VIDEO ANALYTIC DATA REDUCTION MODEL FOR FOG COMPUTING

Abdolreza Abhari
Dipak Pudasaini

Distributed Systems & Multimedia Processing
Laboratory (DSMP lab), Department of Computer
Science, Ryerson University
350 Victoria Street, Toronto, ON, CANADA
{aabhari, dpudasaini}@ryerson.ca

ABSTRACT

The large volumes of video data cause network congestion and high latency in the centralized cloud computing system. Fog computing architecture that enables employing edge devices has already been used to address these problems. This paper proposes an application model called Video Analytic Data Reduction Model (VADRM) that divides video analytic jobs into smaller tasks with fewer processing requirements. The prototype of VADRM application model for typical video analytics applications (i.e., surveillance cameras) is implemented by Convolutional Neural Network (CNN). The analytical model is created based on the workload characterization of the prototype and used in the general simulation to measure the effectiveness of VADRM for employing edge computing instead of the cloud. The results show VADRM can allocate 45% of the data size for edge processing and 55.50% for cloud processing. iFogSim toolkit is used to simulate the fog environment and measure network performance when using VADRM model.

Keywords: Video Analytics, Fog Computing, Edge Computing, iFogSim, Convolutional Neural Network

1 INTRODUCTION

Video analytics is a kind of application that processes streamed video data. The huge volumes of video data should be processed in these applications such as traffic systems, social media, geographic systems, the film industry, and more. Powerful technology is needed to process large volumes of video data that require low network latency. Research about integrating fog/edge computing and cloud computing with IoT devices tries to address the problems of processing large volumes of streaming data such as video analytics applications. In this paper, we propose the model of general video applications that can employ the combination of edge computing and cloud computing which is currently possible by using the fog computing architecture, an emerging technology for the IoT-based systems.

Fog computing is an architecture proposed by Cisco in January 2014 to shift the data processing of IoT to edge devices which are small capacity resources such as servers or terminals with low storage capacity and low processing capacity (Yousefpour, Ishigaki and Jue 2017). A video analytic application is a huge application usually processed by Graphics Processing Unit (GPU) that is not available on edge devices close to the security camera. The video data are directly passed and processed into the cloud has been proposed in (Anjum et al. 2019). However, the problem of the centralized approach is high latency and more network uses to transfer data into the cloud which makes it a bottleneck. The video analytics jobs are large applications that are called edge computing killers (Ananthanarayanan 2017). Thus, edge-enabled applications are more attractive because their distributed processing can minimize high latency and reduce bandwidth consumption. To address this problem, we proposed splitting the video analytic applications to be able to be processed by edge devices that have been presented in (Pudasaini and Abhari 2020).

The idea is there are usually several stages required to complete a video analytic application. Assuming for the P= {s_1, s_2,…,s_n}, the total amount of data is D, and different data processing stages required to complete the problem P are shown by *s* as the needed stages. The goal is to find a function such that $f: D \rightarrow N^+$, where $d_i \in D$ is the original data size, and $d_{i-1} \in N^+$ is the new data size, when applying function f the result of each stage has a smaller size than the previous stage. The novelty of this data reduction method is to divide the video analytics into different stages with a reduced data size for each stage.

The result is the different tasks of the original application can be run in a distributed environment. In our previous research (Pudasaini and Abhari 2021) (Pudasaini 2021), we showed dividing a big video analytic task into subtasks with fewer data processing requirements is possible. In this paper, the process of dividing video analytic applications into subtasks to reduce the size of each task is introduced as Video Analytic Data Reduction Model (i.e., VADRM) which is examined in the fog computing environment. VADRM is a distributed software architecture in which the edge nodes close IoT devices process the data to minimize network bandwidth and latency. The VADRM structure for fog computing is shown in figure 1 as follows:

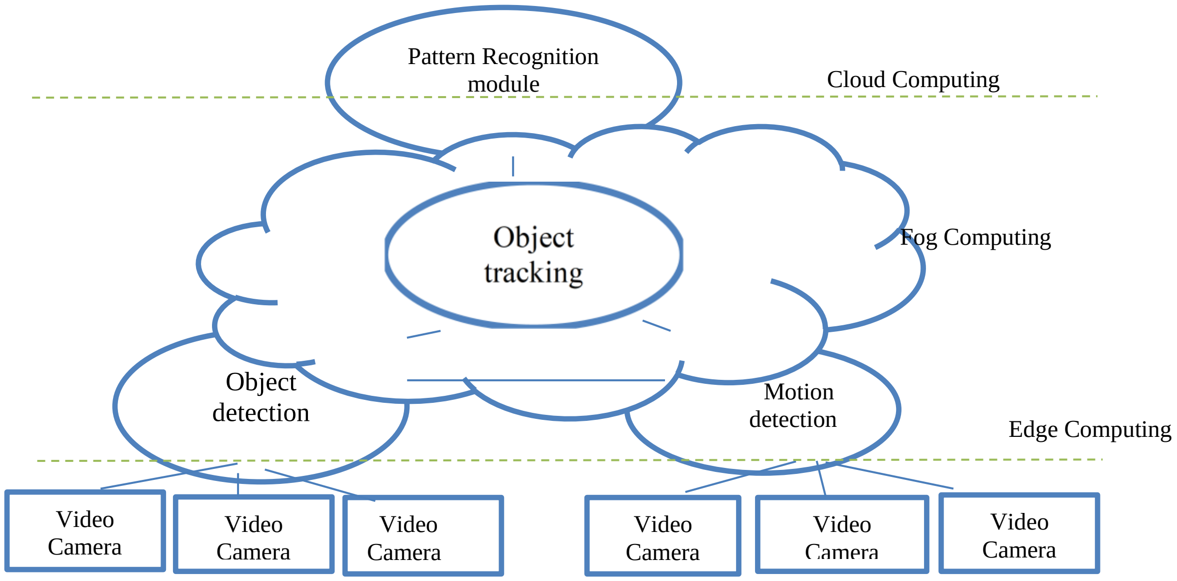


Figure 1: VADRM structure in a fog computing environment

The main contributions of this research that are discussed in this paper are given below:

- Developing VADRM architecture based on Distributed Dataflow Approach: Distributed Dataflow has been proposed in (Giang 2015). It is a distributed software architecture that can be run on edge and fog computing nodes. In this model, the software is designed as the modules in form of a directed graph in a way that the output of each module can be input to the next module. VADRM also has the same structure focusing on reducing the data when transferring to the next module. Therefore, each VADRM module reduces the data size of video files in the consecutive phases of data processing in video analytic applications. We implemented VADRM prototype capable of video size reduction in each consecutive phase of the application by using deep learning.
- General Simulation: VADRM is simulated by developing analytical models for video analytic applications. Random workloads are similar to the real data of VADRM prototype in the simulation.
- Simulating the fog with real data from VADRM prototype: The real video files created by the VADRM prototype are fed into a fog simulator. The simulator is iFogSim which is used to measure the network performance of the video analytics application created based on the VADRM model.

2 LITERATURE REVIEW

Cloud computing is extended into the fog and edge computing with IoT devices. The impact of extended cloud into edge and cloud computing on networking services has been proposed (Shirazi et al. 2017). The real-world matrix to evaluate the performance of edge, fog, and cloud computing has been proposed in (Aslanpour et al. 2020). It is important to evaluate the performance of resource scheduling, resource allocation, and resource execution in the edge-cloud environment. In (Laroui et al. 2021), the benefits of edge computing than that over cloud computing have been described. This paper mainly focused on task scheduling, security, privacy, and blockchain in the edge-IoT environment. The resource management techniques for cloud and edge computing have been proposed (Mijuskovic et al. 2021). It has provided the evaluation techniques that can be applied for edge, cloud, and fog devices.

In (Murshed et al. 2022), the survey of machine learning algorithms in the network edge has been presented. The transmission of raw data to the cloud causes many problems such as increased communication costs, delayed system response, and privacy concerns. If the additional computing devices at the edge of a network closer to IoT devices are placed, they will be used to run machine learning algorithms. By this technique, machine learning is deployed at the edge of computer networks. The edge-based video analytics system uses deep learning to recognize the objects in a large-scale IoT video stream has been proposed (Ali at al. 2018). The three stages of this model that are frame loading, preprocessing and object detection are performed in edge infrastructure, and then object recognition is performed in the cloud.

In this paper dispatching the jobs among the Edge and Cloud processors is considered a real-time decision-making problem, without prior knowledge about the jobs. Sharifi et al. presented a Queueing-based model (Sharifi, Abhari, and Taghipour 2021), and a mixed-integer linear programming (MILP) model (Sharifi, Abhari, and Taghipour 2021), to optimize the dispatching strategy for a non-real-time problem. They have considered that all data in terms of arrival time and the size of the jobs (number of frames) are known at the beginning of the fog mission horizon. Although this was a non-realistic assumption that all data is available, the results could help researchers to develop better dispatching strategies compared to the existing ones.

3 METHODOLOGY

The method is to create the video analytics application as the VADRM model which is shown below:

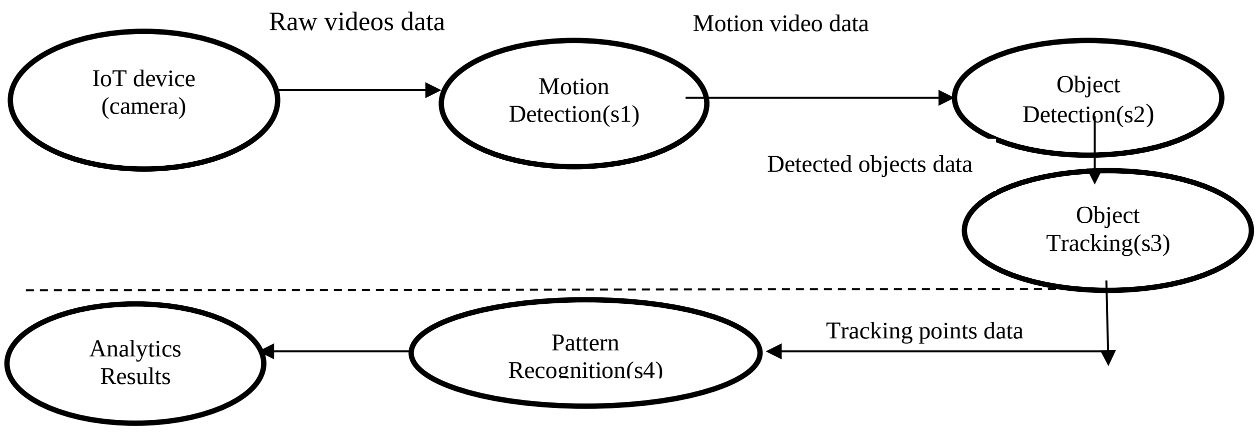


Figure 2: VADRM application pipeline model

This model divides the video analytic application into different phases when using edge computing and cloud computing architecture. The VADRAM application architecture shown in Figure 2 produces small size of data that is passed into the cloud for pattern recognition. Both Figures 1 and 2 show different aspects of the VADRM architecture when used in the fog environment in which a video camera (IoT devices) captures the motion of the vehicles and passes them into the edge devices. The edge nodes close to IoT sensors (i.e., cameras) host object detection and object tracking modules. The results of edge processing can be shown by monitors which are actuators in fog computing. Finally, the trajectories of the detected and tracked objects are sent into the public cloud for pattern recognition.

3.1 Prototype Implementation

The VADRM prototype implemented by CNN in four modules of motion detection (s1), object detection (s2), object tracking (s3), and pattern recognition (s4) that are explained next.

Motion Detection (s1): The motion detection module detects the motion of moving objects by using video cameras. The motion of the vehicles will be captured by the cameras that are installed in different places. The motion video data will be passed into the object detection module for the detection of the objects.

Object Detection (s2): The detected motion of the video is passed into the object detection module. The main mechanism of the detection phase is to extract the features of the objects for detection. You Only Look Once (YOLO) is used which is the CNN-based popular object detection method. In this experiment, we use the YOLO v3 object detector for vehicle detection. YOLO v3 is a fully CNN-based object detection method used in many real-time applications to detect the objects (Redmon and Fardhai 2018), (Redmon et al. 2016). The YOLO uses a special backbone network called the Darknet. The reason for this network in our model is because of the high-speed frame per second (fps) and good accuracy. There are 24 convolutional layers followed by 2 fully connected layers. The convolutional layers are pre-trained with the ImageNet dataset and the framework used in this model is the Darknet.

Object Tracking (s3): The output of the object detection module is forwarded to the object tracking module. The object tracking module tracks the path of the moving objects using CNN. The results from the object detection module ate class categories and bounding boxes (region of interest). The bounding boxes and the regions of interest in the frames are cropped and passed into the CNN-based appearance model. The results from the appearance model are the feature description for all detected objects. The tracking method used in this paper is Deep SORT, which leverages the feature descriptions for the association of the detected objects that uses the Kalman filter to predict the objects' location in the next timestamp. The CNN architecture for the object tracking method uses two convolutional layers, one max-pooling layer, and six residual layers (Zagoruyko and Komodakis 2016).

Pattern recognition (s4): The CNN-based object tracking module tracks the path of the moving objects and then passes it into the pattern recognition module. The purpose of the pattern recognition module is to find the normal or abnormal behaviors of the objects and recommend them to the users. The trajectory pattern recognition method using the supervised and unsupervised learning methods has been explained and tested in (Pudasaini and Abhari 2019).

3.2 Workload Characterization

Dataset: The real data are collected from VADRM prototype and analyzed. To find the distribution model of data, curve fitting was used. The real number of frames (per second) values in different phases of the video analytic model are collected when passing different sizes of video files which are given below.

Motion Detection: 9.21, 8.88, 9.51, 9.26, 9.12, 9.01, 8.95, 9.11, 8.96, 9.03, 8.99, 9.02, 9.30, 9.22, 9.25, 9.14, 9. 17
Object Detection: 7.56, 7.50, 7.65, 7.90, 7.67, 7.66. 7.88, 7.77, 7.85, 7.62, 7.48, 7.69, 7.83, 7.56, 7.49, 7.72, 7.81
Object Tracking: 6.14, 6.0, 6.11, 6.17, 6.09, 6.19, 6.05, 6.13, 6.14, 6.12, 6.23, 6.15, 6.25, 6.29, 6.05, 6.17, 6.22

Curve Fitting Results: The empirical distribution of the above data is determined and used in the curve fitting method to find the closest distributions to model the data. The analysis is performed in Python with the above 51 data values which are collected from the module of VADRM in a real video analytics application tested in our previous works. The curve fitting program compared empirical distribution with other candidate distributions. The two best candidates are Beta and Uniform distributions shown below.

Table1: Confidence score of Beta and Uniform distribution

Distribution	Confidence Score
Beta	7.35
Uniform	8.65

The confidence scores of Beta and Uniform distribution are shown Table1. Based on these scores, Beta distribution and the Uniform distribution are more appropriate for this dataset. Uniform distribution is used as a model for creating the artificial data for the general simulation model explained next.

4 VADRM GENERAL SIMULATIONS WITH ANALYTICAL MODELING

For 5G network, fog computing is proposed to use and manage edge devices. In this section, we want to examine the resource management strategies when the video application is created according to VADRM model and implemented on any type of edge-enabled networking. Since resource management is an optimization problem, the mathematical modeling and objective functions should be defined. Therefore, this section aims to present mathematical equations required for resource management when using VADRM as an application in the simulation.

For simulating the VADRM-based application, the pipeline modules should be modeled. For that reason, a dispatcher as a queue of jobs is considered which can be a database of the waiting jobs in a real system. The dispatcher is responsible for distributing the jobs according to the pipeline execution order over the available edge and cloud devices. Resource management by dispatching the jobs is done in fog computing with the help of fog routers (e.g., Cisco routers and gateways). In Section 5, iFogSim uses its default resource management strategy called Edge-ward strategy that assigns tasks to the different edge and fog devices (Gupta et al. 2017). In this section, a central controller transfers the data of the tasks based on a defined resource management policy. Modules can be hosted on the nodes or can be executed as web services, multi-tenant applications or other fog application placements methods as (Ghobaei-Arani,2020).

To simulate the data transfer between pipeline modules, we assume that after the scheduler sends the job to edge and cloud devices based on their availability, it will receive the resulting data and store it in its database to send to the next module. For that reason, the variable l_j is considered to calculate the two ways latency between the dispatching center and processor j. For execution of a task (e.g., object detection) the dispatcher first checks if the predecessor module (e.g., motion detection) is already executed or not. If previous modules are executed then the required data fetches from the database and together with the task information, will be sent as a new job. If we assume the interarrival and service times are independent, the closest queuing model to the simplified version of this scenario is a G/G/s queue model. Instead of using a generalized queuing model with s servers since the goal of VADRM is to minimize the cost by avoiding sending all the jobs to the cloud, the mathematical models are developed below as the optimization problem to be able to test different dispatching strategies. Equations 1 to 5 are objective functions and formulas, and Equations 6-1 to 6-4 are constraints. The objective function shown by Equation(1) is to minimize the number of jobs sent to the cloud. Thus only when all edge devices get busy and there are still jobs that need to be served the central controller will send the job to the cloud center. The scheduler updates information for nodes and modules when it receives the data after executing a module on the node. In the simulation in this section, jobs are served based on their arrival and availability of the edge device. In section 5.2, the VADRM model is implemented as its pipeline of the modules in the simulated fog environment to measure the pipeline and network latency.

The list of the indexes, parameters and the decision variable which are used in this section are as follows:

Indexes

i: Index of the jobs, $i = 1, \dots, n$,

j: Index of the processors, $j = 1, \dots, m$, in which m belongs to the cloud and $1, \dots, m-1$ are for the EPs.

Parameters

S_i: The size of the job i (frame),

P_j: Maximum processing capacity of the processor j (frame per second),

$t_{i:}$ Arrival time of the job i to dispatching center,

B_i: Begin time of processing job i;

F_i: Finish time of processing part job i;

C_i: Completion time of processing job i;

c_1: The cloud processing cost (frame),

l_j: Latency between dispatching center and processor j,

Decision variable

$x_{i,j}$: Binary variable, if the job i allocates to processor j is equal to 1, else is equal to 0,

Mathematical equations used for calculating job completions are as follows:

$$Min \;\; Z = c_1 \sum_{i=1}^{n} x_{i,m} . S_i \tag{1}$$

$$s.t:$$

$$\sum_{j=1}^{m} x_{i,j} = 1; \quad \forall i = 1, \dots, n \tag{2}$$

$$B_i = t_i + \sum_{j=1}^{m} x_{i,j} . \frac{S_i}{tr_j}; \quad \forall i = 1, \dots, n \tag{3}$$

$$F_i = B_i + \sum_{j=1}^{m} x_{i,j} . \frac{S_i}{P_j}; \quad \forall i = 1, \dots, n \tag{4}$$

$$C_i = F_i + \sum_{j=1}^{m} x_{i,j} . \frac{S_i}{tr_j}; \quad \forall i = 1, \dots, n \tag{5}$$

$$x_{i,j} . \left(t_i + \frac{S_i}{l_j} \right) \geq y_{i,\grave{i},j} . \left(t_{\grave{i}} + \frac{S_{\grave{i}}}{l_j} + \frac{S_{\grave{i}}}{P_j} \right); \quad \forall i = 2, \dots, n; \grave{i} = 1, \dots, i-1; \quad j = 1, \dots, m-1 \tag{6-1}$$

$$y_{i,\grave{i},j} \geq x_{i,j} + x_{\grave{i},j} - 1; \quad \forall i = 2, \dots, n; \grave{i} = 1, \dots, i-1; \quad j = 1, \dots, m-1 \tag{6-2}$$

$$y_{i,\grave{i},j} \le x_{i,j}; \quad \forall i = 2, \dots, n; \grave{i} = 1, \dots, i-1; \quad j = 1, \dots, m-1 \tag{6-3}$$

$$y_{i,\grave{i},j} \le x_{\grave{i},j}; \quad \forall i = 2, \dots, n; \grave{i} = 1, \dots, i-1; \quad j = 1, \dots, m-1 \tag{6-4}$$

$$x_{i,j}, y_{i,\grave{i},j} \in \{0,1\} \tag{7}$$

4.1 Analytical Simulation Results

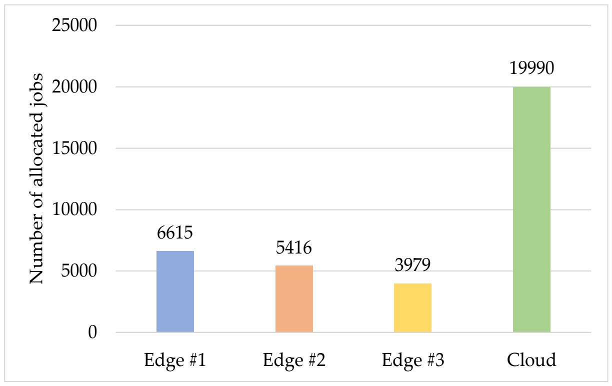
The simple dispatching strategy was used to conduct a simulation of the general VADRM model. The strategy, sorts the EPs based on preferences, such as the power of the EP in terms of frames per second (fps). The arrived job (i.e., data for modules) should be dispatched to the first free EP and the resulting data will be sent back to the dispatcher. By considering the bandwidth between the dispatcher nodes (Equation 6-1) the finishing time for nodes and completion time of the jobs (receiving output data by dispatcher) will be computed according to Equations (4and5) . If there is no idle EP when sending a job, the job should be dispatched to the Cloud Processing (CP) since there is no queue before the EPs. This paper developed a first-in-first-out strategy and sorted the processors based on their processing speed (fps) from high to low. We consider a ten-hour time frame for simulation duration to evaluate the system. So, we have 3600 jobs, which arrive at the dispatching center by a one-second lag. The size of the jobs has a Uniform distribution between five to eight frames. So, we can produce the size of the jobs randomly using Equation (8)

$$S_i = 5 + 3 \times Rand(0,1) \tag{8}$$

In Equation (8), $Rand(0,1)$ is a Uniform random variable between zero and one. We consider three EPs with the processor power of 2, 1,5 fps and a CP with the processing speed of 10 fps. Finally, the bandwidth between the dispatching center and EPs is considered as ten fps (almost equal to 50MB/s) and between the dispatching center and CP as five fps (i.e., 25MB/s) . We codded the model on MATLAB 2021a and ran the codes on a DESKTOP-5TCLFOR with Intel(R) Core (TM) i5-4460 CPU of 3.20 GHz processor, 8.00 GB Installed RAM, and 64-bit operating system, x64-based processor. Since the size of the jobs is random, we ran the codes ten times, and the average values are reported in Table 2 and Figures 3 to 4 as follows:

Table 2: Some metrics of simulation

EP #1	EP #2	EP #3	CP	Total
Number of allocated jobs				
6615	5416	3979	19990	36000
Size of the processed jobs				
43026.6	35224.4	25800.5	129722.6	233774.1
Average size of the processed jobs				
6.50	6.50	6.48	6.49	6.49
Rate of the idle times				
%0.52	%3.68	%7.13	%0.13	---



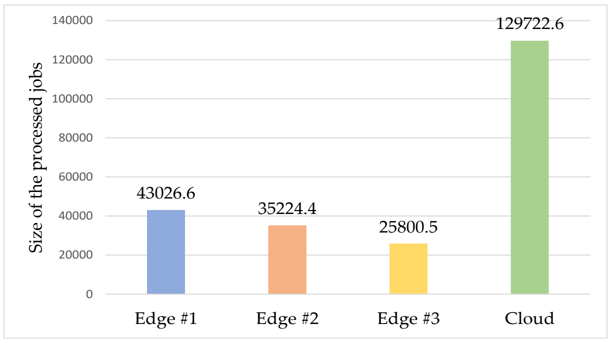


Figure 3: Number of allocated jobs to processors

Figure 4: Size of allocated jobs to processors

The simulation results of the analytical model show that the size of allocated jobs in edge devices is 44.5% and cloud devices is 55.5% respectively, as shown in Figure. 4. The conclusion is that using VADRM can shift 44.5% of jobs that are supposed to be done on the cloud to edge devices.

5 SIMULATION OF VADRM ON FOG COMPUTING

The three-layer structure of fog computing makes it possible to send the data collected by IOT devices to intermediate devices such as edge and fog devices in the network (Laroui et al. 2021). For video analytics applications, the IOT devices are usually security cameras. The data processing is related to detecting and tracking the object of interest by processing the video files. The modular structure of VADRM is suitable fog computing because the modules of object tracking, object detection and object recognition can be hosted by fog devices close to the cameras that can process the video files with low network latency. In the video analytics applications using only cloud computing with high latency creates a bottleneck and as it was shown in section 4.3, VADRM is capable to push the processing of almost half of the data to the edge devices. Thus, if VADRM is employed in the fog network, we expect to get more benefits because the fog routers will be sending more data to the edge network for processing. To test this hypothesis iFogSim simulator is used when assuming VADRM already divided the video analytics jobs into smaller tasks.

5.1 iFogSim Simulator

The structure of iFogSim (Mahmud and Buyya 2018) simulator of the fog computing environment is shown

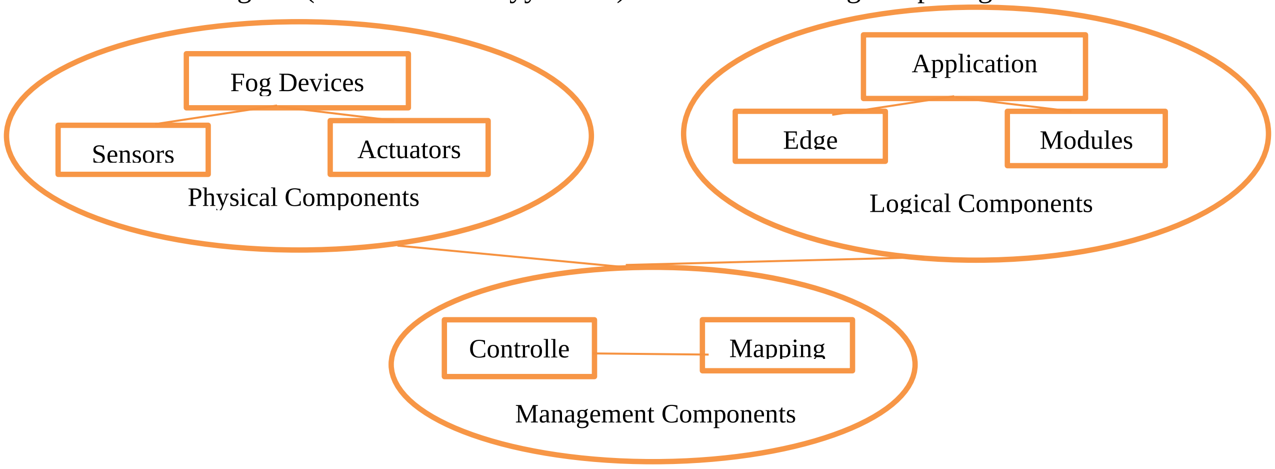


Figure 5: Components of iFogSim(Mahmud and Buyya 2018).

in fig 5. iFogSim is the java-based toolkit derived from the CloudSim simulator. It is used for the evaluation of resource management techniques. It is a discrete event simulator for IoT devices and fog devices. In this system, sensors first collect data and pass them either event-based way or periodically, then fog devices retrieve and process these data. As figure 5 shows, the toolkit consists of a set of physical entities that may be in various types such as fog devices, sensors, etc. It also consists of a set of logical entities used to model the application use cases. IFogsim components can define the resource management and topology for the application. For the video analytic system, we provided the list of devices, modules, and application modules for the controller to run the simulator. The data transferred between modules are called tuples that can be configured for different applications. These iFogSim components are shown in figure 5.

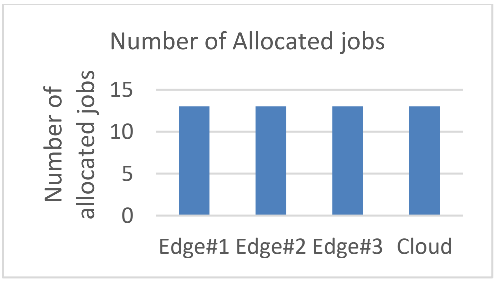
5.2 Data Created from VADRM and Simulation Results using iFogSim

The real data of VADRM used as the tuples in the simulation by modifying the related class in iFogSim simulator. The setting on the simulator modules is based on VADRM model. The scalar factor 150 is considered to convert frames into million instructions because the original work of iFogSim has the CPU length (in Million Instructions) with the values of less than 5000 . In the VADRM prototype, when the input video is passed into the object detection module implemented by YOLO as explained in section 3.1, the output data size will be reduced. This data is passed as input into the object tracking module, which reduces output data size. Based on the real values of fps in VADRM-CNN based prototype, the size of data is reduced from one phase to another phase of video analytic application, which are collected as:

Percentage of video size reduction after object detection (unit is frames) = (9.12-7.68) *100/9.12=15.7%

Percentage of video size reduction after object tracking (unit is frames) = (7.68-6.14) *100/7.68=20.0%

The collected data from the prototype used as the tuples in iFogSim. The average value sizes of tuples in a million instructions are used for the simulation setup parameters of the CPU of edge devices. After running the simulation with the tuples from VADRM data, the number of jobs distributed into edge and cloud devices is shown in Figure 6. The VADRM model reduces the size of data so that 82% of the size of jobs are allocated in edges and only 18% of size of jobs are allocated in cloud devices which is shown in Figure 7. These results show more improvement compared the the job allocation results achieved from general simulation of VADRM one in section 4.The iFogSim also used to find the network performance.



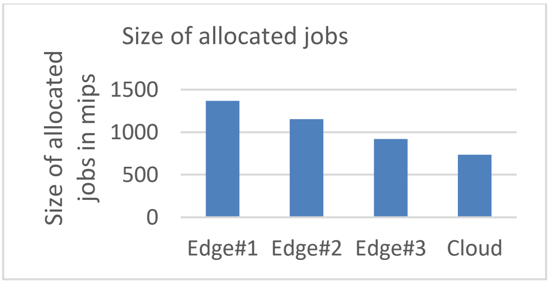


Figure 6: Number of allocated jobs to devices

Figure 7: Size of allocated jobs to devices

There are three cases examined for finding network performance. The VADRM model uses edge computing and cloud computing for the better performance. This case is compared with edge-only and cloud-only computing modes. The network topology and network configurations are similar to the original paper of iFogSim (Gupta et al. 2017). Based on the real data from VADRM, the network performances are measured by iFogSim simulator. The network performance parameters measured in this paper are latency/loop delay, network uses, and energy consumption. The Latency or loop delay is the time taken for an application loop to execute. In the video surveillance system, the application loop starts from the camera sensors producing the video stream, and goes through the motion detection, object detection, object tracking, and finally pattern recognition. Figure 8 shows the network usage which is the maximum amount of data transmitted over the network during the application loop. As shown, when the number of cameras increases, the network usage also increases, but the edge and cloud configuration has less network usage (which is shown by the network traffic in Bytes) than other configurations.

Energy consumption is the amount of energy or power used in the network considering the number of used devices. Figure 10 shows power consumption is almost equal for three configurations when the number of cameras increases. However, it is slightly less than the others for only-cloud configuration because the number of resources is reduced only to the cloud resource. The network latency also is measured by increasing the number of cameras connected to the edge devices. Figure 9 clearly shows the latency of the cloud-only configuration is significantly reduced by two other configurations. The comparison of network uses, latency and energy consumption are explained in Figure 8 to Figure 10. The results show that the network uses are saved, and the latency is reduced using edge and cloud in comparison to only cloud-based approach with the cost of slightly higher energy consumption.

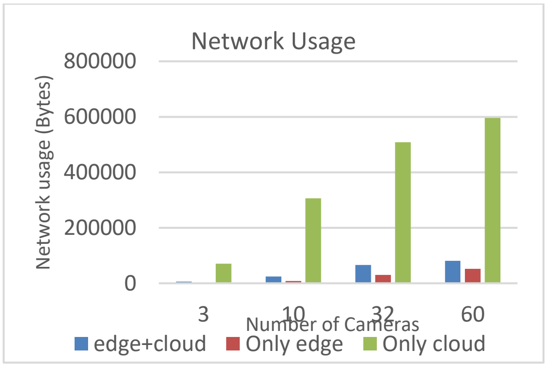


Figure 8: Comparison of network uses on real data

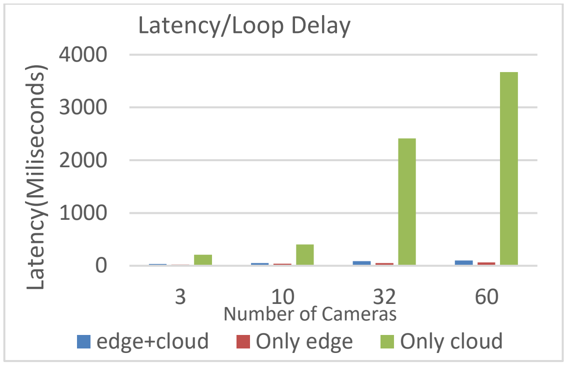


Figure 9: Comparison of latency on real data

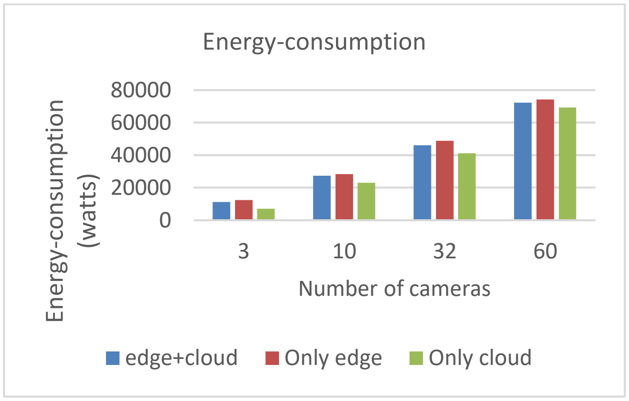


Figure 10: Comparison of energy consumption on real video data

In summary, the results of the analytic simulation of VADRM show shat the size of allocated jobs in edge devices and cloud devices is 44.5% and 55.5%, respectively (as shown in Figure 6). Using the VADRM model in fog computing structure, the jobs allocated in edge devices and cloud devices in iFogSim are 82% and 18%, respectively, as shown in Figure 7. Figure 8 and Figure 9 show that when the number of cameras is increased, network usage and latency reduce significantly in the edge and cloud model, which is less than 13% of using only cloud computing. These results conclude that the size of allocated jobs in the cloud is reduced even more by using fog computing and VADRM video analytics application model.

6 CONCLUSION

This paper tested the proposed approach for video analytic applications called VADRM to process the video data in edge devices and the cloud in the fog computing environment. The VADRM model divides the video analytic application into several small tasks with fewer processing requirements. The curve fitting of real data from VADRM prototype shows Beta and Uniform distributions are more appropriate to model the data of this application. The simulation is performed in two ways to measure VADRM model performance. The analytical simulation for testing the general model by simulating the pipeline of the modules and fog simulation for testing the model in the fog environment.

For the analytical model, simple resource management based on FIFO is developed, and verified the results using simulation. The analytical models presented for the general VADRM simulation can be used to test resource management and load balancing for a network of edge devices. This simulation showed that we can push 45% of the processing to the edge devices by splitting jobs that VADRM does. The simulation results from the actual data of VADRM prototype when it was used in the simulated fog computing showed more improvement. The number of jobs processed in edge and cloud configuration is more than the only traditional cloud-based approach for processing video analytic applications when using fog computing. Finally, the network latency reduced significantly in the edge and cloud model to less than 13% of using only cloud computing in a fog environment.

The conclusion is that using the VADRM model is beneficial in creating video analytics applications. The proposed VADRM model is tested for simple video analytics applications based on security cameras. However, it can be used for board areas of video analytics applications such as smart city development systems, video surveillance systems, traffic management systems, and more. The proposed video analytic model will be more efficient when fog computing and edge devices are used in 5G networks.

Future work's direction will be to improve the proposed model by examining a more advanced CNN-based methods with VADRM model or video analytic applications.

ACKNOWLEDGEMENT

This research is partially funded by FOS Dean Research Fund DRF, Ryerson University, Canada. We thank our colleague Dr. Mani Sharifi who provided insights in the analytical simulation section of this paper.

REFERENCES

Ali, M., A. Anjum, M. U Yaseen1, A.R. Zamani, D. Balouek-Thomert, O. Rana, and M. Parashar. 2018. “Edge Enhanced Deep Learning System for Large-Scale Video Stream Analytics”. *IEEE 2nd International Conference on Fog and Edge Computing*, Washington DC, USA.

Ananthanarayanan, G., P. Bahl, P. Bodik, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha.2017. “Real-Time Video Analytics: The Killer App for Edge Computing”. *Computer*, vol. 50, no. 10, pp. 58-67.

Anjum, A., T. Abdullah, M. Tariq, Y. Baltaci, and N. Antonopoulos. 2019. “Video Stream Analysis in Clouds: An Object Detection and Classification Framework for High Performance Video Analytics”. *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1-16.

Aslanpour, M. S., S. S. Gill, and A.N. Toosi. 2020. “Performance Evaluation Metrics for Cloud, Fog and Edge Computing: A Review, Taxonomy, Benchmarks and Standards for Future Research”. *Internet of Things* vol. 12, pp. 1-20.

Giang, N. K., M. Blackstock, R. Lea, and V. C. M. Leung. 2015. “Developing IoT applications in the Fog: A Distributed Dataflow approach”. *5th International Conference on the Internet of Things (IOT)*, Isfahan, Iran. pp. 155-162.

Ghobaei-Arani, M., Souri, A. and Rahmanian A.A. 2020.“Resource Management Approaches in Fog Computing: a Comprehensive Review”. *Journal of Grid Comput. Springer Nature*, vol. 18, pp**.** 1–42.

Gupta, H., A. V. Dastjerdi, S. K. Ghosh, and R. Buyya. 2017. “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments”. *Software Practice & Experience*, vol. 47, no. 9, pp. 1275- 1296.

Laroui, M., B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani. 2021. “Edge and Fog Computing for IoT: A Survey on Current Research Activities & Future Directions”. *Computer Communications 180*, pp. 210-231.

Mahmud, R. and R. Buyya. 2018. “Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit” *Fog and edge computing: Principles and paradigms*. pp. 1-35.

Mijuskovic, A., A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga. 2021. “Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification”. *Sensors,* vol. 21, no. 5, pp. 1-22.

Murshed, M.G.S., C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain. 2022. “Machine Learning at the Network Edge: A Survey”. *ACM Computing Survey,* vol. 54, no. 8, pp. 1-37.

Pudasaini, D. 2021. “Simulation of Video Analytic Applications Using Deep Learning”. *Proceedings of the 2021 Winter Simulation Conference S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds,* Phoenix, Arizona, pp. 1-2.

Pudasaini, D. and A. Abhari. 2020. “Scalable Object Detection, Tracking and Pattern Recognition Model Using Edge Computing”. *Spring Simulation Conference (SpringSim),* San Diego, CA, pp. 1-11.

Pudasaini, D. and A. Abhari. 2021. “Edge-based Video Analytic for Smart Cities”. *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 7, pp. 1-10.

Pudasaini, D. and A. Abhari. 2019. “Scalable Pattern Recognition and Real Time Tracking of Moving Objects,”. *Spring Simulation* Conference *(SpringSim)*, Tucson, Arizona, pp. 1-11.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi. 2016. “You Only Look Once: Unified, Real-Time Object Detection”. Retrieved from: https://arxiv.org/pdf/1506.02640.pdf.

Redmon, J. and A. Farhadi. 2018. “YOLOv3: An Incremental Improvement”. Retrieved from: https://pjreddie.com/media/files/papers/YOLOv3.pdf.

Sharifi M., A. Abhari, and S. Taghipour. 2021. “A Queueing Model for Video Analytics Applications of Smart Cities”. *Winter Simulation Conference (WSC),* Phoenix, Arizona, pp. 1–10.

Sharifi, M., A. Abhari, and S. Taghipour. 2021. “Modeling Real-Time Application Processor Scheduling for Fog Computing”. *Annual Modeling and Simulation Conference*, Fairfax, Virginia, pp. 1–12.

Shirazi, S. N., A. Gouglidis, A. Farshad, and D. Hutchison. 2017. "The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog from a Security and Resilience Perspective". *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586-2595.

Yousefpour, A., G. Ishigaki, and J. P. Jue. 2017. “Fog Computing: Towards Minimizing Delay in the Internet of Things”. *IEEE 1st International Conference on Edge Computing*, pp. 17-24.

Zagoruyko, S. and N. Komodakis. 2016. “Wide residual networks,” *BMVC,* pp. 1–12.

AUTHOR BIOGRAPHIES

ABDOLREZA ABHARI is a Professor in the Department of Computer Science at Ryerson University and director of DSMP lab (http://dsmp.ryerson.ca). He holds a Ph.D. in Computer Science from Carleton University. His research interests include data mining, web social networks, network simulation, and distributed systems. His email address is aabhari@ryerson.ca.

DIPAK PUDASAINI is a PhD student in the Department of Computer Science at Ryerson University and Assistant Professor at Tribhuvan University, Nepal. He holds a M. Sc. in Computer Science from Ryerson University. His research interests include computer vision, data science, machine learning web service selection and simulation. His email address is dpudasaini@ryerson.ca.