implement a high-

Algorithm 1 Collect Stream

Input: Camera or File U_{rl}
Output: Pass JSON object to Kafka stream using P_r
Initialization: Create MAT object

```
1: procedure JsonCollectionStream(U_rl)
2:     while Thread.Run() do
3:         if StringUtils.isNumeric(U_rl) then
4:             C_obj = VideoCapture(U_rl)
5:         else
6:              Throw Exception
7:         end if
8:         if C_obj.isOpen() then
9:             Thread.Sleep(500)
10:            if C_obj.isOpen() then
11:                Throw Exception
12:            else
13:                while C_obj.read(MAT) do
14:                    M_cl=getCols(MAT)
15:                     M_rw=getRow(MAT)
16:                    M_ty=getType(MAT)
17:                    J_obj=JSONOBJECT( M_cl,M_rw, M_ty)
18:                    P_r.send(J_obj,T) // Send Stream to KAFKA
19:                end while
20:            end if
21:        else
22:            while C_obj.read(MAT) do
23:                M_cl=getCols(MAT)
24:                 M_rw=getRow(MAT)
25:                M_ty=getType(MAT)
26:                J_obj=JSONOBJECT( M_cl,M_rw, M_ty)
27:                P_r.send(J_obj,T) // Send Stream to KAFKA
28:            end while
29:        end if
30:    end while
31: end procedure
```

concurrency, low latency message system. Liu *et al.* [6] used SSM framework. Spring is an open-source Java framework that provides a lot of functionality for dealing with large scale data, for example, Spring-Kafka connector, Spring JPA, Spring Cloud, and Spring CloudStream. For middleware, the database needs a fast key-value in memory and hence, the Redis database is used to store captured data set with key-value pairs. Redis is an open-source, in-memory data structure store, handed as a database, message broker, and cache. Redis supports abstract data structures like bitmaps, strings, lists, sets, sorted sets, maps, etc.

Algorithm 2 shows how Kafka consumers fetch data and put streams into Spark stream engines. Spark uses MapFunction() function to treat each stream data individually. Spark helps to group stream data using the same Camera ID and makes state from the group and pass the state group data into the Event function. The Event() function is VideoFrameDetection in Algorithm 2. Instead of VideoFrameDetection, one can use the Event() function which analyzes each frame individually and writes its own algorithm for the Event() function. groupByKey() function creates a group using Camera ID from JSON stream. mapGroupWithState() function creates a state for the group and serves each data from each group for Event() function.

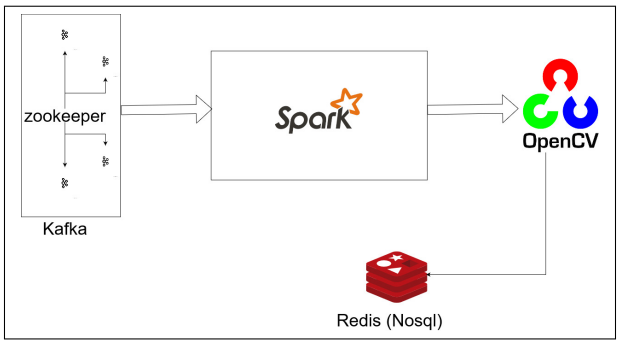


FIG. 4.1. *Kafka streaming and Spark engine*

Algorithm 2 Process Stream

Input: $\mathbf{E_{data}}$ **will be input from kafka**
Output: Video Analysis Operation
Initialization: SparkSession Object

```
1: procedure ProcessStream(E_data,E_key)
2:     S_c=DataType.createStructType()
3:     while True do
4:         D_s=D_s.groupByKey(MapFunction(E_data,E_key)
5:         P_data=D_s.mapGroupWithState(E_data,E_key))
6:         if S_t.exist() then
7:             E_x=S_t.get()
8:         else
9:             P_p=VideoFrameDetection(E_x,K_e,V_v)
10:            if P_p!=Null then
11:                Successful Operation
12:            else
13:                Error In Operation
14:            end if
15:        end if
16:    end while
17: end procedure
```

4.2. Performance analysis of Kafka. In video analysis architecture, it is evident that very high throughput matters for sending video frame stream into distributed networks. In this section, we will discuss Kafka's performance and test results that we encountered. Also, a comparison with other message brokers like RabbitMQ is represented here. Figure 4.2 shows the test results of different resolutions from data tested on Kafka and RabbitMQ. A total of 5379 frames are projected from a three-minute MKV video with different resolutions like 640x480, 720x480, and 1080x720. In Figure 4.2, the x-axis represented even number is Kafka's data and that with an odd number is RabbitMQ's data. You will see that graph of Kafka drop frame rate is quite negligible because Kafka is mainly designed for distributed publishers and subscriber architecture. In our case, Kafka broker is a multi-threaded node, and hence, in-case if any broker fails, then Kafka has supportive fault tolerance management services. Secondly, Kafka can highly maintain order from different partitions with the same topic with the key. Henceforth, the reason why RabbitMQ frame drop average rate is high because RabbitMQ is a single-threaded message queue.

RabbitMQ guarantees to deliver messages but not in order. RabbitMQ does not provide good enough fault tolerance service compared to Kafka because Kafka uses internal storage to store messages. We can store

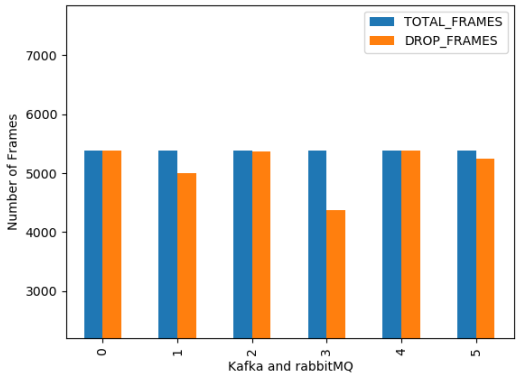


FIG. 4.2. *Analysis of Kafka and RabbitMQ*

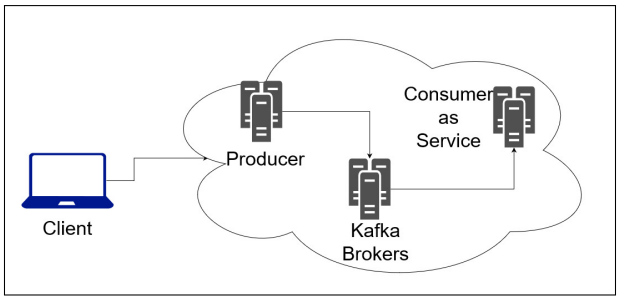


FIG. 5.1. *Full-Service-Mode*

a particular message for a maximum of 7 days using Kafka. In case of any failure, the Zookeeper maintains a pointer for the commit log. We can retrieve data with the help of the pointer. Whereas, in RabbitMQ, queue services discard data if data goes out of the queue. Single thread services in RabbitMQ proves to be a disadvantage for video analysis architecture as it architects need of a distributed environment for large scale services.

Kafka provides publisher and subscriber architecture, and hence, one can serve a stream to multiple consumers with the same order but the same cannot be done on RabbitMQ because, for each user, it needs to maintain queue services. If scaling broker > 3 becomes complicated, it can have a negative performance impact on RabbitMQ. In case, one wants to use RabbitMQ for streaming, then the usage of Redis services with RabbitMQ is mandatory.

5. Cloud based services. Many cloud service vendors provide Kafka as Infrastructure as a Service (IaaS), Wherein cloud vendors provide different resources for Kafka like CPU, storage, RAM over the Internet, which is on payment basis as per the use. The main advantage to move Kafka on the cloud turns out to be automatic scaling up and down of resources according to their requirements. Currently, many cloud vendors like Amazon Manages Streaming Kafka (MSK), Apache Kafka HDInsight provided by Microsoft Azure, Event Streams by IDM provide Kafka services. In this paper, we have used MSK for the Kafka cluster. Two different modes are provided by cloud service vendors. AWS offers a full-service mode where the Kafka cluster and producer resides at the cloud server and the consumer being another service of cloud as shown in Figure 5.1. Another mode is a cluster-service mode where Kafka broker resides at cloud service but producer and consumer reside at local devices like standalone workstations or mobile devices. We have inherited the full-service mode concerning our research work.

Cloud helps in various functions of video streaming like encoding and decoding, media streaming engine, CDN, etc. Five-part development involves a content server, reverse proxy server, load balancing server, stream-

ing server, and storage for VoD clients on smart devices. The content server is responsible for managing video resources on the storage side like a media source database and serving that media source to the end-user in browser information. Load balancing and reverse proxy cache servers are responsible for processing requests faster. NGINX is a webserver which used as a reverse proxy server and load balancer. For streaming media, the Red5 server is used which is an open-source media server that provides LVS services over the cloud. AWS media elemental streaming service or Wowza streaming engine can also be used for the same purpose. In the end, cloud storage technology could use either HDFS or AWS S3.

Content servers were divided into two parts. The first is to give navigation information toward client devices' distributed server and second is to manage user resources and video information and upload file media. Also, for the best caching performance with NGINX, CDN can be used. Users can get a cached video from a nearby edge location. Continuous video streaming with standard encoding technique H.264 and sliding window protocol for continuous frame delivery in mobile is required. H.264 is a compression technology and media streaming engine that helps to convert video streams into H.264 format which is a globally accepted format for almost all devices. AWS media elemental services help to encode video into H.264, RTMP, and Apple HLS. The following explains the points to be taken care of in video streaming.

- *Buffer:* It helps to store incoming video frames and delivers the buffered frame without any delay. It helps in the continuous streaming of video. Buffers and timers are controlled by real-time messaging protocol (RTMP) players. RTMP contains its buffer that holds some beforehand frame in case of any network error and network failure. Thus, buffer helps in fault tolerance from frame loss.
- *Timer:* It helps to synchronize within media streaming servers and mobile devices. The time arrival of the video frame is solved using the same. Synchronization can also be done by timer while decoding.
- *Decoder:* It receives video from the timer section and converts the encoded video into a particular format that a receiver mobile supports. Decoder modules are in-built in the RTMP player.
- *RTMP player:* The player built using the android platform helps to perform LVS & VoDS. It contains a timer and a decoder section. It can help LVS by a buffer frame. HTML embed tag is used to embed URLs of media streaming servers and RTMP automatically starts playing from the initial default frame in the first queue manner. The streaming frame pointer data is stored in the in-memory database Redis. It is used for low latency streaming and requires no buffer.
- *Apple HLS:* This protocol uses adaptive bitrate streaming and supports almost all major streaming devices like browsers, android devices, and operating systems. For audio codecs, Apple HLS helps to stream Advanced Audio Coding - Low Complexity (AAC-LC), MPEG Audio Layer-3 (MP3), High-Efficiency Advanced Audio Coding (HE-ACC+V1&V2). For video codecs, it uses H.265 and H.264.
- *Bandwidth predictor:* But at first, the server requires bandwidth of user for frame resolution that should be done every particular second. It uses Round Trip Time (RTT) sent by the user device. At the client-side, the sliding window protocol is used to calculate the RTT.

5.1. Content delivery network. CDN helps to cache frames from a video and store all frames at a nearby location of the user that we call as an edged location server. The cloud vendors provide different QoE among varying users. Different regions have different scalability, cost, and cache hit rate. AWS CloudFront provides better quality of experiences than Google Cloud CDN for ASIA, AUSTRALIA, and NORTH AMERICA. Table 5.1 shows test results from different AWS CloudFront CDN edge location. It projects that South America has more frame delay around > 5000 msec and hence, can also use another CDN provider to deliver the same content delivered by AWS.

Google Cloud CDN performed better than AWS in South America and Europe. It has been observed that if two users are using the same video streaming link at a time, a little latency is observed which leads to QoE drop. To conclude that QoE drop is not related to latency, different factors are responsible such as low-level computational power for encoding and decoding. If video streaming provides the worldwide area, then we can use different cloud CDN vendors for delivering the best user experience with video. Figure 5.2 shows AWS media services that help in resizing each frame into different resolutions and store it on S3. AWS CloudFront helps to deliver frames using CDN edge servers.

5.2. Performance of AWS services for LVS & VoDS. Figure 5.4 depicts VoDS with different frame resolution times required to get delivered to the cloud and time required to transcode that video frame into

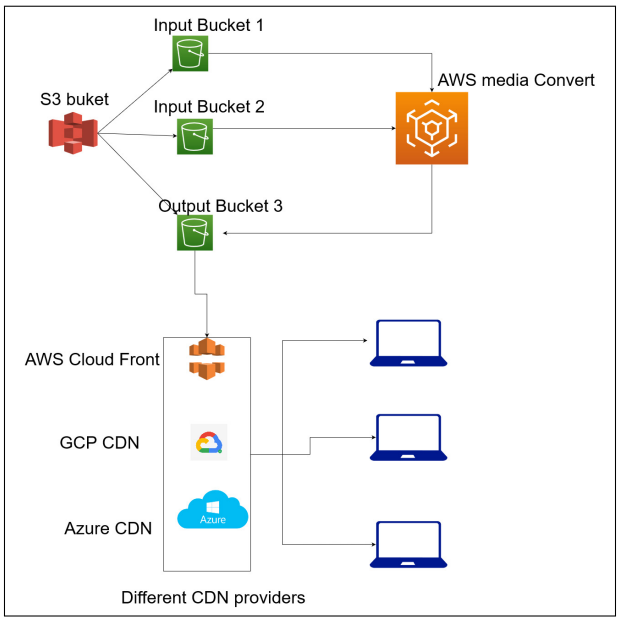


FIG. 5.2. *AWS S3 to different CDN*

TABLE 5.1
Different CDN edge locations

Edge Location	Monitoring Time	Duration (msec)	Status
Madrid	04/13/2020 06:49:50 PM	30359	S
Beijing	04/13/2020 06:49:51 PM	33262	S
Sydney	04/13/2020 06:49:50 PM	30136	S
Copenhagen	04/13/2020 06:49:50 PM	22224	S
Seattle	04/13/2020 06:49:49 PM	14610	S
San Francisco	04/13/2020 06:49:50 PM	79491	F
Mumbai	04/13/2020 06:49:50 PM	30813	S
Warsaw	04/13/2020 06:49:50 PM	28574	S
Paris	04/13/2020 06:49:50 PM	22874	S
Johannesburg	04/13/2020 06:49:51 PM	30971	S
Buenos Aires	04/13/2020 06:49:50 PM	30156	S
Shanghai			F(Timeout)
Amsterdam	04/13/2020 06:49:50 PM	30208	S
Dallas	04/13/2020 06:49:49 PM	8894	S
Brisbane	04/13/2020 06:49:50 PM	30176	S
Denver	04/13/2020 06:49:49 PM	10664	S
Frankfurt	04/13/2020 06:49:50 PM	22362	S
Montreal	04/13/2020 06:49:49 PM	7106	S
Hong Kong	04/13/2020 06:49:50 PM	30211	S
Tokyo	04/13/2020 06:49:50 PM	30142	S
N. Virginia	04/13/2020 06:49:49 PM	10609	S
Washington DC			F(Timeout)
Miami	04/13/2020 06:49:49 PM	17542	S
London	04/13/2020 06:49:50 PM	18354	S
New York	04/13/2020 06:49:49 PM	10354	S

different resolutions. Here, video frame are divided into two formats MP4 and HLS and both frames contain 4 different resolution videos 640x360, 960x540, 1280x720, 1920x1080. Figure 5.5 depicts LVS time taken by frame to get delivered to end-users. This live stream transcode are present in 5 different resolutions 640x360, 640x480, 720x480, 1080x720, 1920x1080. Note that we had used AWS Elemental Media Convert and AWS Elemental Media Live and AWS Elemental Media Package. Figure 5.3 represents whole system architecture.

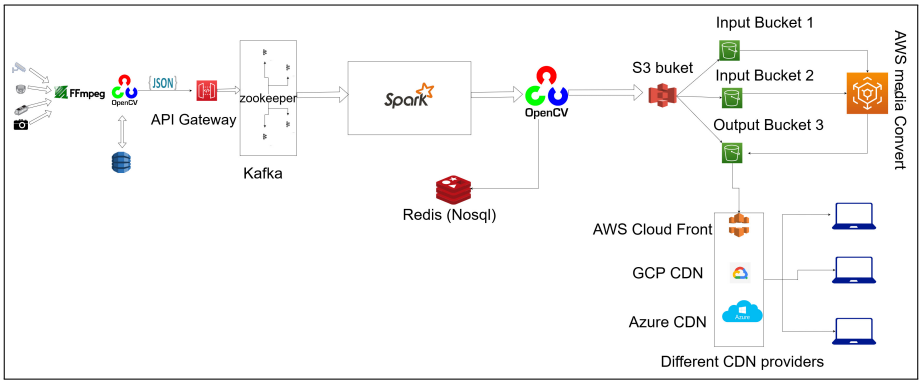


FIG. 5.3. *System Architecture*

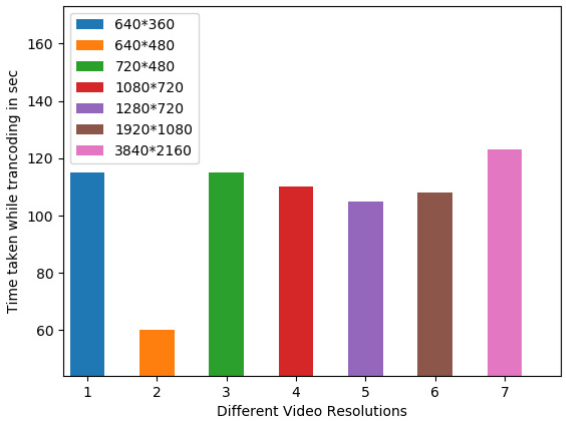


FIG. 5.4. *Time required to transcode VoDS video streaming using AWS Elemental MediaConvert*

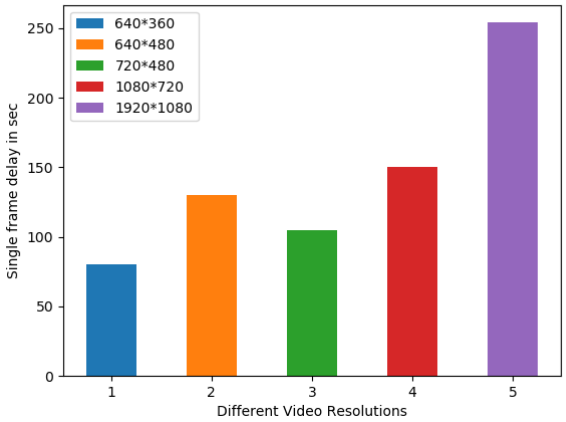


FIG. 5.5. *Time required to deliver single frame in LVS using AWS Elemental MediaLive*

We can see that time taken while transcoding LVS is less as compared to that of in VoDS. LVS is always more important than VoDS because one cannot miss out a frame in LVS due to live streaming. Hence, the QoE parameter is much improved.

6. Performance of Analysis of QoE, Cost and Bandwidth. According to CISCO visual network index, 80-90% of traffic in 2021 will be video-based. Cost, bandwidth, and QoE will be analyzed using different parameters. The parameter is age, viewing habits of the user, early leaving, steady user, viewing pattern of the different users at different times, viewing the content in different devices.

In this system architecture, we have placed a database for video analysis and user observation data for further QoS analysis. For example, we can store the user's different parameters which help to analyze the user's viewing pattern. The first parameter is "who is watching the video" points to the user profile. We can store user profiles that serve as the best video according to user viewing history. The second parameter is "when do users watch video" which means in the database we can store user viewing time with which category show they watch, for example, kids watch cartoon shows during the day, and this, we can suggest cartoon shows during the daytime. The third parameter is "which content is popular within an area". For this, we can acquire edge server location according to popular content being watched there. The fourth parameter is "status of the user for particular video" which denotes video summarization and helps to analyze which parts in video are most popular or user views most of the time. The last parameter is "user's devices", for example, video suggestions might have differences between TV and personal devices like mobile phones and laptops.

Data set for the above operation (D, H, M, S) divide data into 4 columns separated as Display, Hours, Minute, and Seconds. Data set can be created in MongoDB and Cassandra NoSQL database and a second table for profile information which can be stored into the Relational Database Management System (RDBMS). The user table should have 3 columns UserId, Gender, and Age. Spark for data and Zeppelins for data analysis can be used. We conclude that different users impact on QoE, bandwidth, and cost analysis.

To understand the impact of age effect on the server, let us take an example. Two major video content delivery platforms i.e. Netflix and YouTube have different region demands. For example, the Netflix series i.e. House of Cards is very popular in the USA's generation age group of 18-32, whereas, Sacred Games is popular in India's generation age group of 20-30. Thus, Netflix can cutoff costs by not uploading shows like Sacred Games and only upload shows contained among the demand region's edge location. This will decrease cost on the server and Netflix can distribute the load in a nearby location most of that time duration. One can also analyze viewing content as per gender group, to recognize the shows of Netflix that is popular in a particular gender.

Content centric viewing pattern is also different for videos. Some videos are popular among the kids and they are most active during the day time. Hence, we can make different microservices for kid's shows. This can be scaled up and down depending on most likely viewing time. Video browsing behavior also has several categories like watching numerous videos within an hour or watching only interesting part of a video or watching content for a short duration only. In the end, all these parameters help to create a digital marketing strategy for a particular show on a particular platform.

7. Conclusion. This paper presents the possibility of LVS & VoDS video analysis with the OpenCV library of Java webcam capture API to encode video. CPS is a combination of physical processes, networking, and computation and requires real-time captures and suggestions. The presence of CPS and its applications demands on-time video streaming with maintained high quality. Video streaming needs to be very quick from the camera to the cloud and from the cloud server to end-users for streaming video frames quickly. We have a comparative analysis of Kafka and RabbitMQ messaging services and found that drop rate in Kafka is very negligible as compared to RabbitMQ. Hence, we have used the Kafka streaming engine for sending video frames into JSON objects to different consumers. As video analysis needs a fast streaming data frame, therefore, Spark provides that fast streaming platform for analysis. For analysis, the Spark MLlib library is used. In the end, to store all video frames on to distributed storage, we have used AWS S3. To deliver video frames to end-users with the best experience, we should make use of a suitable content delivery network providing good results to users. This paper also devises different factors for QoE. We saw that time taken for transcoding frames in LVS was lesser as compared to that of VoDS and hence giving a better QoE to user. It makes the user to experience the video streaming without any breaks. Single architecture helps to achieve different tasks to

analyze video frames. Video streaming is proven to be growing rapidly since a decade in the every field thus, quality parameters such as QoS, QoE, etc. play a major role for a user's convenience.

REFERENCES

[1] J. Lee, B. Bagheri, and H.-A. Kao, A cyber-physical systems architecture for industry 4.0-based manufacturing systems, *Manufacturing Letters*, vol. 3, pp. 18 – 23, 2015.

[2] Video streaming - statista, `https://www.statista.com/outlook/206/119/video-streaming--svod-/india`, accessed: 2020.

[3] A. Kathuria and S. N. Panda, Video capturing and streaming over ad-hoc networks, in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 2017, pp. 379–382.

[4] A. Ichinose, A. Takefusa, H. Nakada, and M. Oguchi, A study of a video analysis framework using kafka and spark streaming, in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2396–2401.

[5] P. Le Noac'h, A. Costan, and L. Bougé, A performance evaluation of apache kafka in support of big data streaming applications, in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 4803–4806.

[6] Z. Yuan, B. Pang, Y. Du, X. Liu, J. Yao, and C. Kong, Design and implementation of internet of things message subscription system based on kafka, in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, 2019, pp. 603–606.

[7] Y. Xue, H. Zhang, and H. Ma, Performance evaluation of image and video cloud services, in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2018, pp. 733–741.

[8] Z. Liu, Q. Wang, J. Huang, Y. Wu, Y. Wang, X. Jia, and H. Chen, Cloud-based video-on-demand services for smart tv, in *2017 Seventh International Conference on Information Science and Technology (ICIST)*, 2017, pp. 81–84.

[9] S. P. Tamizhselvi and V. Muthuswamy, Adaptive video streaming in mobile cloud computing, in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1–4.

[10] C. Wang, A. Jayaseelan, and H. Kim, Comparing cloud content delivery networks for adaptive video streaming, in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 686–693.

[11] S. Rahman, H. Mun, H. Lee, Y. Lee, M. Tornatore, and B. Mukherjee, Insights from analysis of video streaming data to improve resource management, in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018, pp. 1–3.

[12] C. E. C. F. Batista, T. L. Samilto, L. E. C. Leite, G. L. de Souza, and G. E. da Silveira, Big videos on small networks. a hierarchical and distributed architecture for a video on demand distribution service, in *2005 1st International Conference on Multimedia Services Access Networks, 2005. MSAN '05.*, 2005, pp. 15–19.

[13] K. Yi, Y. Lee, and J. Joo, A fast video decoding technique by means of converting input video stream into forward-oriented format stream in little-endian systems, in *2015 7th International Conference on Multimedia, Computer Graphics and Broadcasting (MulGraB)*, 2015, pp. 15–18.

[14] F. Machida, M. Fujiwaka, S. Koizumi, and D. Kimura, Optimizing resiliency of distributed video surveillance system for safer city, in *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2015, pp. 17–20.

[15] X. Li, M. A. Salehi, and M. Bayoumi, High performance on-demand video transcoding using cloud services, in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016, pp. 600–603.

[16] J. Wang, W. Xu, and J. Wang, A study of live video streaming system for mobile devices, in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 2016, pp. 157–160.

[17] H. Su, S. Han, and C. Yang, Caching policy optimization for rate adaptive video streaming, in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 713–717.

[18] K. Patel and G. Panchal, Smooth video streaming in bandwidth fluctuating environment, in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 79–83.

[19] S. Kanrar and N. K. Mandal, Approximation of bandwidth for the interactive operation in video on demand system, in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 343–346.

[20] B. R. Hiraman, C. Viresh M., and K. Abhijeet C., A study of apache kafka in big data stream processing, in *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, 2018, pp. 1–3.

[21] H. Wu, Research proposal: Reliability evaluation of the apache kafka streaming system, in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2019, pp. 112–113.

[22] H. Wu, Z. Shang, and K. Wolter, Performance prediction for the apache kafka messaging system, in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019, pp. 154–161.

[23] Y. Tu, Y. Lu, G. Chen, J. Zhao, and F. Yi, Architecture design of distributed medical big data platform based on spark, in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019, pp. 682–685.

[24] M. A. Uddin, A. Alam, N. A. Tu, M. S. Islam, and Y.-K. Lee, Siat: A distributed video analytics framework for intelligent video surveillance, *Symmetry*, vol. 11, no. 7, 2019. [Online]. Available: `https://www.mdpi.com/2073-8994/11/7/911`

[25] Mahapatra, Tanmaya and Prehofer, Christian, Graphical flow-based spark programming, *Journal of Big Data*, vol. 7, no. 4, 2020.

[26] W. E. Shabrina, D. Wisaksono Sudiharto, E. Ariyanto, and M. A. Makky, The qos improvement using cdn for live video

streaming with hls, in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, 2020, pp. 1–5.
[27] G. van Dongen and D. Van den Poel, Evaluation of stream processing frameworks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1845–1858, 2020.
[28] S. Anveshrithaa and K. Lavanya, Real-time vehicle traffic analysis using long short term memory networks in apache spark, in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–5.

Edited by: Anand Nayyar
Received: May 15, 2020
Accepted: Jul 14, 2020