QATAR UNIVERSITY

COLLEGE OF ENGINEERING

QoE-AWARE RESOURCE ALLOCATION FOR CROWDSOURCED LIVE STREAMING: A MACHINE LEARNING APPROACH

BY

FATIMA HAOUARI

A Thesis Submitted to

the Faculty of the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2019

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Fatima Haouari defended on 01/06/2019.

Dr. Aiman Erbad
Thesis/Dissertation Supervisor

Prof. Amr Mohamed
Committee Member

Prof. Mohsen Guizani
Committee Member

Dr. Tamer Elsayed
Committee Member

Dr. Hazem Hajj
Committee Member

Approved:

Abdel Magid Hamouda, Dean, College of Engineering

ABSTRACT

HAOUARI, FATIMA, Masters : June : [2019:], Masters of Science in Computing

Title: QoE-Aware Resource Allocation For Crowdsourced Live Streaming: A Machine Learning Approach.

Supervisor of Thesis: Aiman, Mahmood, Erbad.

In the last decade, empowered by the technological advancements of mobile devices and the revolution of wireless mobile network access, the world has witnessed an explosion in crowdsourced live streaming. Ensuring a stable high-quality playback experience is compulsory to maximize the viewers' Quality of Experience and the content providers' profits. This can be achieved by advocating a geo-distributed cloud infrastructure to allocate the multimedia resources as close as possible to viewers, in order to minimize the access delay and video stalls.

Additionally, because of the instability of network condition and the heterogeneity of the end-users capabilities, transcoding the original video into multiple bitrates is required. Video transcoding is a computationally expensive process, where generally a single cloud instance needs to be reserved to produce one single video bitrate representation. On demand renting of resources or inadequate resources reservation may cause delay of the video playback or serving the viewers with a lower quality. On the other hand, if resources provisioning is much higher than the required, the extra resources will be wasted.

In this thesis, we introduce a prediction-driven resource allocation framework, to maximize the QoE of viewers and minimize the resources allocation cost. First, by exploiting the viewers' locations available in our unique dataset, we implement a

machine learning model to predict the viewers' number near each geo-distributed cloud site. Second, based on the predicted results that showed to be close to the actual values, we formulate an optimization problem to proactively allocate resources at the viewers' proximity. Additionally, we will present a trade-off between the video access delay and the cost of resource allocation.

Considering the complexity and infeasibility of our offline optimization to respond to the volume of viewing requests in real-time, we further extend our work, by introducing a resources forecasting and reservation framework for geo-distributed cloud sites. First, we formulate an offline optimization problem to allocate transcoding resources at the viewers' proximity, while creating a tradeoff between the network cost and viewers QoE. Second, based on the optimizer resource allocation decisions on historical live videos, we create our time series datasets containing historical records of the optimal resources needed at each geo-distributed cloud site. Finally, we adopt machine learning to build our distributed time series forecasting models to proactively forecast the exact needed transcoding resources ahead of time at each geo-distributed cloud site. The results showed that the predicted number of transcoding resources needed in each cloud site is close to the optimal number of transcoding resources.

DEDICATION

"For my family, for their constant support, love, and faith in me."

ACKNOWLEDGMENTS

First and foremost, I would like to thank the Almighty for giving me the strength and patience to work on this thesis, so that today I can stand proudly with my head held high.

I extend my heartiest gratitude to my parents. I thank my mother for her unconditional love and support, and my father for being such a great model and setting an example to me. I owe thanks to a very special person, my husband, Houssem for his continued and unfailing love, understanding and emotional support during my pursuit of Master's degree that made the completion of thesis possible. He was always around at times I thought that it is impossible to continue; he gave me the extra strength and motivation to get things done. I deeply appreciate his confidence in me and for always showing me how proud he is. I appreciate my kids, Adel and Mariam for tolerating my being preoccupied and the patience they showed during this journey. Words would never be enough to express how grateful I am to all of you. I consider myself the luckiest in the world to have such a lovely and caring family, standing beside me with their love and unconditional support.

I would like to sincerely thank my thesis supervisor, Dr. Aiman Erbad, for believing in me and pushing me to always give my best. I would also like to thank Prof. Amr Mohamed, and Prof. Mohsen Guizani for their valuable comments and encouragement all the time. My deepest thanks to Dr. Emna Baccour for her valuable guidance, timely suggestions, and support throughout the thesis journey.

This contribution was made possible by NPRP grant 8-519-1-108 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1: INTRODUCTION

1.1 Overview and Motivation

Crowdsourced live video streaming where users broadcast their captured live videos is on the rise, and it continues to grow every single day. As per Cisco mobile video traffic statistics, mobile video content is predicted to present 82% of the global Internet traffic in 2021 compared to 73% in 2016[1]. The rise in popularity of crowdsourced live streaming can be attributed to technological advancement, proliferation of smartphones and wireless network availability, which have led crowdsourcers to broadcast their live videos to various content providers.

One of the most popular video live streaming platform is Facebook, which had 2.19 billion active users per month in the first quarter of 2018[2]. 78% of Facebook online users are watching live videos, and 1 out of 5 videos on Facebook is live[3].

The industry and academia have shown an overwhelming interest in crowdsourced streaming recently in terms of achieving the best Quality of Experience (QoE) as it is the key to increase the audiences' number and the content providers' revenues. A series of recent studies have been conducted to determine the main factors that affect the viewers' QoE [1] [2]. These studies revealed that viewers QoE is primarily dependent on two factors: first, the video startup delay and playback buffering stalls, and second, the video quality which depends on the viewers' internet connectivity quality and available video representations. They also showed that viewers who experienced low QoE are less likely to revisit the content provider's application within

[1]https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visualnetworking-index-vni/mobile-white-paper-c11-520862.html
[2] https://www.statista.com/statistics/264810/number-of-monthly-active-facebook
[3] https://www.wordstream.com/blog/ws/2017/11/07/facebook-statistics

a specific period of time. Therefore, video startup, rebuffering delays and low video quality have high impact on viewers' QoE. However, the challenge is to serve the viewers with the best QoE possible, while minimizing the cost of the resources allocated.

Cloud computing is a powerful technology in terms of offering elastic and cost-effective computing resources [3] for live streaming applications. In fact, geo-distributed cloud live video applications can benefit from on demand resource renting, where cloud instances can be paid on hourly basis without upfront payment or any long term commitment [4]. The challenge is that live streaming applications have strict video startup delay requirement, including transcoding and streaming delay, while in fact it takes time for a cloud instance to be activated. As per He et al. [4] experiments, it takes two minutes for an Amazon EC2 cloud instance to boot up and function. Therefore, to minimize the initialization delay, a certain amount of cloud instances should be pre-rented for the upcoming time frame, where upfront payment can be made for a long-term reservation [4]. Moreover, various cloud providers offer up to 75% discount for reserving cloud instances proactively as opposed to on demand cloud instances pricing[4]. The challenge of cloud instances pre-renting is that resources can be insufficient to transcode all the videos into viewers matching video quality requests, or over-provisioned, which may lead to significant additional costs to the service providers.

Many pioneer works have been done on optimizing crowdsourced live videos on resource allocation for geo-distributed clouds to maximize the QoE. He et al. [4] introduced a dynamic programming approach for transcoding resources scheduling to

[4] https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/

minimize the cost and maximize the QoE. K. Bilal et al. [5] proposed a QoE-aware resource allocation framework for multiview crowdsourced live streaming to choose the optimal cloud site locations for transcoding. Wu et al. [6] formulated an optimal viewing request distribution in the geo-distributed clouds; they predicted users future demands based on their social influences using an epidemic model. The drawback of these traditional algorithms is the near optimal solutions they provide. First, they assume that the popularity of the videos is known at the start of the video streaming based on the number of views which is not the case as the number of views can be determined at the end of streaming only. Second, they assume that the viewers are in one region near the broadcaster' region, however the videos' viewers are usually geographically distributed. So they lack the ability to allocate the needed resources beforehand. This may either lead to over-provisioning of resources that may incur significant costs to the service providers, or under-provisioning of resources that may cause delays to the viewers. Therefore, addressing such a trade-off proactively is a real challenge that requires some accurate prediction techniques. Moreover, these works considered on demand renting of cloud instances, which is not always adequate for live streaming systems due to the startup time needed to boot servers.

To the best of our knowledge, there is no research work that applied machine learning techniques for resource allocation or reservation to maximize QoE and minimize the cost. Only a few studies adopted machine learning to improve the viewers QoE, with their focus varies from dealing with the buffering and the bitrate selection [7] to determining Adaptive Bitrate (ABR) best parameters in order to improve adaptive video streaming [8]. Petrangeli et al [7] proposed a video freeze predictive model to detect possible factors that lead to video stalling at the viewers' side. A recent study by Le et al. [8] proposed using decision trees to choose the best ABR parameters to

improve the adaptive video streaming.

Moreover, few recent studies have used machine learning for predicting the viewers' QoE. Zhu et al. [9] predicted the users' engagement score, by considering users engagement as a function of Quality of Service (QoS) factors and viewers preferences. Balachandran et al. [1] proposed a classification model for users' engagement, where users' engagement was quantified in terms of users' number of visits and video watching time.

1.2 Thesis objectives and contributions

The aim of this research is to first, tackle the problem of predictive-driven resource allocation to maximize the content providers profit and the viewers QoE. Specifically, we aim to predict the video popularity of each live video at the start of the live streaming, in order to allocate the live videos replicas at the proximity of the viewers to minimize the access delay.

Second, we study the problem of proactive transcoding resources reservation to minimize the network system cost and maximize the QoE. Specifically, we aim to predict the number of computational cloud instances needed for transcoding at each geo-distributed cloud site, in order to reserve them in advance, and consequently minimize the access delay and maximize the content providers' profit.

The research questions we aim to study in this thesis are the following:

1. How to decide the popularity of the videos at the start of live streaming?
2. How to minimize the system cost while guaranteeing serving distributed viewers from their proximity around the world?
3. How to reserve the exact number of transcoding resources in advance to minimize the cost and delay without over or under provisioning of resources?

The contributions of this thesis are as follows:

- We address the problem of predicting the live videos popularity at the start of live streaming. In particular, we consider predicting the number of viewers near each geo-distributed cloud site for each incoming live video, in order to proactively allocate resources at the proximity of the viewers.
    1. Using Facebook 2018 live videos dataset[5] containing records of viewers' locations for each video, we develop a regression model using machine learning techniques that predicts the number of viewers near different geo-distributed cloud sites for each incoming live video.
    2. To serve the predicted viewers such that they experience the minimum startup delay with a minimal cost to the content provider, we formulate an optimization problem for allocating resources as close as possible to the viewers.
- We present a proactive distributed transcoding resource reservation framework. First, we consider an offline resource allocation optimization that uses past incoming videos to decide the optimal number of transcoding cloud instances at each cloud site, while respecting the latency and requested video bitrates constraints. Second, based on the optimization decisions on past data, we adopt machine learning to proactively forecast the needed computational resources at each cloud site for the next time frame.
    1. We preprocess Facebook 2018 live videos dataset [10] containing records of viewers' locations to calculate the number of viewers for each video bitrate representation near different geo-distributed cloud sites.

[5] https://sites.google.com/view/facebookvideoslive18/home

2. We develop an offline resource allocation optimization for allocating transcoding resources, and serving the viewers from their nearest cloud site, with the objective to minimize the overall system cost while maximizing the viewers' QoE.
3. Based on the optimizer resource allocation decisions on historical live videos, we create our time series datasets containing historical records of the rented resources at each geo-distributed cloud site.
4. To proactively reserve the exact transcoding resources for incoming live videos, we adopt machine learning to build our distributed time series resources forecasting models.

1.3 Thesis overview

This chapter provided an overview of the research problem, and presented the thesis objectives and contributions. We organize the remainder of the thesis as follows. We introduce the main concepts used in our work, and we review the related works in chapter 2. Our prediction-driven resource allocation framework system model and evaluation results are presented in chapter 3. We present our resources forecasting and reservation framework system model and evaluation results in chapter 4. Finally we conclude and present possible future work in chapter 5.

CHAPTER 2: BACKGROUND/RELATED WORK

In this chapter, we will review some of the related works on resource allocation for crowdsourced live streaming along with some background on the concepts related to this thesis.

2.1 Background

2.1.1. Crowdsourced Live Streaming

In 2005, crowdsourcing was introduced by Meriam-Webster [4], where content providers' resources are collected from crowds of users instead of suppliers or employees. Over the past six years, crowdsourced live streaming have emerged, with various content providers, such as Facebook, YouTube, Periscope and Twitch, to name a few. In such applications, users broadcast their captured live videos, e.g., online game scenes or live events, to the data center. The received live videos will be encoded in the data center, and optionally transcoded into various bitrate representations [11]. In fact, every live video may have multiple versions with various resolutions and bitrates [4] in order to maximize the viewers' satisfaction.

2.1.2. Quality Of Experience (QoE)

The industry and academia have shown an overwhelming interest in crowdsourced streaming recently in terms of achieving the best QoE as it is the key to increase the audiences' number and the content providers' revenues. A series of recent studies have been conducted to determine the main factors that affect the viewers' QoE [1] [2]. These studies revealed that viewers' QoE is primarily dependent on two factors: first, the video startup delay and playback buffering stalls, and second, the video quality which depends on the viewers' internet connectivity quality and available video bitrate representations. Krishnan et al. [2] highlighted that the higher the startup delay is, the more the viewers' abandonment increases.

They also showed that viewers who experienced high startup delay are less likely to revisit the content provider's application within a specific period of time. Therefore, video startup and rebuffering delays have high impact on viewers' QoE. Moreover,

Viewers' devices heterogeneity on their screen resolutions, computational and bandwidth capacity demands transcoding the original live video into multiple bitrates such as 240p, 360p, 480p and 720p. To handle this heterogeneity, most video service providers deployed Adaptive Bitrate (ABR) and lately dynamic adaptive streaming over HTTP (Dash) [12]. Video transcoding is a computationally expensive process where generally a single cloud instance needs to be reserved to produce one single video bitrate representation [4]. The higher the number of viewers' demands, the more cloud instances will be used. Therefore, not all live videos are transcoded into all video bitrate representations, leading to lower QoE.

2.1.3. Elastic cloud computing

Cloud computing is a powerful technology in terms of offering elastic and cost-effective computing resources [3] for live streaming applications. In fact geo-distributed cloud live video applications can benefit from on demand resource renting, where cloud instances can be paid on hourly basis without upfront payment or any long term commitment [13]. The challenge is that live streaming applications have strict video startup delay requirement, including transcoding and streaming delay, while in fact it takes time for a cloud instance to be activated. As per He et al. [4] experiments, it takes two minutes for an Amazon EC2 cloud instance to boot up and function. Therefore, to minimize the initialization delay, a certain number of cloud instances should be pre-rented for the upcoming time frame, where upfront payment can be made for a long-term reservation [13]. Moreover, various cloud providers offer up to 75% discount for reserving cloud instances proactively as opposed to on demand

cloud instances pricing [14].

The challenge of cloud instances pre-renting is that resources can be insufficient to transcode all the videos into viewers matching video quality requests, or over-provisioning, which may lead to significant additional costs to the service providers.

2.1.4. Time series forecasting

A time series is a set of observations $o_{1,}\ o_{2,}\ o_{3,\ldots,}\ o_N$, each one being recorded at a specific time t [15]. Time series forecasting is predicting future values o_{N+h} at time N for h steps, Given that h is the lag time or forecasting horizon. Both the time the forecasting is made and the forecasting horizon should be specified before forecasting [16].

2.1.5. Machine learning algorithms

2.1.5.1 Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP), is a feed forward multilayer artificial neural network which is based upon the Back Propagation rule [17] . This learning rule applies the extended gradient-descent technique. A neural Network traverses through two phases. Firstly, the forward pass phase, where outputs are computed and their error from the expected output is measured. Secondly, it goes through the backward pass, where the error calculated in the previous phase is used to adjust the associated weights. This is achieved by implementing the backpropagation algorithm and helps in minimizing the error. This two-step process undergoes repeated iterations during the training process until an acceptable error rate is reached. The multilayer perceptron is a set of of simple interconnected neurons as illustrated in Fig. 1.

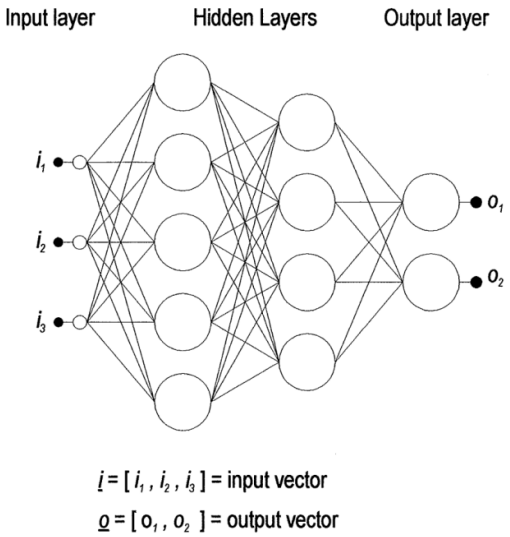

Figure 1. MLP architecture [17].

2.1.5.2 Decision Trees (DT)

Decision trees are a well-established machine learning technique. It is a tree-like model for classification or regression problems. Each node in the tree splits a data set into subsets while the decision tree is incrementally developed. In fact, the resulting tree will contain decision nodes and leaf nodes. A decision node has two or more edges. A Leaf node represents a decision or classification.

2.1.5.3 Random Forest (RF)

RF is an efficient machine learning algorithms, which has been successfully used, and proved to produce great results for many classification and regressions tasks. RF is an ensemble of DTs. In fact, RF builds various DT models then merges them to produce more accurate predictions [18].

2.1.5.4 XGboost

XGboost, extreme gradient boosting is a distributed implementation of the gradient boosting decision tress [19]. Boosting is an ensemble method where new models are added sequentially to predict the residuals or errors of existing models and then added

together to make the final prediction. The process of adding models will continue until no extra improvement can be achieved. XGboost is widely used by data scientists and provides state-of-the-art results on many classification and regression problems [19].

2.1.5.5 Long Short Term Memory (LSTM)

LSTM is a deep learning algorithm that has time-varying inputs and outputs. The core idea of an LSTM network is that the hidden neuron is treated as a memory unit [20] that can maintain the temporal state. The LSTM memory unit is comprised of three gates namely, input gate i, forget gate f, and output gate o as illustrated in Fig. 2. In fact, the LSTM basic process can be expressed as follows [21]: First, the new input information will be stored to the memory unit if the input gate is activated. Second, if the forget gate is activated, it will forget the past unit status. Finally, once the output gate is activated, it will propagate the final state.

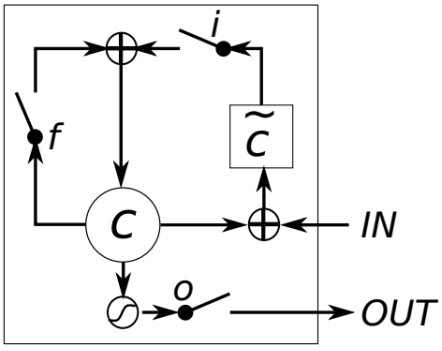

Figure 2. LSTM memory unit [22].

2.1.5.6 Gated Recurrent Unit (GRU)

Like LSTM, GRU has a memory unit. However, it has only two gates namely, the reset and update gate illustrated as r and z respectively in Fig. 3. Its memory content

is fully exposed at each time step. Moreover, the previous and the current memory content are balanced using leaky integration [22].

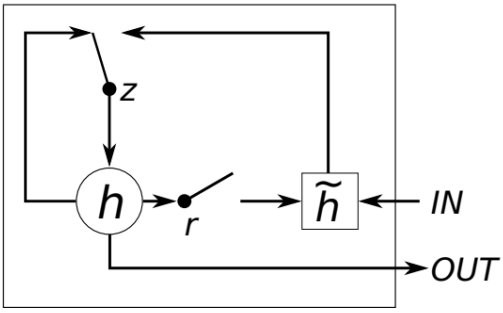

Figure 3. GRU memort unit [22].

2.1.5.7 Convolutional Neural Network (CNN)

Convolutional neural networks (CNN) are a special case of the neural network. It is a non-fully-connected neural network that consists of multi convolutional layers, RELU layer, pooling layers, which are followed by multi connected layers as in a regular neural network [23].

2.2 Related works

2.2.1 Geo-distributed clouds to maximize QoE

Live streaming applications are highly dynamic applications that demand strict video startup delay requirements. Therefore, it is challenging to design a cost-effective live streaming application. Geo-distributed clouds are proposed to support large-scale live streaming applications to minimize the overall system cost and to enhance the QoE [4] [5] [6] [24]. As they provide a cost-effective solution by offering on-demand cloud resources that meets the ever-increasing demands of bandwidth and storage, and by serving on the fly frequent viewer's requests.

2.2.2 Resource allocation to maximize QoE

Many pioneer works have been done on optimizing crowdsourced live streaming on

resource allocation for geo-distributed clouds to maximize the QoE. He et al. [4] presented a resource allocation framework to allocate geo-distributed cloud service to crowdsourcers for transcoding and serving viewers, by proposing a cloud rental strategy. They proposed a cloud rental approach based on dynamic programming, where they took into consideration the limitation of elastic cloud supply in each geo-distributed cloud. They further extended their experiments by proposing a heuristic that pre-rank the cloud instances in advance, in order to achieve faster running time.

K. Bilal et al. [5] presented a QoE-aware resource allocation framework for crowdsourced multiview live streaming to choose the optimal transcoding cloud site location, and the optimal set of video representations. They first formulated the resource allocation as an optimization problem. However, because of the size and complexity of the problem, they introduced a greedy heuristic that proved to achieve a near optimal solution. Chen et al. [24] introduced a cost-effective framework for cloud resources provisioning to cope with geo-distributed video broadcasters. They built a prototype for crowdsources live streaming using Amazon Cloud and Microsoft Azure to evaluate their system, which proved to be effective in terms of cost and streaming quality. However, the mentioned studies assumed that the popularity of the videos based on the number of views is known at the start of the video live streaming, which is not the case in reality. Moreover, they assume that the viewers are in one region near the broadcaster' region, but the video' viewers are usually geo-distributed. In our work, we exploit the viewers' locations in our dataset to map the viewers of each video into Amazon geo-distributed cloud sites. Then, we predict the popularity of each video at the start of streaming by predicting the number of viewers near each geo-distributed cloud site.

Wu et al. [6] formulated an optimal viewing request distribution in the geo-distributed

clouds, they predicted users future demands based on their social influences using an epidemic model. They then served the predicted viewers by introducing one-shot optimization. They evaluated their system performance using Amazon Elastic cloud computing (EC2). The results proved that their method outperforms some heuristics algorithms.

2.2.3 Machine learning to maximize QoE

Only a few studies adopted machine learning to improve the viewers QoE, with their focus varies from dealing with the buffering and the bitrate selection [7], to determining Adaptive Bitrate (ABR) best parameters in order to improve adaptive video streaming [8]. Petrangli et al. [7] proposed machine learning based framework to prevent video stalling to maximize the viewers QoE. As part of their framework, they introduced a video freeze predictive model to detect possible factors that lead to video stalling at the viewers' side. A recent study by Le et al. [8] proposed adaptation of existing decision trees algorithms to choose the best ABR parameters, in order to improve the adaptive video streaming. The authors showed that the performance of an ABR algorithm can be improved by 8.59% by applying their approach.

Moreover, few recent studies have used machine learning for predicting the viewers' QoE. Zhu et al. [9] predicted the users' engagement score, by considering users engagement as a function of Quality of Service (QoS) factors and viewers preferences. They then formulated an optimization problem to map the users to content delivery networks (CDN) in order to maximize the QoE.

Balachandran et al. [1] proposed a classification model for users' engagement, where users' engagement was quantified in terms of users' number of visits and video watching time. Their predictive model is useful to handle the video delivery mechanisms for the content providers and the video player designers. To our

knowledge, no prior studies examined machine learning for resource allocation to maximize QoE for crowdsourced live streaming. Our contribution in this context is to first, predict the popularity of the videos at the start of the live streaming in order to allocate videos replicas at the proximity of the viewers to maximize their QoE. Second, forecasting the transcoding resources ahead of time at each geo-distributed cloud site to minimize the system cost and delay without over or under provisioning of resources.

CHAPTER 3: QOE-AWARE RESOURCE ALLOCATION FOR CROWDSOURCED LIVE STREAMING: A MACHINE LEARNING APPROACH

In this chapter, we address the problem of predicting the live videos popularity at the start of live streaming. In particular, we consider predicting the number of viewers near each geo-distributed cloud site for each incoming live video, in order to proactively allocate resources at the proximity of the viewers.

3.1 Contributions

1. Using Facebook 2018 live videos dataset [10] containing records of viewers' locations for each video, we develop a regression model using machine learning techniques that predicts the number of viewers near different geo-distributed cloud sites for each incoming live video.
2. To serve the predicted viewers such that they experience the minimum startup delay with a minimal cost to the content provider, we formulate an optimization problem for allocating resources as close as possible to the viewers.

3.2 System model

In our system, we adopt a geo-distributed cloud infrastructure as shown in Fig. 4 that consists of multiple geographically distributed cloud sites. Our predictive model and resource allocation optimizer are deployed in a centralized master server. A set of geo-distributed crowdsourcers broadcast their videos in real time, which will be allocated by default in their nearest cloud site. Each broadcaster cloud site will report the master server with the incoming live videos information. The predictive model will predict the number of viewers expected near each cloud site. Based on the predicted results, the optimizer will allocate live videos replicas across the geo-distributed cloud sites

near the viewers' proximity to minimize the delay and video stalls with the minimum possible cost. Moreover, the optimizer determines from which cloud site the viewers should be served.

In our work, we consider only the storage resources, while the computation resources for video transcoding are not considered in this framework.

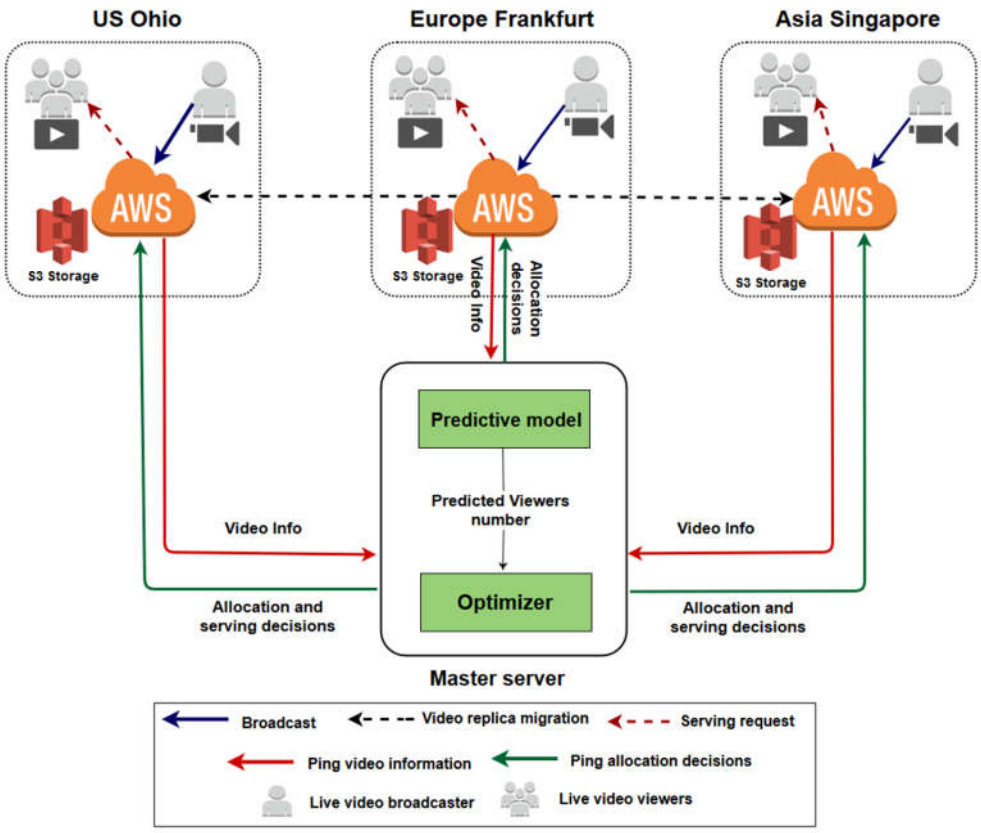

Figure 4. Proactive resource allocation system model.

3.2.1 Predicting live videos viewers

3.2.1.1 Dataset

In our work, we are using the Facebook 2018 live videos dataset [10], containing more than two million Facebook live video streams. The active video streams metadata are

fetched every 3 minutes in different periods on January, February, March, May, June and July 2018. As a result, we obtained a list of fetches related to each video and containing the number of viewers at the recording time. The live videos are collected with many features such as creation time and date, broadcaster location, number of likes and most importantly the viewers' locations. In this work, we selected six features for each video namely, the broadcaster name, content category, created time, created day, broadcaster location and the viewers' locations as illustrated in Fig. 5. The viewers' locations were selected from the video fetch with maximum number of viewers.

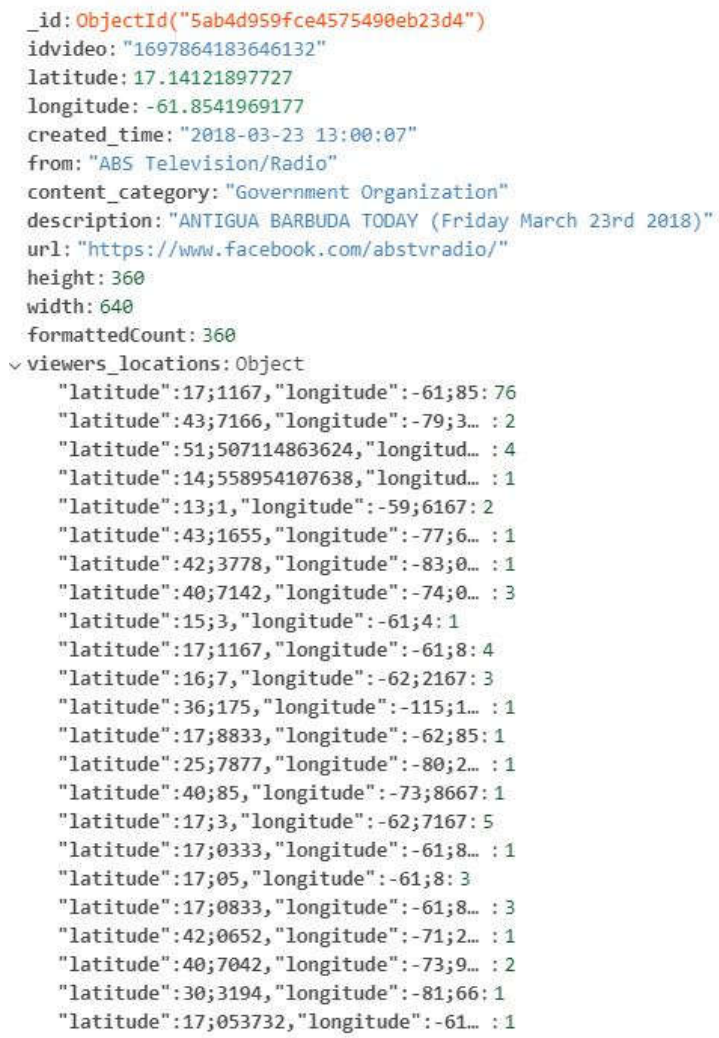

Figure 5. A live video stream metadata/features

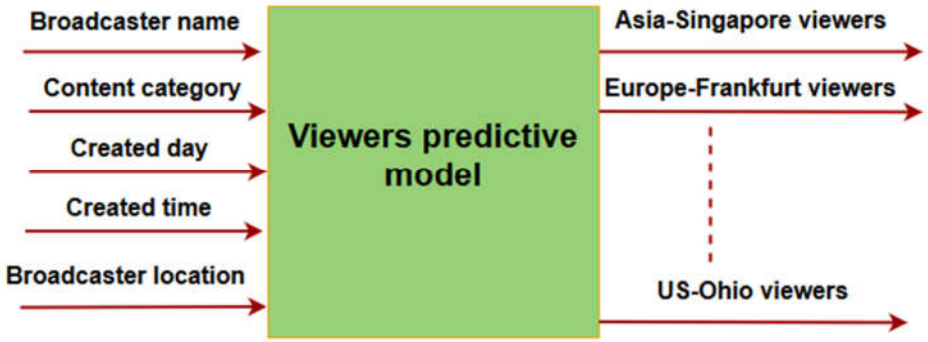

Figure 6. Viewers predictive model.

3.2.1.2 Preprocessing

As our objective is to predict the viewers' number near various geo-distributed cloud sites, there was a need to preprocess our raw data. First, as illustrated in Figure 7, we mapped the viewers' locations into 10 Amazon Web Services (AWS) cloud sites locations[6] namely, Asia-Mumbai, Asia-Seoul, Asia-Singapore, China-Ninxgia, Europe-Frankfurt, Europe-Paris, South America-Sao paulo, US East-Ohio, US East-Virginia and US West-California. This was done by calculating the shortest distance between the viewer's locations and the 10 AWS cloud sites location[7]. Furthermore, we calculated the number of viewers near each cloud site for each video. We did the same to the broadcaster location, where we mapped his location into the nearest AWS cloud site.

Moreover, we clustered the created time into 6 time periods. Finally, we applied the categorical one-hot encoding to the time period, created day and broadcaster location features, while we used feature hashing introduced by [25] to transform the high-cardinality features namely broadcaster name and content category into hashed feature

[6] https://aws.amazon.com/about-aws/global-infrastructure/
[7] https://github.com/turnkeylinux/aws-datacenters/blob/master/input/datacenters

vectors. An overview of how we processed our features is presented in Table 1.

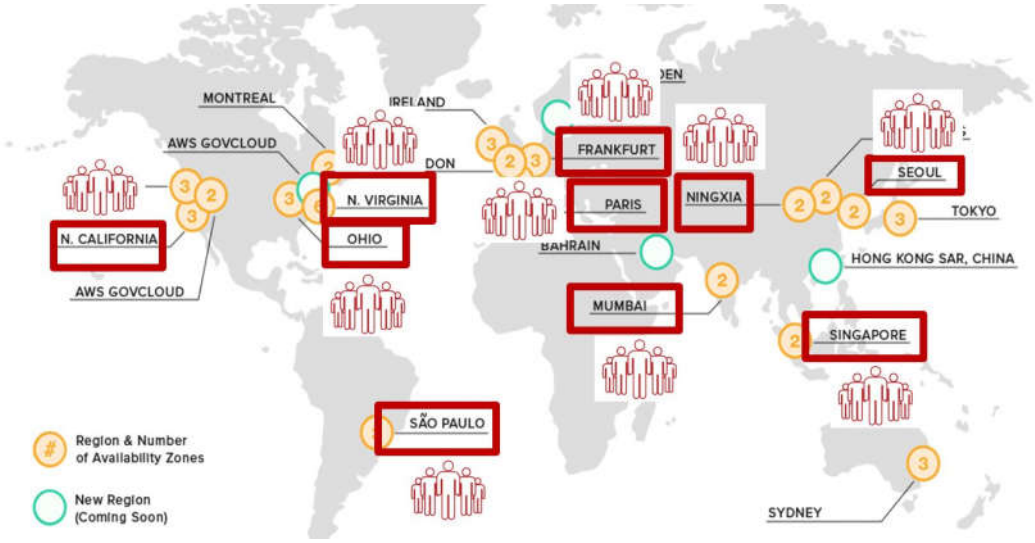

Figure 7. Mapping viewers to AWS datacenters.

Table 1. Overview Of Features For Predicting Number Of Viewers.

Feature	**Feature representation**
Broadcaster Name	Feature hashing: a vector of 6 bits
Content Category	Feature hashing: a vector of 6 bits
Created day	One hot encoding of 7 days
Created time	One hot encoding of 6 time periods
Broadcaster Location	One hot encoding of 10 locations

3.2.1.3 Predictive models

The dataset used to train our models included 224,839 live video records collected in March, May and June 2018. 80% of the records were randomly selected for training and 20% were used for validation.

We trained our regression models to produce 10 outputs as illustrated in Fig. 5, each represents the number of viewers near the 10 AWS cloud sites mentioned previously.

We adopted three different ML algorithms namely, Multilayer-perceptron (MLP),

Decision trees (DT) and Random Forest (RF).

We used Scikit-learn python library[8] to build several models using each ML algorithm, as there is no method to predetermine the best combination of hyperparameters, such as the number of hidden layers and neurons for MLP models, number of forests for RF models and the max depth for DT models. Finally, the best models were selected by a grid search approach considering the best determination coefficient R^2 values, which is used to assess the goodness of fit of our regression models. R^2 values approaching 1 indicate that the model provides accurate predictions, and it is calculated according to Eq.1 :

$$R^2=1-\frac{\sum_{i=1}^{m}(A_i - P_i)^2}{\sum_{i=1}^{m}(A_i - \bar{A})^2} \qquad \text{Eq. 1}$$

Where m is the number of videos, A_i is the actual number of viewers for video i, P_i is the predicted number of viewers for video i, and $\bar{A}$ is the mean of the actual number of viewers of all videos. In Table 2, we present our best models hyperparameters configurations. In our MLP models we adopted the dropout proposed by [26], where some neurons are disabled during the training process to avoid overfitting.

Table 2. Best Predictive Models Configurations.

Model	**Configurations**
MLP /3 layers	Neurons: 500,1000,500 Optimizer: Adagrad, Dropout: 0.4 Loss function: Mean Square Error (MSE) Activation function: Rectified Linear Units (ReLU)
MLP/5 layers	Neurons: 500, 1000, 2000, 1000, 500 Optimizer: Adagrad, Dropout: 0.4 Loss function: MSE Activation function: ReLU

[8] https://scikit-learn.org/stable/

Model	Configurations
MLP/7 layers	Neurons: 500, 1000, 1500, 2000, 1500, 1000, 500 Optimizer: Adagrad Dropout: 0.4 Loss function: MSE Activation function: ReLU
Random Forest	Loss function: MSE Number of estimators: 50
Decision trees	Loss function: MSE Max_depth: None

3.2.1.4 Predictive models results

After training the models, the validation results, depicted in Fig. 6, showed that RF outperforms the other ML algorithms by achieving for example an R^2 of 0.91 for Seoul, 0.89 for Sao Paulo, 0.85 for Ohio, 0.86 for California and 0.74 for China. The DT model achieved the lowest R^2 as opposed to MLP and RF. The results showed that increasing the number of layers for the MLP models improves the results. However, due to the complexity of the models, and because we noticed that there is a slight difference between the performance of the 5 layers model and the 7 layers model, we did not increase the layers above 7.

The results also showed that for all ML models, the predicted number of viewers near some regions achieved a higher R^2 compared to other regions, China achieved the lowest, while Seoul and Sao paulo achieved the best R^2. This could be attributed to the fact that some regions have a higher number of videos broadcasted near to them. We further tested our models on unseen data of live videos collected from July 1 to July 6, 2018. The models performed the same as with validation data in some regions, slightly less or higher in other regions as shown in Fig. 7. We then extended our experiments by performing the predictions on hourly basis for 24 hours using the live

videos of July 3, 2018. The RF and MLP 7 layers models were used for prediction, since they performed better than other models. The predicted number of viewers for the hourly incoming live videos versus the actual number of viewers for all cloud sites are presented in Fig. 8 to Fig. 17. Since our results demonstrate that the RF predictions are the closest to the actual values, we will adopt this model in our system.

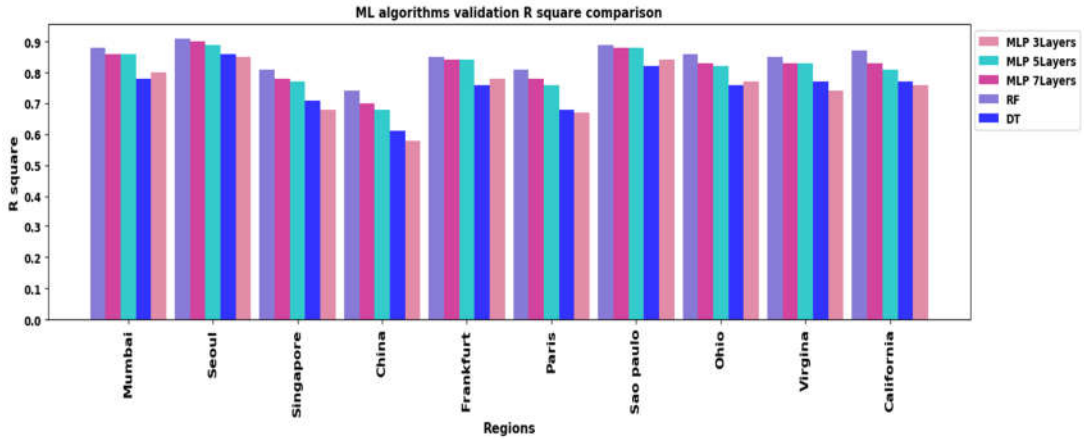

Figure 8. Predictive models R^2 validation comparisons.

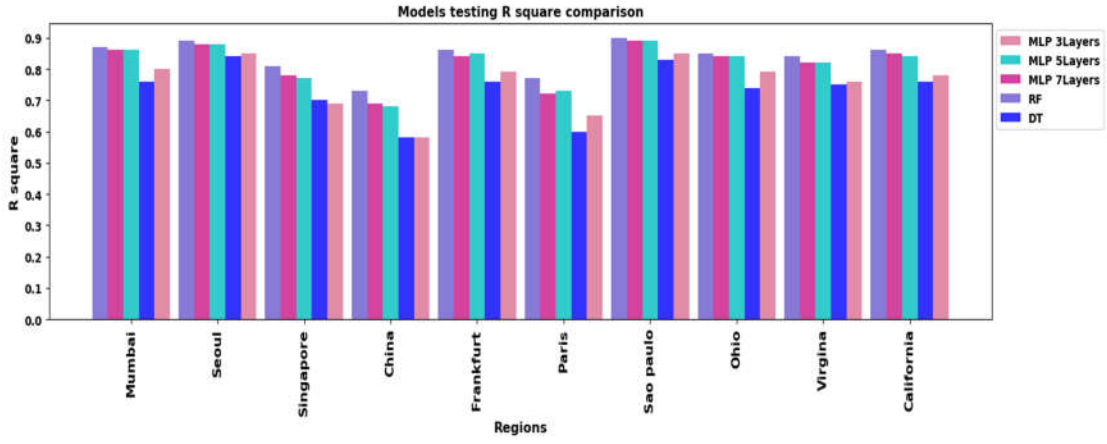

Figure 9. Predictive models R^2 testing comparisons.

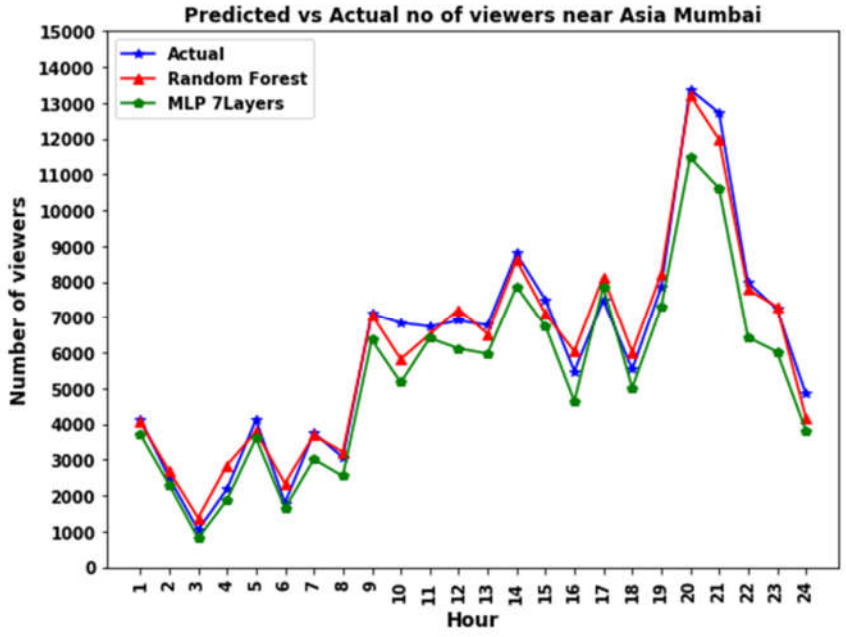

Figure 10. Asia Mumbai predicted vs actual number of viewers.

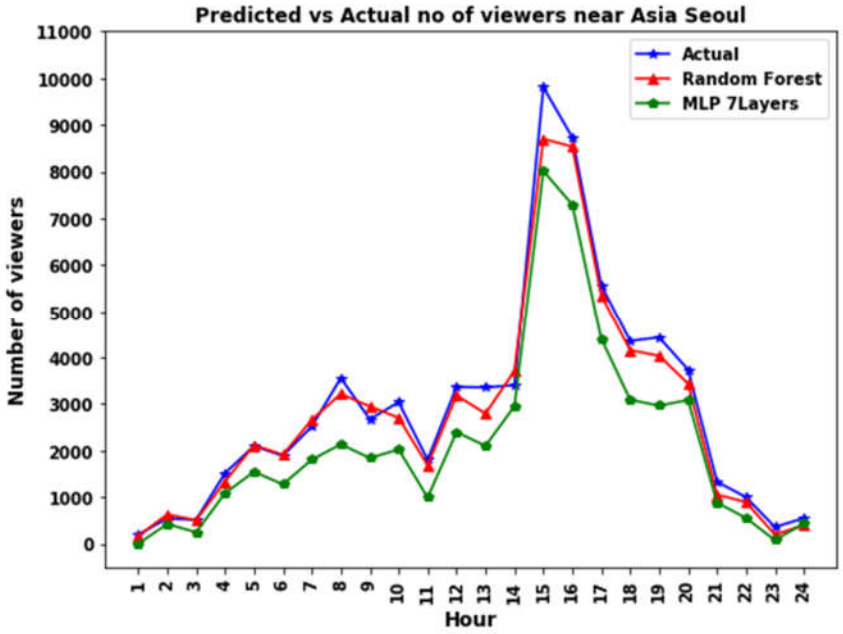

Figure 11. Asia Seoul predicted vs actual number of viewers.

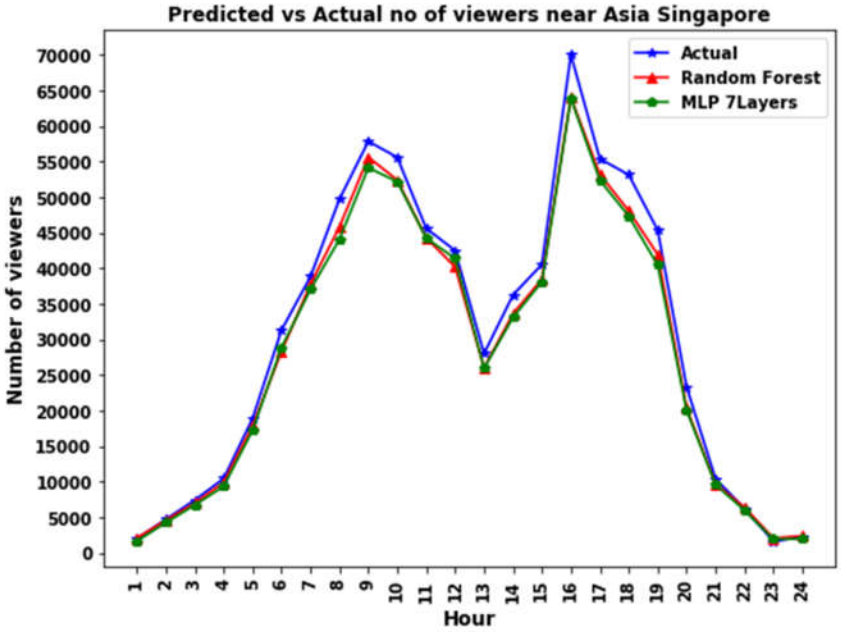

Figure 12. Asia Singapore predicted vs actual viewers number.

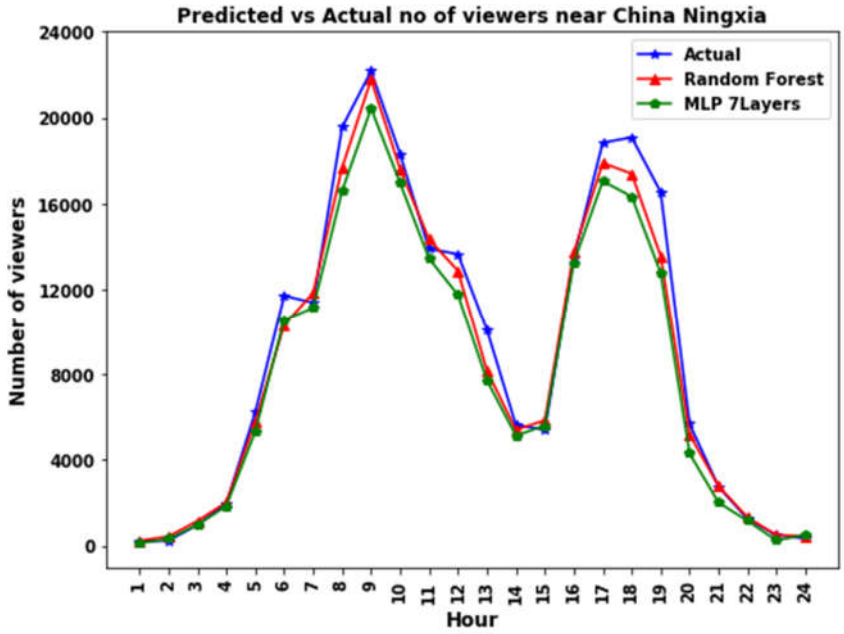

Figure 13. China Ningxia predicted vs actual viewers number.

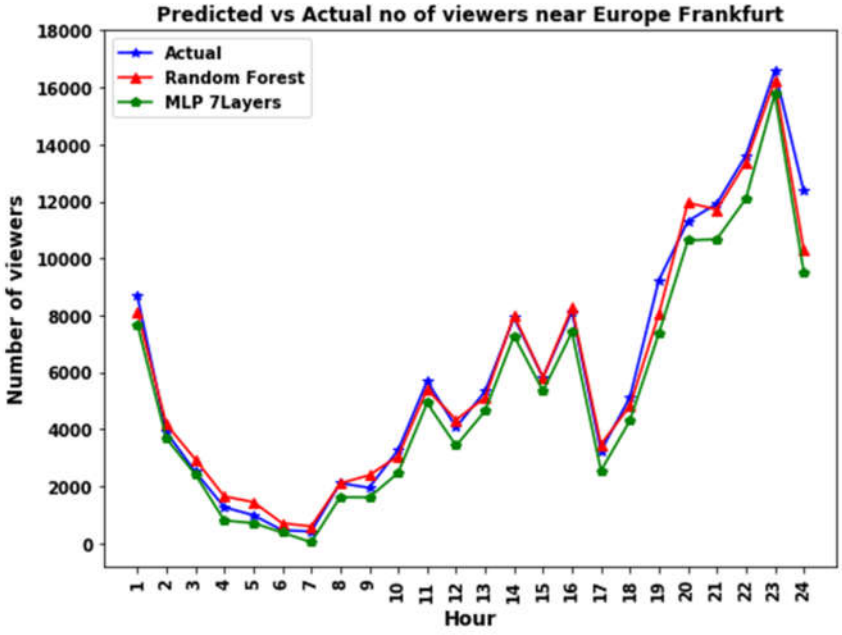

Figure 14. Europe Frankfurt predicted vs actual viewers number.

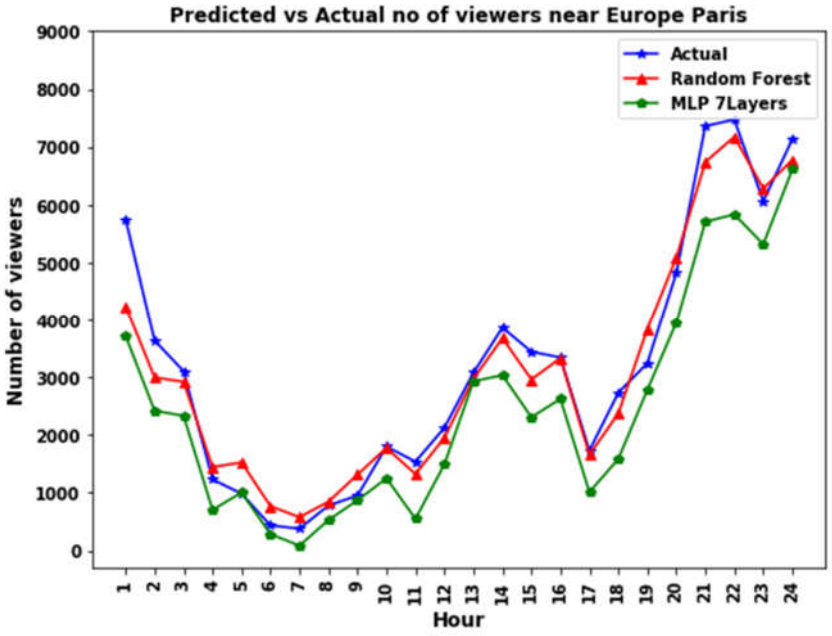

Figure 15. Europe Paris predicted vs actual viewers number.

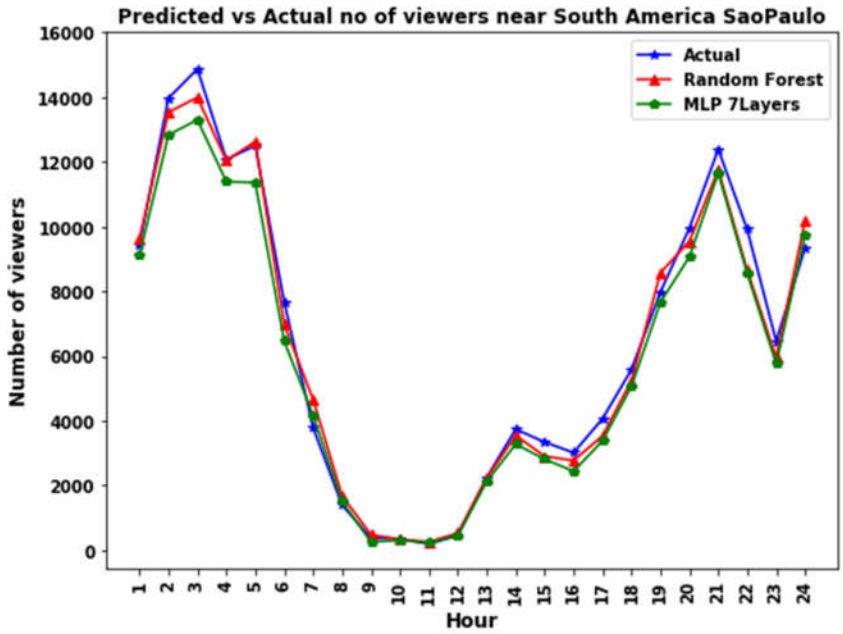

Figure 16. South America Sao Paulo predicted vs actual viewers number.

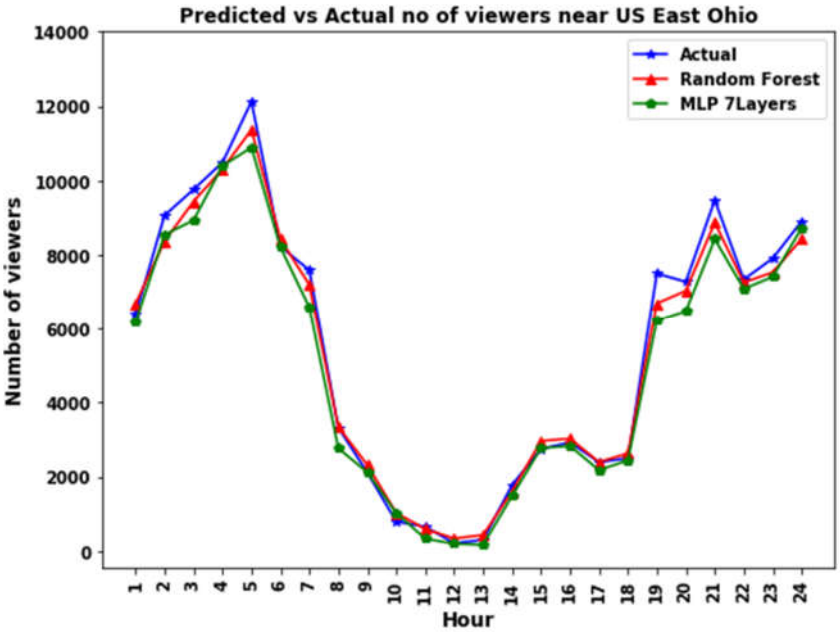

Figure 17. US East Ohio predicted vs actual viewers number.

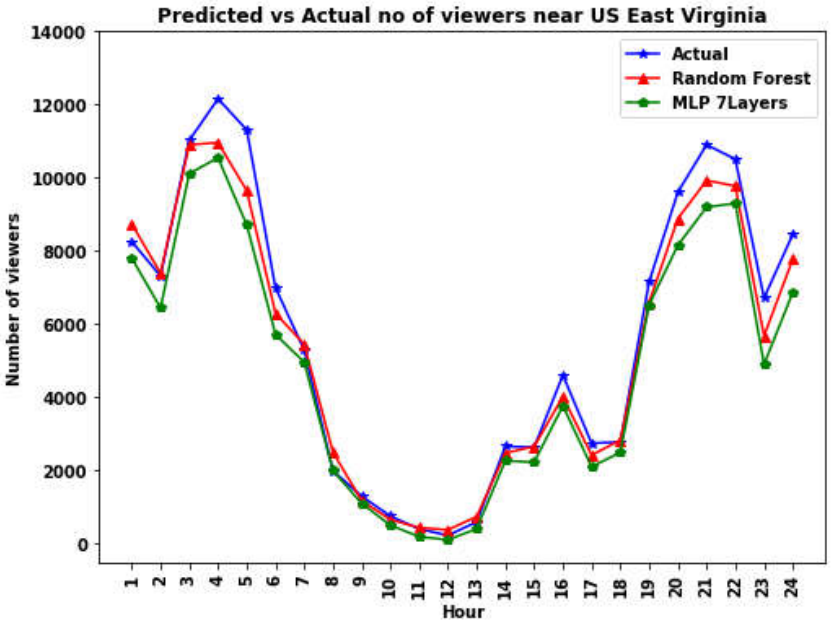

Figure 18. US East Virginia predicted vs actual viewers number.

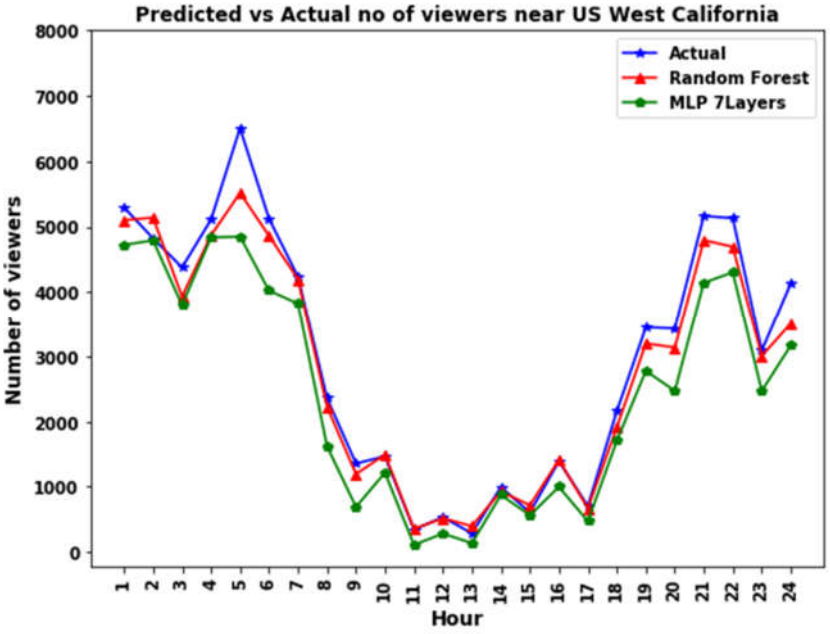

Figure 19. US West California predicted vs actual viewers number.

3.2.2 Proactive live video allocation and viewers serving

In this section, we formulate the problem of proactive resource allocation, to derive the optimal number of video allocation cloud sites and the nearest cloud site to serve the viewers, with an objective of minimizing the cost constrained by the access delay. We then, present our proactive resource allocation algorithm.

3.2.2.1 Problem formulation

The set of incoming videos at period t is denoted by V (t) = $\{v_1, v_2, v_3, \ldots v_m\}$. The set of regions is represented by R =$\{r_1, r_2, r_3, \ldots, r_n\}$. Let r_b, r_a and r_w denote the broadcasting region, video allocation region and video serving region respectively. The round trip delay from r^a to r^w is represented by $d_{r^a r^w}$.

Let P(t) = $\{P_{v_1}, P_{v_2}, P_{v_3}, \ldots\ldots, P_{v_m}\}$ represent the set of predicted viewers for the incoming videos at period t . As each video has predicted viewers in different regions, let P_v=$\{p_1, p_2, p_3, \ldots\ldots p_n\}$ denote the set of the number of predicted viewers at different regions for each video v. The broadcasters' regions for the incoming videos at period t is denoted by B(t)= $\{r_1^b, r_2^b, r_3^b \ldots\ldots r_m^b\}$.

Due to the fact that some videos do not have any viewers near some cloud sites, let E(v,r^w) present a binary variable equal to 1, if video v has predicted viewers near the region r^w, and 0 otherwise.

The decision variable A(v, r^a) is equal to 1, if video v is allocated in region r^a, and 0 otherwise. While the decision variable W(v, r^a, r^w) is equal to 1, if viewers at region r^w are served from region r^a and 0 otherwise. The problem formulation notations are presented in Table. 3.

We consider renting S3 storage[9] servers at each cloud site. Three types of costs are

[9] https://aws.amazon.com/s3/pricing/

taken into account: (1) the storage cost at each cloud site; (2) the migration cost of a video replica from one cloud site to another and (3) the cost of serving viewers. We assume that the storage capacity can be provisioned based on the application demand. On allocation cloud site at region r^a, let α_{r^a} be the storage cost per GB, which varies based on site location and the storage thresholds fixed by Amazon S3. For example, Amazon charges 0.023$ per GB for the first 50TB, while it charges 0.021$ when exceeding 500TB in the case of US East Virginia region[9].

Given that K is the video size, the total storage cost $\mathbb{S}$ can be calculated as presented in Eq. 2.

$$\mathbb{S}= \sum_{v \in V(t)} \sum_{r^a \in R} \alpha_{r^a} * K * A(v, r^a) \quad \text{Eq. 2}$$

Given that η_{r^b} is the cost to migrate a copy of a video from the broadcaster region r^b to allocation region r^a, which is the data transfer cost from one cloud site to another per GB, the total migration cost $\mathbb{M}$ is calculated as presented in Eq. 3.

$$\mathbb{M}=\sum_{v \in V(t)} \sum_{r^a \in R} \eta_{r^a} * K * A(v, r^a) \quad \text{Eq. 3}$$

Given that ω_{r^a} is the serving request cost from region r^a, which is the data transfer cost from that region to the internet per GB, the total serving request cost $\mathbb{R}$ is calculated as presented in Eq. 4. The overall cost $\mathbb{C}$ to serve viewers is shown in Eq. 5.

$$\mathbb{R}=\sum_{v \in V(t)} \sum_{r^a \in R} \sum_{r^w, \in R} \omega_{r^a} * K * p_{r^w} * W(v, r^a, r^w) \quad \text{Eq. 4}$$

$$\mathbb{C}=\mathbb{S}+\mathbb{M}+\mathbb{R} \quad \text{Eq. 5}$$

Our objective is to minimize the cost for period t as shown in Eq. 6:

$$min_{A(v,r^a)\, W(v,r^a,r^w)} \mathbb{C} \quad \text{Eq. 6}$$

Subject to the following constraints:

Every video is allocated by default in the broadcaster nearest cloud site.

$$A(v, r^b)=1 \quad \forall v \in V(t), \forall r^b \in B(t) \qquad \text{Eq. 6a}$$

A video v can be served from region r^a to viewers at region r^w, only if it is allocated at region r^a .

$$W(v, r^a, r^w) \leq A(v, r^a) \quad \forall v \in V(t), \forall r^a \in R, \forall r^w \in R \qquad \text{Eq. 6b}$$

A video v can be served from region r^a to r^w only if there exists viewers at r^w.

$$W(v, r^a, r^w) \leq E(v, r^w) \quad \forall v \in V(t), \forall r^a \in R, \forall r^w \in R \quad \text{Eq. 6c}$$

If there exists viewers for video v at region r^w, they can only be served from one region.

$$\sum_{r^a \in R} W(v, r^a, r^w)=E(v,r^w) \quad \forall v \in V(t), \forall r^w \in R \quad \text{Eq. 6d}$$

The average serving request delay to serve a video v should not exceed a threshold *D*.

$$\frac{\sum_{r^a \in R} \sum_{r^w \in R} p_{r^w} * d_{r^a r^w} * W(v, r^a, r^w)}{\sum_{r^w \in R} p_{r^w}} \leq D \quad \forall v \in V(t) \quad \text{Eq. 6e}$$

Binary decision variables that can be set to 0 or 1.

$$A(v, r^a), W(v, r^a, r^w) \in \{0,1\} \qquad \text{Eq. 6f}$$

Table 3. Notations For The Formalized Problem.

Notation	**Description**
V(t)	Set of incoming live videos at period t
R	Set of regions
B(t)	Set of broadcasters regions for videos at period t
r^a, r^w, r^b	Region of video allocation, Region of serving and Region of broadcasting
P(t)	Set of predicted viewers for live videos at period t
p_v	Set of predicted viewers at different R for video v

Notation	Description
SU	Set of storage used at each region
W(v, r^a, r^w)	Binary decision variable that indicates the serving site
A(v, r^a)	Binary decision variable that indicates the allocation site
E(v,r^w)	Binary variable that indicates viewers existence
$d_{r^a r^w}$	Round trip delay between r^a and r^w
RTT	Matrix for round trip delay between the different R
D	Delay threshold
K	Video size
α_{r^a}	Storage cost per GB at region
η_{r^b}	Migration cost per GB from broadcaster region r^b
ω_{r^a}	Serving request cost per GB from r^a
S	Total storage cost
M	Total migration cost
R	Total serving request cost
C	Overall cost

3.2.2.1 Proactive resource allocation

The proposed proactive resource allocation algorithm is presented in Algorithm. 1. In fact, at each period t, the system receives a set of incoming videos, which will be an input to the viewers' predictive model. Based on the predicted viewers, the optimal number of allocation cloud sites and the nearest cloud site to serve the viewers will be decided by the optimizer presented in 3.2.2. The storage resources at each cloud site

is reserved based on the allocation decisions, and released for ended live videos from the previous periods. Moreover, the viewers are served from their closest cloud site based on the serving decisions.

Algorithm 1. Proactive resources allocation.

Algorithm 1 Proactive resources allocation

Input: R, $\{\alpha_1, ..., \alpha_n\}$, $\{\eta_1, ..., \eta_n\}$, $\{\omega_1, ..., \omega_n\}$, RTT, κ
Storage usage at each cloud site initialization: $SU_1 = 0, ..., SU_n = 0$
for $t \in \{1, .., T\}$ **do**
    - Receive videos informations $V(t)$ and their broadcasters $B(t)$
    - Run predictive model to predict $P(t)$ for videos $V(t)$
    - Derive optimal solution $\min_{A(v,r^a)W(v,r^a,r^w)}$ $\mathbb{C}$ as per Eq. 6
    for $j \in \{1, .., n\}$ **do**
        - Update SU_j based on allocation decisions $A(v, r^a)$
        - Release storage resources for ended videos from previous periods.
        - Serve viewers based on serving decisions $W(v, r^a, r^w)$

3.1 Performance evaluation

3.3.1 Simulation settings

In this section, we evaluate the performance of our system using the RF hourly predicted viewers of July 3, 2018 to get the hourly optimal resource allocation for T=24(hours) and t=1(hour). We implemented the optimization problem using Matlab in CVX solver. The number of hourly incoming videos, and the hourly predicted viewers used in our simulation are presented in Fig. 18. In our system, we assume that the video duration is 4 hours, which is the maximum video duration for a Facebook live video. We assume that if a video is allocated in a set of cloud sites at period t, it will be allocated in the same cloud sites for the remaining time periods of streaming. Moreover, because video quality is out of the scope of this thesis we assume that the viewers are served with the best video quality, where we set the video size K to 0.738 Gbit. We constructed our round trip time (RTT) matrix $d_{r^a r^w}$ by calculating the

average RTT from the different cloud sites using[10] accessed on September 19, 2018. The storage and data transfer prices of Amazon S3[9] are considered in our simulation to model α, ω and η. We varied the latency thresholds constraints *D* for serving a video to 8.8ms, 60ms, 120ms, 171ms, 220ms and 371ms. 8.8ms is the latency needed to serve a viewer from its closest cloud region [5].

3.3.2 Simulation results

Fig. 19 shows that we can establish a trade-off between the video access delay and the resource allocation cost. Indeed, the hourly optimal cost is high when the system is forced to serve the viewers from their region by setting the latency threshold to 8.8ms. Relaxing the threshold leads to minimizing the cost. Therefore, the content provider can sacrifice in terms of cost to enhance the QoE or the opposite based on her requirements. It is worth mentioning that the optimal cost is higher in some periods as opposed to others, because, as illustrated in Fig. 18, the number of incoming videos and predicted viewers varies from period to another.

In order to evaluate the total system cost over the 24 hours with various latency thresholds, we calculated the hourly total cost, as presented in Fig. 20. The hourly total cost is defined as the sum of the network cost at period t and the cost of storage of still running videos, which is presented in Eq. 7, given that SU_n is the storage usage at region n until period t.

$$\text{Hourly total cost (t)} = \mathbb{C}(t) + \sum_{r^a \in R} \alpha_{r^a} * SU_{r^a} \qquad \text{Eq. 7}$$

The system total cost is calculated as shown in Eq. 8:

$$\text{System total cost} = \sum_{t=1}^{T} Hourly\ total\ cost(t) \qquad \text{Eq. 8}$$

Furthermore, we calculated the hits percentages, which represents the percentage of

[10] https://wondernetwork.com/pings

videos served from the same region of viewers as shown in Fig. 21. Setting the latency to 8.8ms resulted in hits percentage of 100% in every hour, as all viewers will be served from their region. While it is in the range of 20% to 30% with 60ms latency threshold. Moreover, when the latency threshold was set to 120ms, 171ms, 220ms and 371ms, less than 20% of videos were served from the same region of viewers. The hits percentage was very low with high latency thresholds, as the system is not forced to serve the viewers from their closest region.

Finally, to evaluate the accuracy of our resource allocation framework, we calculated the hourly average latency using the proactive serving decisions with variant latency thresholds *D*. In fact, we calculated the latency of serving the actual number of viewers based on our proactive video allocation and we compared it to the latency derived from the predictive model. The results as shown in Fig. 22 proved that the average latency to serve the actual viewers is very close to the average latency serving the predicted viewers. Moreover, the average latency to serve the actual viewers did not exceed the latency thresholds *D*.

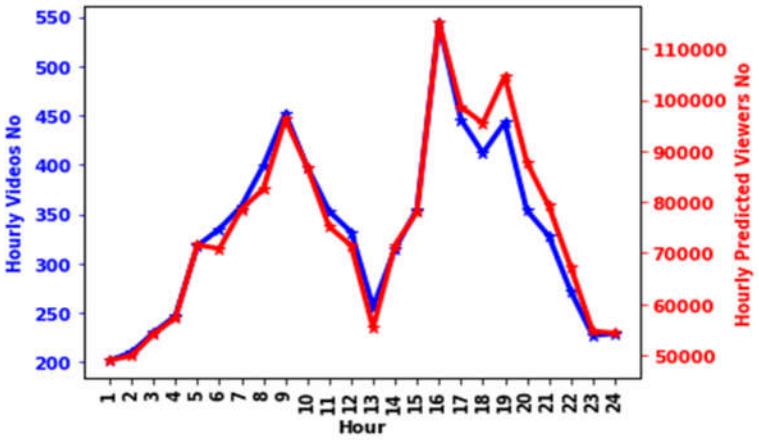

Figure 20. Hourly incoming videos/ Hourly predicted viewers.

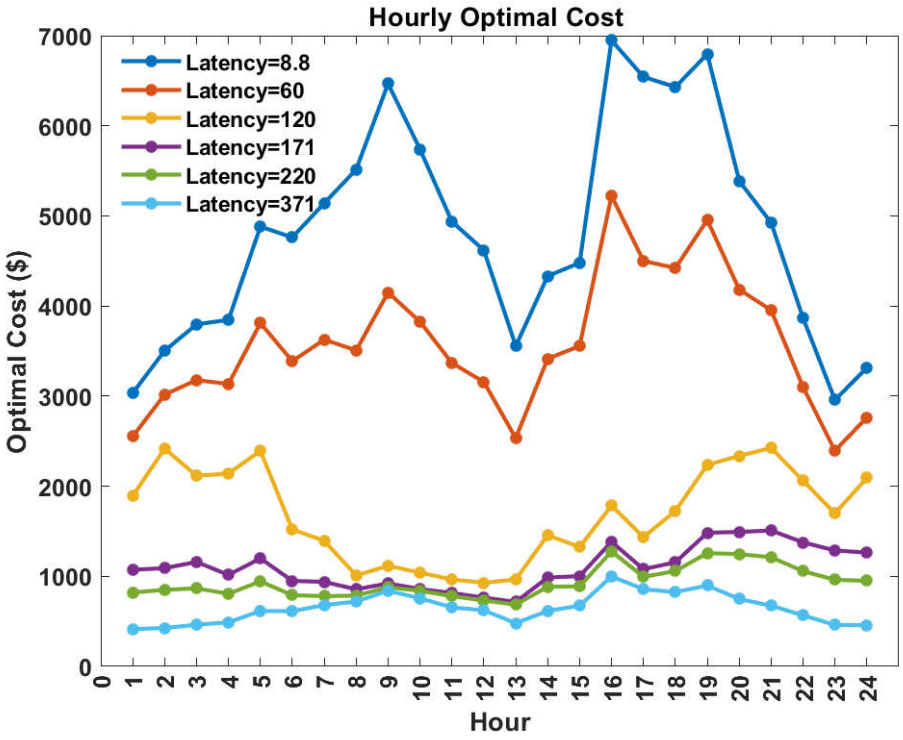

Figure 21. Hourly optimal cost.

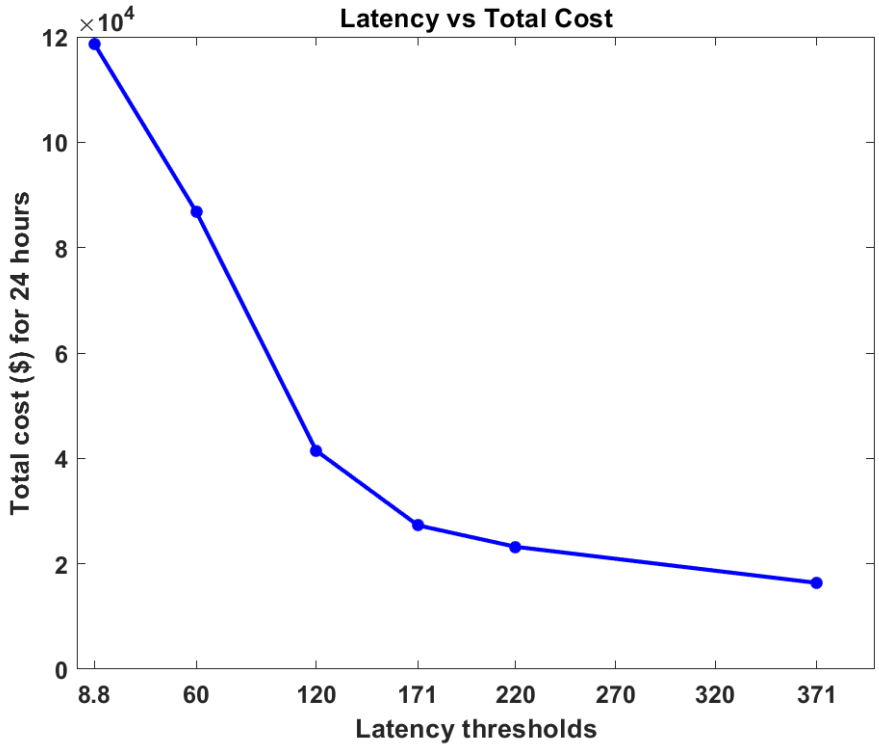

Figure 22. Total system cost vs latency threshold.

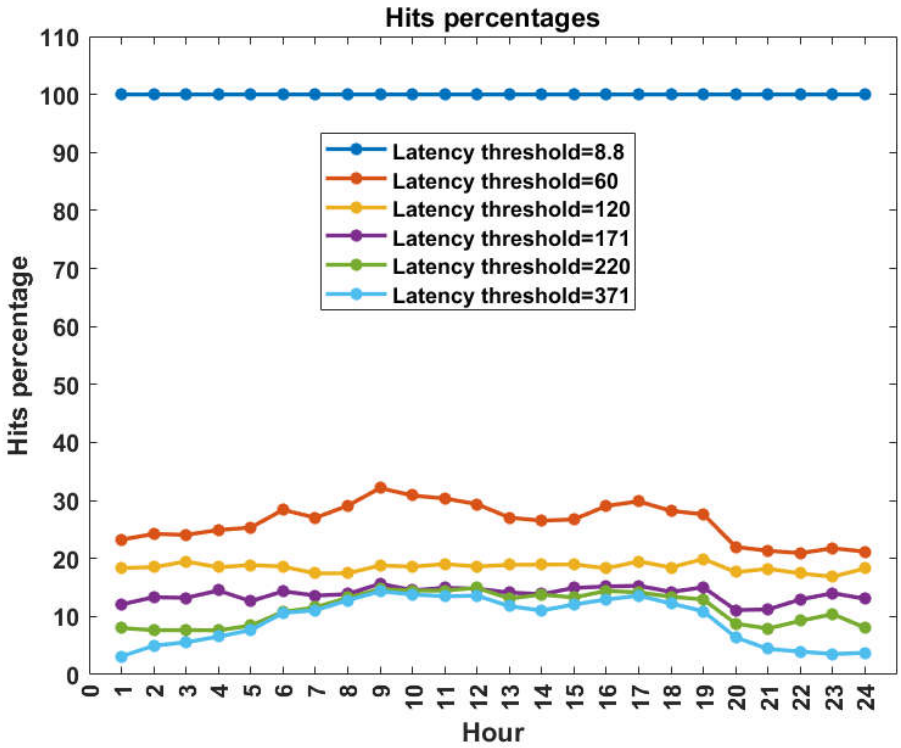

Figure 23. Serving hits percentages.

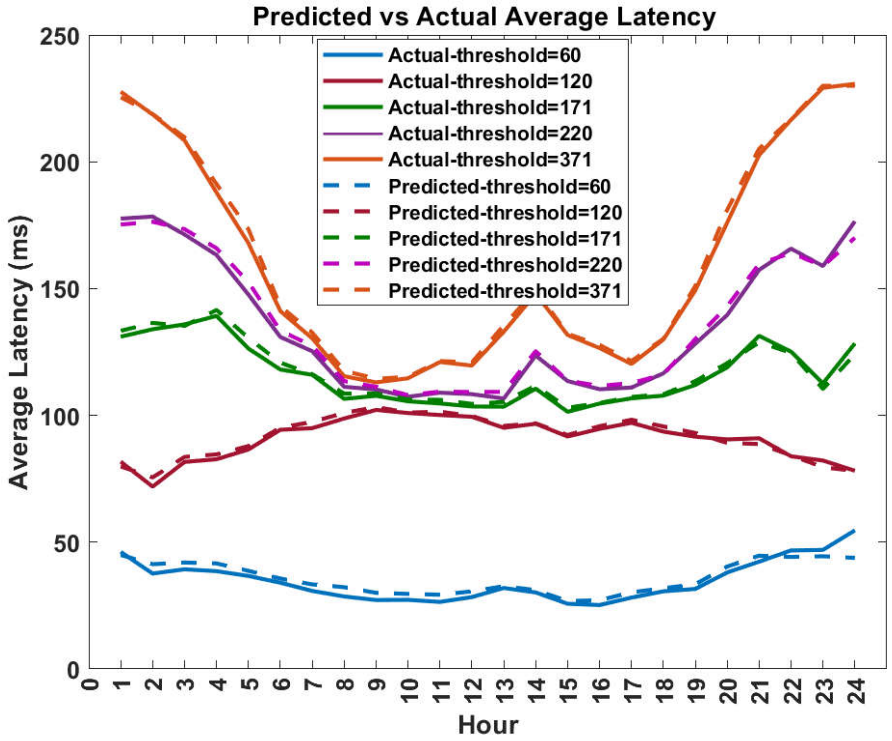

Figure 24. Predicted vs actual hourly average latency.

CHAPTER 4: TRANSCODING RESOURCES FORECASTING AND RESERVATION FOR CROWDSOURCED LIVE STREAMING

In this chapter, we present a proactive distributed transcoding resource reservation framework. First, we consider an offline resource allocation optimization that uses past incoming videos to decide the optimal number of transcoding cloud instances at each cloud site, while respecting the latency and requested video bitrates constraints. Second, we adopt machine learning to proactively forecast the needed computational resources at each cloud site for the next time frame.

4.1 Motivation

- The complexity and infeasibility of the optimal solution, in our proposed framework presented in Chapter 3, to respond to the volume of viewing requests in real-time.
- The centralization of our predictive model in the master server, in our proposed framework presented in Chapter 3, which makes it less fault tolerant.
- Assuming that all the viewers will receive the highest quality is not a realistic scenario, because viewers' devices are usually heterogeneous, and the original broadcasted video quality is not always high.
- In our previous framework, resources are allocated on the fly after predicting the potential number of viewers in each site. However, we aim to reserve the computational resources beforehand to minimize the access delay, and minimize the system cost.
- Transcoding the original video into multiple video qualities 240p, 360p, 480p and 720p is computationally intensive.

4.2 Contributions

1. We develop an offline resource allocation optimization for allocating transcoding resources, and serving the viewers from their nearest cloud site, with the objective of minimizing the overall system cost while maximizing the viewers' QoE.
2. To proactively reserve the exact transcoding resources for incoming live videos, we adopt machine learning to build our distributed time series resources forecasting models.

4.3 System model

In our work, we adopt a geo-distributed cloud infrastructure as shown in Fig. 23 that consists of multiple geographically distributed data centers. Our offline resource allocation optimizer is deployed in a centralized master server. A set of geo-distributed broadcasters broadcast their live videos, which will be allocated by default with their original bitrate representations in their nearest cloud sites. The incoming live videos information including, viewers' locations and original broadcasted video quality will be collected in each broadcaster data center. In fact, our system is two-phase.

In the first phase, a collection of live videos is performed for a period T. This collection of historical videos information will be sent to our optimizer, deployed in a master server, in order to decide the optimal number of computational cloud instances rented across the geo-distributed data centers. The optimizer decisions guarantee that the viewers are served their requested video quality with minimum delay and the minimum system cost. Moreover, the optimizer determines from which cloud site the viewers should be served. Decisions are then used to create our time series datasets containing records of the number of cloud instances rented for each past time interval t in T. Such that, each cloud site has its own independent dataset.

Then, a predictive model is trained and deployed at each cloud site, in order to predict the number of cloud instances to be reserved for the upcoming time interval.

Phase two is executed in real-time. In fact, at the beginning of period t, the optimizer, will receive historical incoming videos from period t-1 to decide the optimal computational resources at each cloud site cloud site for t+1, as illustrated in Fig. 23. Specifically, the optimizer decisions will be sent to the distributed predictive models in each cloud site, in order to forecast and reserve the required cloud instances for the future period t+1.

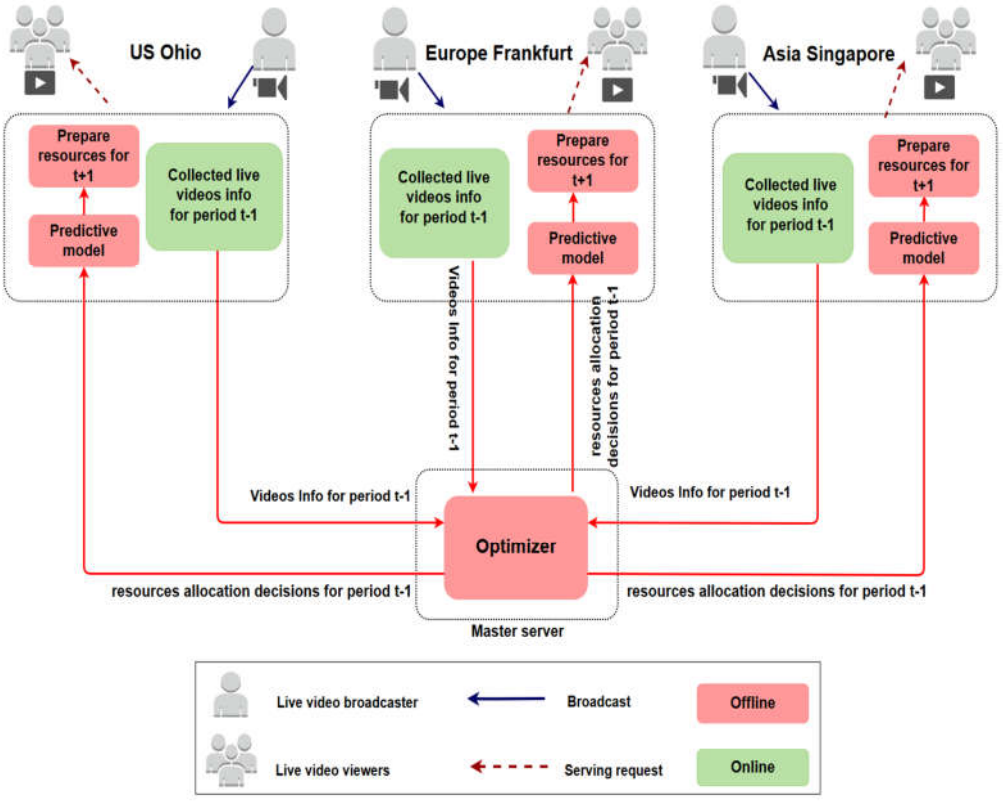

Figure 25. Resources forecasting and reservation system model.

4.3.1 Optimal transcoding resources for historical videos

In this section, we formulate the problem of offline resource allocation, to derive the optimal number of video transcoding instances and the nearest cloud site to serve the viewers with their requested bitrate representation, with an objective of minimizing the system cost constrained by the startup delay.

4.3.1.1 Dataset

In our work, we are using the Facebook 2018 live videos dataset collected by our team [10], containing more than two million Facebook live video streams. The active video streams metadata are fetched every 3 minutes in different periods on January, February, March, May, June, July, September and October 2018. As a result, we obtained a list of fetches related to each video and containing the number of viewers at the recording time. The live videos are collected with many features such as creation time and day, broadcaster location, number of likes and most importantly the viewers' locations. In this work, we selected four features for each video namely, the broadcaster location, viewers' locations, width and height of the video. The viewers' locations were selected from the video fetch with maximum number of views.

4.3.1.2 Preprocessing

As we adopted a geo-distributed cloud infrastructure to build a distributed offline transcoding resource preparation framework, there was a need to preprocess our raw data. First, we mapped the viewers' locations into 10 Amazon Web Services (AWS) cloud sites locations [27] namely, Asia-Mumbai, Asia-Seoul, Asia-Singapore, China-Ninxgia, Europe-Frankfurt, Europe-Paris, South America-Sao paulo, US East-Ohio, US East-Virginia and US West-California. This was done by calculating the shortest distance between the viewer's locations and the 10 AWS cloud sites locations. We also mapped the broadcaster location to his nearest AWS cloud site. Furthermore, we

extracted the bitrate representation for each video according to its width and height[11], assuming that the frame rate is 30fps and the amount of the motion in the image is medium for all videos. Finally, since requested qualities for each stream are missing from our dataset, and based on bandwidth and screen resolutions statistics[12] in each continent, we classified the requested qualities of each video into four bitrate representation classes: 240p, 360p, 480p and 720p. Note that the highest bitrate representation allowed by Facebook to broadcast live videos is 720p. Finally, for each video, we calculated the number of viewers for each bitrate representation, where we took into consideration that a video cannot have viewers for a video quality higher than the original broadcasted quality.

4.3.1.3 Problem formulation

The set of incoming live videos at period t is denoted by V(t)=$\{v_1, v_2, v_3,...v_m\}$. The set of video bitrate representations is represented by Q=$\{q_1, q_2, q_3, q_4\}$. 240p, 360p, 480p and 720p are the bitrate representations in Q. Let q^b, q^{re} and q^{tr} denote the original broadcasted, requested and transcoded video quality respectively. Let QT=1 if $q^{re} \leq q^b$ and 0 otherwise. Let QB(t)=$\{q_1^b, q_2^b, q_3^b,..., q_m^b\}$ denote the broadcasted video bitrate representations for the set of incoming videos at period t. The set of regions is denoted by R=$\{r_1, r_2, r_3,....r_n\}$. Let r^b, r^{tr} and r^w denote the broadcasting region, video transcoding region and video serving region respectively. The round trip delay from r^{tr} to r^w is represented by $d_{r^{tr}r^w}$.

Let P=$\{P_{v_1}, P_{v_2},..... P_{v_m}\}$ represent the set of viewers for the incoming videos at period t. As each video has viewers in different regions, let P_v=$\{p_1,p_2, p_3,.... p_n\}$ denote the number of viewers at different regions for a video v. At each region, the number of

[11] Rhiannon https://vzaar.com/wp-content/uploads/2018/05/logo-padded-1-300x138.png
[12] http://gs.statcounter.com/screen-resolution-stats

viewers requesting different bitrate representations is represented by $p_r(Q)$={$p_r(q_1)$, $p_r(q_2)$, $p_r(q_3)$, $p_r(q_4)$}. The broadcasters' regions set for the incoming videos at period t is denoted by B(t)={ r_1^b, r_2^b, r_3^b,..., r_m^b}.

Due to the fact that some videos do not have any viewer for a specific bitrate representation near some cloud sites, let E(v, q, r^w) present a binary variable, equal to 1, if video v has viewers for bitrate representation q near the region r^w, and 0 otherwise.

Video transcoding is a computationally expensive process that requires renting a single elastic cloud instance to transcode a video to a single bitrate representation. In our work, we consider renting an Amazon EC2 c5.large compute instance[13] for transcoding a higher video bitrate to a lower one.

The decision variable I(v, q, r) is equal to 1, if video v is allocated and transcoded to quality q in region r, and 0 otherwise. While the decision variable W(v, q^{re}, r^{tr}, r^w) is equal to 1, if viewers at region r^w are served a video with the requested quality q^{re} from region r^{tr} and 0 otherwise.

Three types of costs are taken into account: (1) the computational cost for renting a number of cloud instances at each cloud site; (2) the migration cost of the original video replica from one cloud site to another and (3) the cost of serving viewers. On transcoding cloud site at region r^{tr}, let α^{tr} the cost of renting a cloud instance per hour, which varies based on site location. For example, Amazon charges for c5.large 0.085$ at Ohio region, while it charges 0.131$ at Sao Paulo [28]. The total transcoding cost **T** can be calculated as presented in Eq. 9.

[13] https://aws.amazon.com/ec2/pricing/on-demand/

$$\mathbb{T}= \sum_{v \in V(t)} \sum_{q^{tr} \in Q} \sum_{r^{tr} \in R} \alpha_{r^{tr}} * \mathrm{K} * \mathrm{I}(\mathrm{v}, q^{tr}, r^{tr}) \qquad \text{Eq. 9}$$

Given that η_{r^b} is the cost to migrate a copy of a video from the broadcaster region r^b to transcoding region r^{tr}, which is the data transfer cost from one cloud site to another per GB, and $\mathrm{K}(q^{tr})$ is the size of a video with bitrate representation q^{tr}, the total migration cost $\mathbb{M}$ is calculated as presented in Eq. 10.

$$\mathbb{M}=\sum_{v \in V(t)} \sum_{q^{tr} \in Q} \sum_{r^{tr} \in R} \eta_{r^b} * \mathrm{K}(q^b) * \mathrm{I}(\mathrm{v}, q^{tr}, r^{tr}) \qquad \text{Eq. 10}$$

Given that $\omega_{r^{tr}}$ is the serving request cost from region r^{tr}, the total serving request cost $\mathbb{R}$ is calculated as presented in Eq. 11.

$$\mathbb{R}=\sum_{v \in V(t)} \sum_{q^{re} \in R} \sum_{r^{tr} \in R} \sum_{r^w \in R} \omega_{r^a} * \mathrm{K}(q^{re}) * p_{r^w}(q^{re}) * W(v, q^{re}, r^{tr}, r^w) \quad \text{Eq. 11}$$

The serving cost is defined as the data transfer cost from that region to the internet per GB, that varies based on cloud region and data transfer thresholds fixed by Amazon EC2, for example, Amazon EC2 charges 0.09$ for the first 9.99TB of data transferred, 0.085$ for the next 40TB, while it charges 0.05$ when exceeding 150TB of data transfer for Ohio cloud data center[13]. In this thesis, we suppose that the data is transferred in chunks having the same length and different sizes that depend on the bitrate version. The overall cost $\mathbb{C}$ to serve viewers is shown in Eq. 12.

$$\mathbb{C}=\mathbb{T}+\mathbb{M}+\mathbb{R} \quad \text{Eq. 12}$$

Our objective is to minimize the cost for period t as shown in Eq. 13:

$$min_{\mathrm{I(v,q,r)}\, W(v, q^{re}, r^{tr}, r^w)} \mathbb{C} \quad \text{Eq. 13}$$

Subject to the following constraints:

Every video is allocated with its original quality by default in the broadcaster closest cloud site.

$$\mathrm{I}(\mathrm{v}, q^b, r^b)=1 \quad \forall\, \mathrm{v} \in \mathrm{V(t)}, \forall\, q^b \in \mathrm{QB(t)}, \forall\, r^b \in \mathrm{B(t)} \qquad \text{Eq. 13a}$$

A video v can be served with quality q^{re} from region r^{tr} to viewers at region r^w, only

if it is allocated and transcoded at region r^{tr} to quality q^{re}

$$W(v, q^{re}, r^{tr}, r^{w}) \leq \mathrm{I}(v, q^{re}, r^{tr})$$

$$\forall\, v \in V(t), \forall\, q^{re} \in Q, \forall\, r^{tr} \in R, \forall\, r^{w} \in R \quad \text{Eq. 13b}$$

A video v can be served from region q^{re} from region r^{tr} from region r^{w} only if there exist viewers for quality q^{re} at r^{w}.

$$W(v, q^{re}, r^{tr}, r^{w}) \leq \mathrm{E}(v, q^{re}, r^{w})$$

$$\forall\, v \in V(t), \forall\, q^{re} \in Q, \forall\, r^{tr} \in R, \forall\, r^{w} \in R \quad \text{Eq. 13c}$$

If there exist viewers for video v with quality q^{re} at region r^{w}, they can only be served with one quality from one region.

$$\sum_{r^{tr} \in R} W(v, q^{re}, r^{tr}, r^{w}) = \mathrm{E}(v, q^{re}, r^{w})$$

$$\forall\, v \in V(t), \forall\, q^{re} \in Q, \forall\, r^{w} \in R \quad \text{Eq. 13d}$$

A video can not be transcoded to a quality higher than the broadcasted quality.

$$\mathrm{I}(v, q^{tr}, r^{tr}) \leq QT \quad \forall\, v \in V(t), \forall\, q^{tr} \in Q, \forall\, r^{tr} \in R \quad \text{Eq. 13e}$$

The average serving request delay to serve a video v should not exceed a threshold *D*.

$$\frac{\sum_{r^{tr} \in R} \sum_{r^{tr} \in R} \sum_{r^{w} \in R} p_{r^{w}}(q^{re}) * d_{r^{tr} r^{w}} * W(v, q^{re}, r^{tr}, r^{w})}{\sum_{r^{w} \in R} p_{r^{w}}} \leq D \;\; \forall v \in V(t) \quad \text{Eq. 13f}$$

Binary decision variables that can be set to 0 or 1.

$$\mathrm{I}(v, q^{tr}, r^{tr}), \mathrm{W}(v, q^{re}, r^{tr}, r^{w}) \in \{0,1\} \quad \text{Eq. 13g}$$

Table 4. Notations For The Formalized Problem.

Notation	**Description**
V(t)	Set of incoming live videos at period t
R	Set of regions
B(t)	Set of broadcasters regions for videos at period t

Notation	Description
QB(t)	Set of broadcasted bitrates for videos at period t
r^{tr}, r^{w}, r^{b}	Region of video transcoding, Region of serving and Region of broadcasting
P(t)	Set of viewers for live videos at period t
P_v	Set of viewers at different R for video v
$p_r(Q)$	Set of viewers for different representations for video v at region r
SU	Set of storage used at each region
W(v, q^{re}, r^{tr}, r^{w})	Binary decision variable that indicates the serving site and quality
I(v, q^{tr}, r^{tr})	Binary decision variable that indicates the allocation, transcoding site and quality
E(v, q^{tr}, r^{tr})	Binary variable that indicates viewers existence for a video quality
$d_{r^{tr}r^{w}}$	Round trip delay between r^{tr} and r^{w}
RTT	Matrix for round trip delay between the different
D	Delay threshold
K	Video size
$\alpha_{r^{tr}}$	Cloud instance cost at region
$\eta_{r^{b}}$	Migration cost per GB from broadcaster region r^{b}
$\omega_{r^{tr}}$	Serving request cost per GB from r^{tr}
T	Total transcoding cost

Notation	Description
M	Total migration cost
R	Total serving request cost
C	Overall cost

4.3.1.4 Historical rented resources datasets

We used our offline optimizer decisions to calculate the number of rented computational cloud instances at each past time frame t in a past period T. We then constructed new datasets for each cloud site containing records of the number of historical rented resources for each t in T. Finally, we restructured the time series data into a supervised learning using the sliding window technique, where a sequence of previous ε time steps will be used as an input to our models in order to predict resources for a time step ahead, as illustrated in Fig. 24.

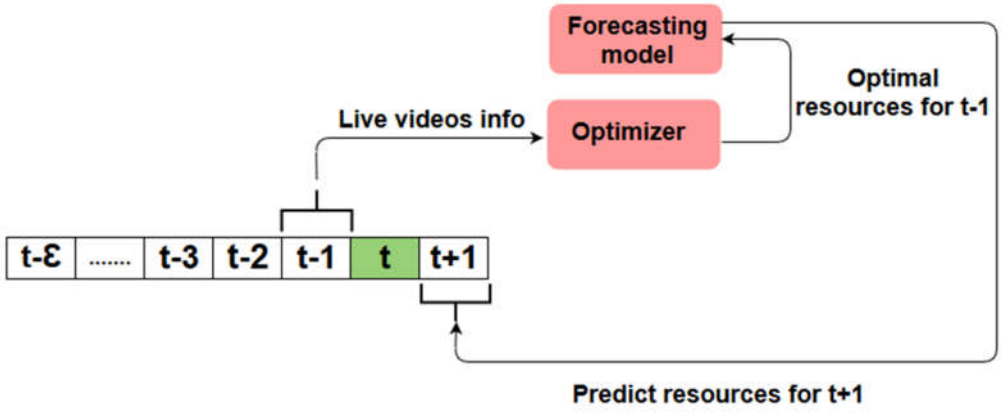

Figure 26. System model timeline at the start of t.

4.3.2 Time series resources forecasting

In our work, we adopted five different machine learning algorithms namely, Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural

Network (CNN), MultiLayer Perceptron (MLP) and XGboost to train our models for each cloud site. We set the mean absolute error (MAE) as a loss function to train our models. We built several models using each ML algorithm, as there is no method to predetermine the best combination of hyperparameters, such as the number of hidden layers and neurons for LSTM, GRU, CNN and MLP models, number of estimators for XGboost models. Finally, the best models as shown in Table. 5 were selected considering the best determination coefficient R^2 values, which is used to assess the goodness of fit of our regression models. R^2 values approaching 1 indicate that the model provides accurate predictions, and it is calculated according to Eq. 14:

$$R^2=1-\frac{\sum_{t=1}^{T}(A_t - P_t)^2}{\sum_{i=1}^{m}(A_t-\bar{A})^2} \quad \text{Eq. 14}$$

Where T is the number of time steps, A_t is the actual number of resources at time step t, P_t is the predicted number of resources for time step t, and $\bar{A}$ is the mean of the actual number of resources of all time steps.

Table 5. Best Resources Predictive Models Configurations.

Model	**Configurations**
GRU	Layers=1/Neurons: 100 Optimizer: Adagrad Dropout: 0.4 Loss function: Mean Absolute Error (MAE)
LSTM	Layers=1/Neurons: 100 Optimizer: Adagrad Dropout: 0.4 Loss function: MAE
CNN	Conv1D with 64 filters and kernel_size=2 Layers=1/Neurons: 100 Optimizer: Adagrad Dropout: 0.4 Loss function: MAE

Model	Configurations
MLP	Layers=1/Neurons: 100 Optimizer: Adagrad Dropout: 0.4 Loss function: MAE
XGboost	Loss function: MAE Estimators No: 1000

4.3.3 Proactive resources reservation

The proposed proactive resource reservation algorithm is presented in Algorithm. 2.

Algorithm 2. Proactive resources reservation.

In fact, in real-time at the start of each period t, the offline resource allocation optimizer will receive the set of the collected videos information of period t-1, as illustrated in Fig. 24, in order to decide the optimal number of transcoding cloud sites and the nearest cloud site to serve the viewers. Based on the optimizer decisions, the number of optimal cloud instances at t-1 is calculated for each geo-distributed cloud site and sent to the corresponding forecasting model. The distributed forecasting models at each cloud site will predict the number of cloud instances needed for the incoming videos for time frame t+1.

Algorithm 3. Proactive resources reservation.

Algorithm 2 Proactive resources reservation

Input: R, $\{\alpha_1, ..., \alpha_n\}$, $\{\eta_1, ..., \eta_n\}$, $\{\omega_1, ..., \omega_n\}$, RTT, $\kappa(q_1)$, $\kappa(q_2)$, $\kappa(q_3)$, $\kappa(q_4)$
for $t \in \{1, .., T\}$ **do**
    - Send videos $V(t-1)$, their bitrate $QB(t-1)$ and broadcasters $B(t-1)$ to the optimizer
    - Get optimal solution $\min_{I(v,q^{tr},r^{tr})W(v,q^{tr},r^{tr},r^{w})}$ $\mathbb{C}$ as per Eq. 5
    for $region\ r^{tr} \in R$ **do**
        - Calculate resources No used for videos $V(t-1)$
        - Send resources No used in period t-1 to the forecasting model.
        - Run forecasting model to predict resources for period t+1
        - Reserve the required computational cloud instances for t+1.

4.4 Performance Evaluation

In this section we evaluate the performance of our system. We first present our simulation settings, and then we discuss the results of our simulation.

4.4.1 Simulation settings

To evaluate the performance of our system we used 23rd June 2018 live videos to get the hourly optimal resource allocation for T=24(hours) and t=1(hour). We implemented the optimization problem using Matlab in CVX solver. In our system, we assume that the video duration is 1 hour. Moreover, we assume that the viewers are served with their requested video quality, where we set the video size for each bitrate representation as follows: K(q_1), K(q_2), K(q_3) and K(q_4) to 0.405, 0.495, 0.603 and 0.738 Gbit respectively.

We constructed our round trip time (RTT) matrix $d_{r^{tr}r^{w}}$ by calculating the average RTT from different cloud sites using[10] accessed on September 19, 2018. The computational cloud instances and data transfer prices of Amazon EC2 c5 large[13] are

considered in our simulation to model α, ω and η.

We varied the latency thresholds constraints *D* for serving a video to 8.8ms, 120ms and 180ms. 8.8ms is the latency needed to serve a viewer from its nearest cloud site [5].

To construct our time series datasets, we used our offline resources allocation optimizer decisions on 3rd to 30th June 2018 live videos using variant latency thresholds namely, 8.8ms, 120ms and 180ms. First, we calculated the optimal number of hourly used computational cloud instances at each cloud site. Second, we constructed three time series datasets for each cloud site containing the hourly rented resources using the three variant latency thresholds. Finally, we restructured the time series data into a supervised learning by setting the window size Ɛ=24, which means that at the start of period t, our models will use the previous 24 hours rented cloud instances number in order to predict the resources needed for time step t+1 as shown in Fig. 24. It is worth mentioning that because we are constrained by the size of our live videos dataset, we considered the hourly time series forecasting. However, for a larger dataset, we can decompose the time series into t equal to days, weeks and even months. In this way, the resources prediction and renting can be done before a longer period of time. We splitted our data into training and testing where we trained our models with data of 3rd to 24th June and tested with 25th to 30th June data.

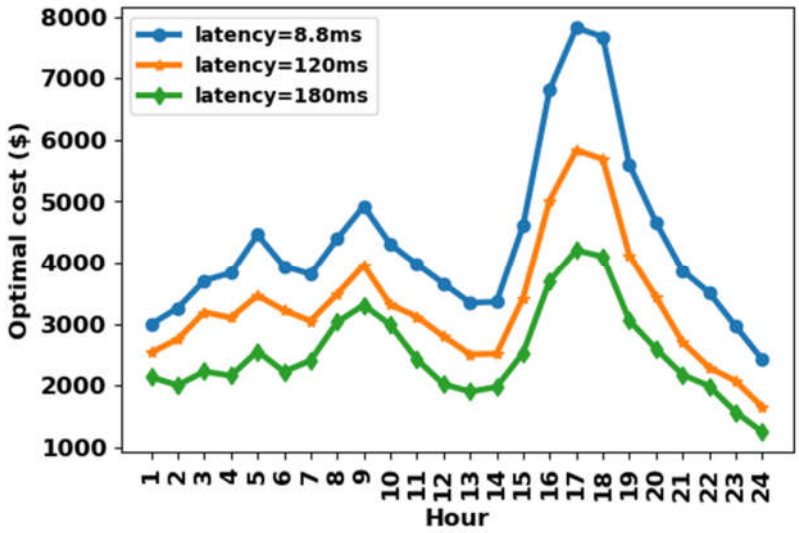

Figure 27. Hourly optimal cost.

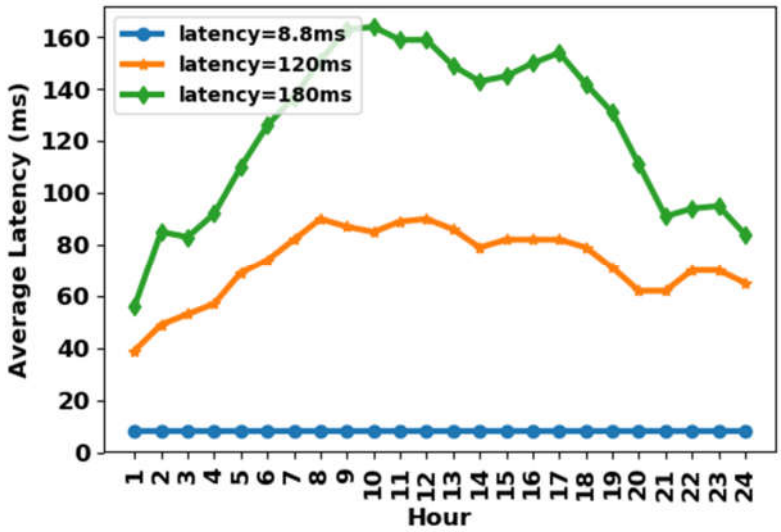

Figure 28. Hourly average latency.

4.4.2 Simulation results

Fig. 25 and 26 represent the results of the optimization to create the optimal set of resources. Fig. 25 illustrates that a trade-off between the video serving delay and the resource allocation cost can be established. In fact, the hourly optimal cost is high when the system is forced to serve the viewers from their nearest cloud site by setting

the latency threshold to 8.8ms, because the system is obliged to allocate and transcode multiple copies of the videos. Reducing the threshold leads to minimizing the cost. Therefore, the content provider can sacrifice in terms of cost to enhance the QoE or vice versa based on his system requirements. It is worth mentioning that the fluctuation in the hourly optimal cost is due to the fact that some hours have a higher number of incoming videos compared to others. Fig. 26 presents the hourly average latency achieved to serve a video. We notice that setting the latency threshold to 120ms and 180ms always achieves a much lower hourly average latency that does not even exceed 100ms and 170ms for 120ms and 180ms thresholds respectively. Moreover, setting the latency threshold to 8.8ms achieves an average latency of 8.8ms all the time, because it is the lowest latency performed when the system by serving the viewers from their regions.

After creating our time series datasets from the optimization results and training our forecasting models, we compared the performance of different techniques. The testing results, depicted in Table 6, 7 and 8 showed that in general GRU, LSTM and MLP achieved the best for most of the models, we noticed that XGboost achieved the best for some models, and CNN was the least in performance all the time. Moreover, there is no single ML algorithm that worked the best for all, but as we are adopting a distributed system, where we have a different dataset and a different forecasting model for each cloud site, different ML algorithms can be adopted. We noticed that our models performed differently on datasets having different thresholds because varying the latency thresholds causes the system to rent a different amount of cloud instances hourly at each cloud site which will affect the shape of the data. The predicted number versus the actual number of cloud instances for the live videos of 25th to 30th June 2018 (144hrs) at Singapore, Frankfurt and Virginia data centers are presented in Fig.

27 to Fig. 35. The results proved that the predicted number of resources is close to the optimal resources.

Table 6. R^2 Testing Results For 8.8ms Latency Threshold Dataset.

	GRU	LSTM	MLP	CNN	XGboost
Mumbai	0.76	0.77	**0.81**	0.66	0.76
Seoul	0.92	0.92	**0.93**	0.86	0.92
Singapore	0.92	**0.93**	0.92	0.88	**0.93**
China	**0.94**	0.93	0.93	0.91	0.93
Frankfurt	**0.85**	0.83	0.82	0.74	0.76
Paris	0.78	**0.81**	0.80	0.71	0.65
Sao Paulo	0.81	**0.82**	0.80	0.72	0.82
Ohio	0.78	0.78	0.77	0.54	0.68
Virginia	0.78	**0.79**	0.77	0.71	0.75
California	0.68	**0.73**	0.66	0.62	0.60

Table 7. R^2 Testing Results For 120ms Latency Threshold Dataset.

	GRU	LSTM	MLP	CNN	XGboost
Mumbai	**0.79**	0.71	0.70	0.67	0.58
Seoul	**0.94**	**0.94**	0.92	0.89	0.93
Singapore	**0.95**	0.94	0.94	0.92	0.92
China	**0.94**	**0.94**	**0.94**	0.89	0.92
Frankfurt	**0.83**	0.80	**0.83**	0.81	0.67

	GRU	LSTM	MLP	CNN	XGboost
Paris	**0.73**	0.72	0.73	0.61	0.72
Sao Paulo	0.78	**0.79**	0.75	0.63	0.78
Ohio	**0.77**	**0.77**	0.71	0.62	0.73
Virginia	0.77	0.76	**0.80**	0.58	0.78
California	0.73	0.73	**0.78**	0.69	0.64

Table 8. R^2 Testing Results For 180ms Latency Threshold Dataset.

	GRU	LSTM	MLP	CNN	XGboost
Mumbai	**0.71**	0.70	0.69	0.67	0.68
Seoul	0.89	**0.90**	**0.90**	0.87	**0.90**
Singapore	**0.94**	0.93	0.93	0.93	0.91
China	0.93	**0.94**	**0.94**	0.92	0.93
Frankfurt	0.78	**0.79**	0.77	0.68	0.62
Paris	**0.76**	0.74	**0.76**	0.75	0.75
Sao Paulo	0.78	0.78	0.78	0.73	**0.81**
Ohio	**0.71**	0.70	0.67	0.51	0.64
Virginia	**0.71**	0.61	0.55	0.47	0.52
California	**0.81**	0.80	0.77	0.62	0.70

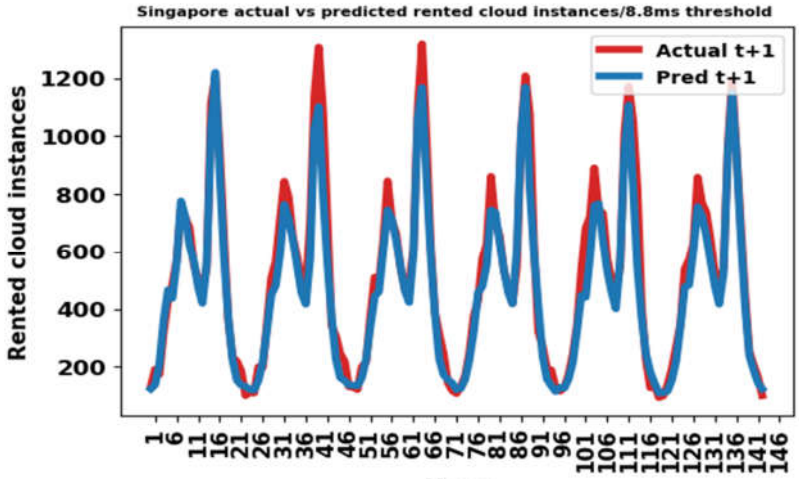

Figure 29. Actual vs predicted cloud instances at Singapore/8.8ms latency threshold dataset.

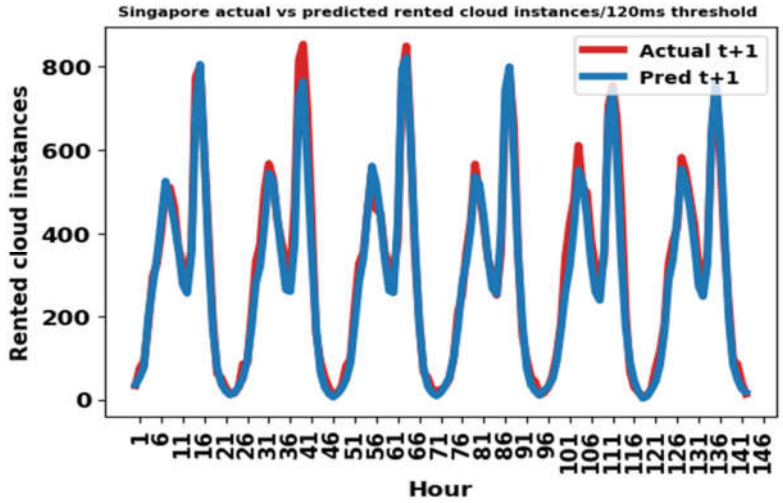

Figure 30. Actual vs predicted cloud instances at Singapore/120ms latency threshold dataset.

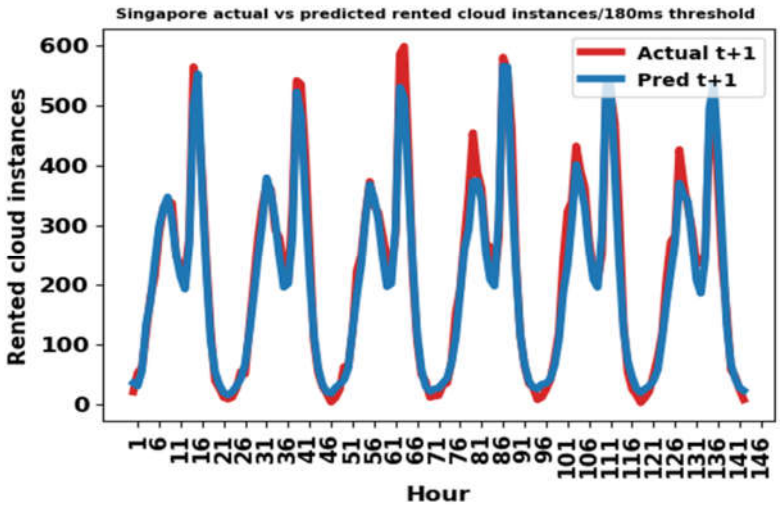

Figure 31. Actual vs predicted cloud instances at Singapore/180ms latency threshold dataset.

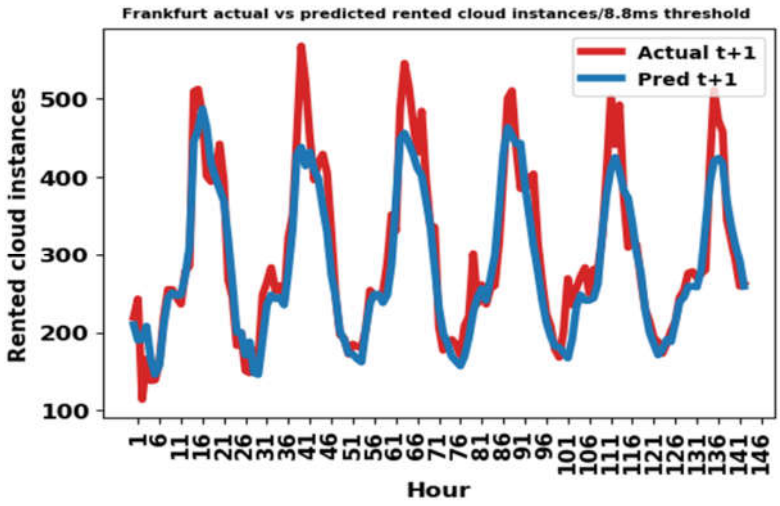

Figure 32. Actual vs predicted cloud instances at Frankfurt/8.8ms latency threshold dataset.

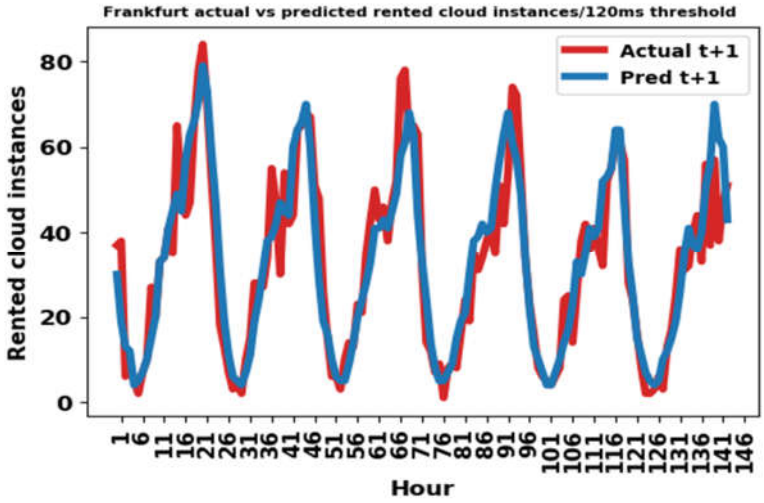

Figure 33. Actual vs predicted cloud instances at Frankfurt/120ms latency threshold dataset.

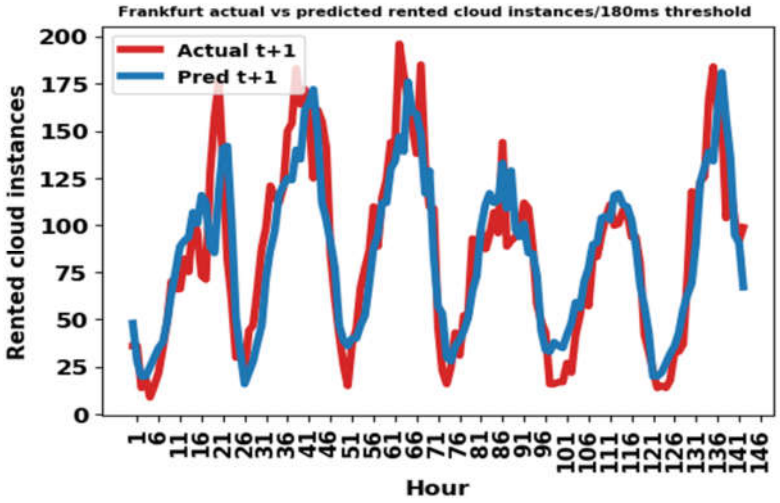

Figure 34. Actual vs predicted cloud instances at Frankfurt/180ms latency threshold dataset.

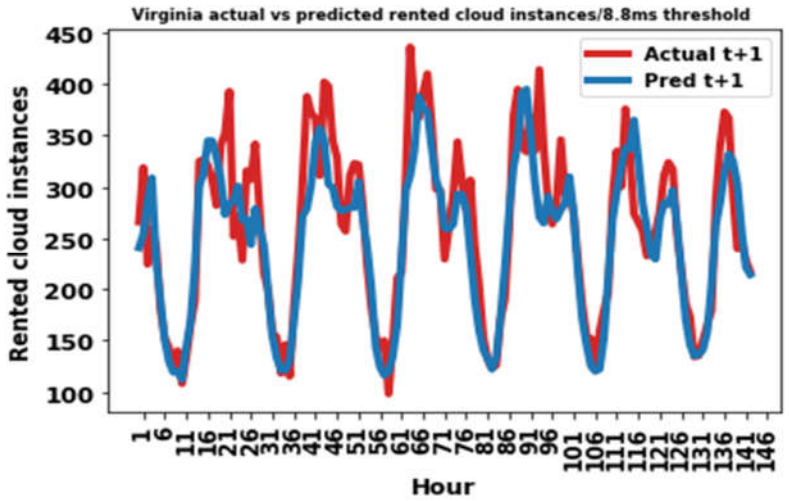

Figure 35. Actual vs predicted cloud instances at Virginia/8.8ms latency threshold dataset.

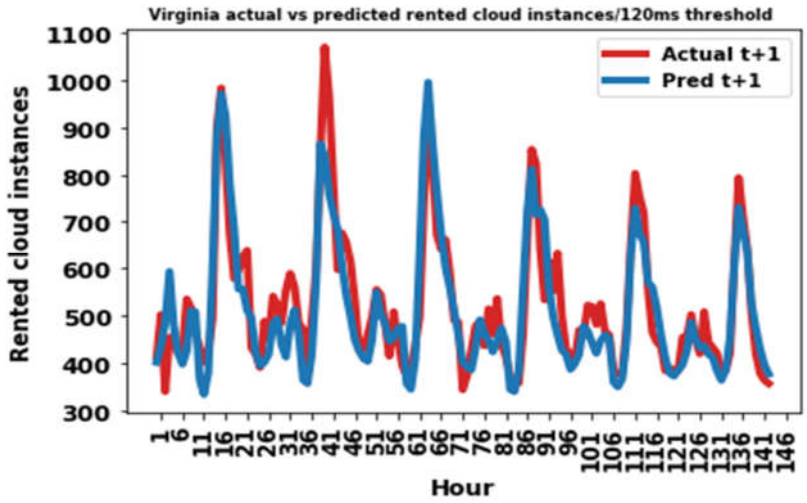

Figure 36. Actual vs predicted cloud instances at Virginia/120ms latency threshold dataset.

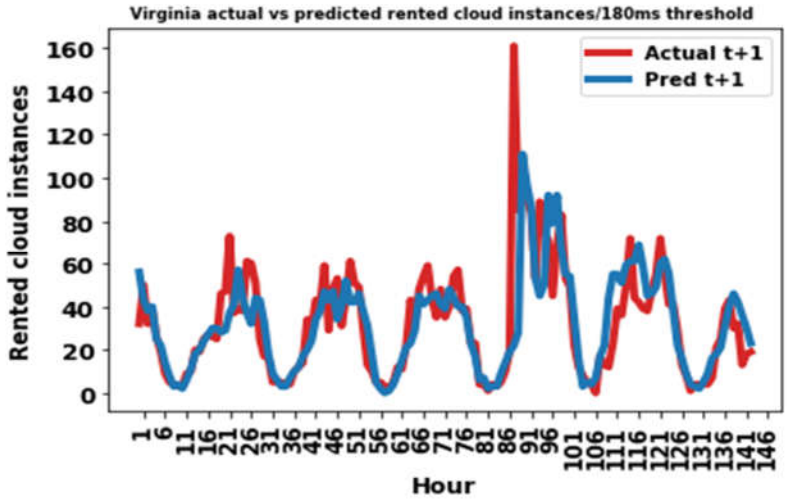

Figure 37. Actual vs predicted cloud instances at Virginia/180ms latency threshold dataset.

CHAPTER 5: CONCLUSION

In this thesis, we proposed a proactive resource allocation framework. First, we adopted machine learning to build a predictive model that captures the viewers' number near each geo-distributed cloud site. Then, based on the predicted results that are shown to be close to the actual values, we formulated our resource allocation model as an optimization problem to optimally allocate resources across the geo-distributed cloud sites. Additionally, we presented a trade-off between the video access delay and the cost of resource allocation. Moreover, we proved that the average latency to serve the actual viewers is very close to the average latency serving the predicted viewers. Our presented framework is a solution to overcome the drawback of the near optimal resource allocation algorithms, that lack the ability to allocate the exact resources needed beforehand. Near optimal solutions may either lead to over-provisioning of resources that may incur significant costs to the service providers, or under-provisioning of resources that may cause delays to the viewers.

We further proposed a novel geo-distributed proactive transcoding resources reservation framework. First, we formulated our offline resources allocation model as an optimization problem to optimally allocate transcoding resources across the geo-distributed cloud sites. Then, based on the optimizer resources allocation decisions on historical live videos, we constructed our time series reserved resources datasets for each geo-distributed cloud site. Finally, we adopted machine learning to build a forecasting model for each cloud site to predict the number of resources needed for the upcoming time frame. The novelty of our framework is attributed to the ability to rent the exact computational cloud instances beforehand as opposed to on demand renting of cloud instances, which is not always adequate for live streaming systems due to the startup time needed to boot servers. Moreover, reserving an arbitrary

number of cloud instances can be insufficient to transcode all the videos into viewers matching video quality requests, or over-provisioned, which may lead to significant additional costs to the service providers. Additionally, various cloud providers offer up to 75% discount for reserving cloud instances proactively as opposed to on demand cloud instances pricing.

5.1 Limitations and Future Work

There are still many directions of research to extend this work that remain to future. We list some limitations and our future work briefly in the following:

- Considering the complexity and infeasibility of the optimal solution to respond to the volume of viewing requests in real-time. We plan to improve our proactive resource allocation framework, by designing a heuristic to solve the resource allocation problem in real-time with near-optimal solution.
- The centralization of our predictive model in our proposed framework presented in Chapter 3, makes it less fault tolerant.
- Assuming that all the viewers will receive the highest, in our proposed framework presented in Chapter 3, is not a realistic scenario because viewers' devices are usually heterogeneous, and the original broadcasted video quality is not always high. In addition, serving the viewers with their requested quality will maximize the QoE.
- As a future work, we plan to design a heuristic to allocate the real time incoming live videos on the reserved resources predicted by our resources forecasting models presented in Chapter 4.
- We will evaluate the performance of our proactive resources reservation framework, presented in Chapter 4, against on demand resources renting

frameworks proposed by other works [4] [5].

- Last but not least, we are also interested in implementing predictive models for the number of incoming live videos, the live video duration, and the live videos viewing time.

PUBLICATIONS

1. Accepted

[1] F.Haouari, E.Baccour A.Erbad, A.Mohamed, M.Guizani. 'QoE-Aware Resource Allocation for Crowdsourced Live Streaming: A Machine Learning Approach' IEEE International Conference on Communications 2019.

2. Submitted

[2] F.Haouari, E.Baccour A.Erbad, A.Mohamed, M.Guizani. 'Transcoding Resources Forecasting and Reservation for Crowdsourced Live Streaming' IEEE Global Communications Conference 2019.

3. Under preparation

[3] F.Haouari, E.Baccour A.Erbad, A.Mohamed, M.Guizani. 'Predictive-driven Transcoding Resources Allocation for Crowdsourced Live Streaming' journal paper

[4] F.Haouari, E.Baccour A.Erbad, A.Mohamed, M.Guizani. 'PGMC: Latency-Aware Proactive Resource Allocation for Crowdsourced Live Streaming' journal paper

REFERENCES

[1] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM*, 2013.

[2] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking (TON),* vol. 21, pp. 2001-2014, 2013.

[3] L. Wei, J. Cai, C. H. Foh and B. He, "QoS-aware resource allocation for video transcoding in clouds," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 27, pp. 49-61, 2017.

[4] Q. He, J. Liu, C. Wang and B. Li, "Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach," *IEEE Transactions on Multimedia,* vol. 18, pp. 916-928, 2016.

[5] K. Bilal, A. Erbad and M. Hefeeda, "QoE-aware distributed cloud-based live streaming of multisourced multiview videos," *Journal of Network and Computer Applications,* vol. 120, pp. 130-144, 2018.

[6] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Transactions on Networking (TON),* vol. 23, pp. 689-702, 2015.

[7] S. Petrangeli, T. Wu, T. Wauters, R. Huysegems, T. Bostoen and F. De Turck, "A machine learning-based framework for preventing video freezes in HTTP adaptive streaming," *Journal of Network and Computer Applications,* vol. 94, pp.

78-92, 2017.

[8] A. M. Le, "Improving Adaptive Video Streaming through Machine Learning," 2018.

[9] G. Zhu, C. Mo, Z. Wang and W. Zhu, "User Mapping Strategies in Multi-Cloud Streaming: A Data-Driven Approach," in *GLOBECOM, 2016 IEEE*.

[10] https://sites.google.com/view/facebookvideoslive18/home, *FacebookVideosLive18 Dataset.*

[11] K. Bilal and A. Erbad, "Impact of multiple video representations in live streaming: a cost, bandwidth, and QoE analysis," in *Cloud Engineering (IC2E), 2017 IEEE International Conference on*, 2017.

[12] B. Li, Z. Wang, J. Liu and W. Zhu, "Two decades of internet video streaming: A retrospective view," *ACM transactions on multimedia computing, communications, and applications (TOMM),* vol. 9, p. 33, 2013.

[13] J. He, Y. Wen, J. Huang and D. Wu, "On the Cost--QoE tradeoff for cloud-based video streaming under Amazon EC2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 24, pp. 669-680, 2014.

[14] https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/, *EC2 Instance Pricing for Amazon Web Services (AWS),* Amazon.

[15] P. J. Brockwell, R. A. Davis and M. V. Calder, Introduction to time series and forecasting, vol. 2, Springer, 2002.

[16] C. Chatfield, Time-series forecasting, Chapman and Hall/CRC, 2000.

[17] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment,* vol. 32, pp. 2627-2636, 1998.

[18] L. Breiman, "Random forests," *Machine learning,* vol. 45, pp. 5-32, 2001.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

[20] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems,* vol. 11, pp. 68-75, 2017.

[21] H. Liu, X. Mi and Y. Li, "Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM," *Energy Conversion and Management,* vol. 159, pp. 54-64, 2018.

[22] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Gated feedback recurrent neural networks," in *International Conference on Machine Learning*, 2015.

[23] M. Peng, C. Wang, T. Chen and G. Liu, "Nirfacenet: A convolutional neural network for near-infrared face identification," *Information,* vol. 7, p. 61, 2016.

[24] F. Chen, C. Zhang, F. Wang and J. Liu, "Crowdsourced live streaming over the cloud," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[25] K. Weinberger, A. Dasgupta, J. Langford, A. Smola and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research,* vol. 15, pp. 1929-1958, 2014.

[27] https://aws.amazon.com/about-aws/global-infrastructure/, *Amazon Web Services| AWS,* Amazon.

[28] https://aws.amazon.com/ec2/pricing/on-demand/, *EC2 Instance Pricing – Amazon Web Services (AWS),* Amazon.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg and others, "Scikit-learn: Machine learning in Python," *Journal of machine learning research,* vol. 12, pp. 2825-2830, 2011.

[30] F. Jokhio, T. Deneke, S. Lafond and J. Lilius, "Analysis of video segmentation for spatial resolution reduction video transcoding," in *Intelligent Signal Processing and Communications Systems (ISPACS), 2011 International Symposium on*, 2011.

[31] T. Guarnieri, I. Drago, A. B. Vieira, I. Cunha and J. Almeida, "Characterizing QoE in large-scale live streaming," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 2017.

[32] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM Computer Communication Review*, 2011.

[33] T. Deneke, H. Haile, S. Lafond and J. Lilius, "Video transcoding time prediction for proactive load balancing," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, 2014.

[34] R. Buyya, R. Ranjan and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *International Conference on Algorithms and Architectures for Parallel*

Processing, 2010.

[35] I. R. Alzahrani, N. Ramzan, S. Katsigiannis and A. Amira, "Use of Machine Learning for Rate Adaptation in MPEG-DASH for Quality of Experience Improvement," in *5th International Symposium on Data Mining Applications*, 2018.

[36] K. Bilal, E. Baccour, A. Erbad, A. Mohamed and M. Guizani, "Collaborative joint caching and transcoding in mobile edge networks," *Journal of Network and Computer Applications,* 2019.

[37] D. Bhamare, M. Samaka, A. Erbad, R. Jain and L. Gupta, "Exploring microservices for enhancing internet QoS," *Transactions on Emerging Telecommunications Technologies,* vol. 29, p. e3445, 2018.

[38] D. Bhamare, A. Erbad, R. Jain, M. Zolanvari and M. Samaka, "Efficient virtual network function placement strategies for Cloud Radio Access Networks," *Computer Communications,* vol. 127, pp. 50-60, 2018.