Fareed Ahmed Jokhio

Video Transcoding in a Distributed Cloud Computing Environment

Turku Centre *for* Computer Science

TUCS Dissertations
No 170, January 2014

Video Transcoding in a Distributed Cloud Computing Environment

Fareed Ahmed Jokhio

To be presented, with the permission of the Department of Information Technologies at Åbo Akademi University, for public criticism in Auditorium Lambda, at ICT building, Turku, Finland, on January 31, 2014, at 13.00 noon.

Åbo Akademi University
Department of Information Technologies
Joukahaisenkatu 3–5 A, 20520 Turku, Finland

2014

Supervisor

Professor Johan Lilius
Department of Information Technologies
Åbo Akademi University
Joukahaisenkatu 3–5 A, 20520 Turku, Finland

Advisor

Dr. Sébastien Lafond
Department of Information Technologies
Åbo Akademi University
Joukahaisenkatu 3–5 A, 20520 Turku, Finland

Reviewers

Dr. Tuan Anh Trinh
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
1117 Budapest
Magyar tudosok krt. 2
Hungary

Associate Professor Lionel Morel
Département Informatique
INSA Lyon
Domaine Scientifique de la Doua, INSA Lyon
Batiment Claude Chappe, 6, avenue des Arts,
69621 Villeurbanne Cedex

Opponent

Dr. Tuan Anh Trinh
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
1117 Budapest
Magyar tudosok krt. 2
Hungary

ISBN 978-952-12-3004-2
Turku, Finland, 2014

Abstract

Video transcoding refers to the process of converting a digital video from one format into another format. It is a compute-intensive operation. Therefore, transcoding of a large number of simultaneous video streams requires a large amount of computing resources. Moreover, to handle different load conditions in a cost-efficient manner, the video transcoding service should be dynamically scalable. *Infrastructure as a Service Clouds* currently offer computing resources, such as virtual machines, under the pay-per-use business model. Thus the *IaaS Clouds* can be leveraged to provide a cost-efficient, dynamically scalable video transcoding service.

To use computing resources efficiently in a cloud computing environment, cost-efficient virtual machine provisioning is required to avoid over-utilization and under-utilization of virtual machines. This thesis presents proactive virtual machine resource allocation and de-allocation algorithms for video transcoding in cloud computing. Since users' requests for videos may change at different times, a check is required to see if the current computing resources are adequate for the video requests. Therefore, the work on admission control is also provided. In addition to admission control, temporal resolution reduction is used to avoid jitters in a video. Furthermore, in a cloud computing environment such as Amazon EC2, the computing resources are more expensive as compared with the storage resources. Therefore, to avoid repetition of transcoding operations, a transcoded video needs to be stored for a certain time. To store all videos for the same amount of time is also not cost-efficient because popular transcoded videos have high access rate while unpopular transcoded videos are rarely accessed. This thesis provides a cost-efficient computation and storage trade-off strategy, which stores videos in the video repository as long as it is cost-efficient to store them. This thesis also proposes video segmentation strategies for bit rate reduction and spatial resolution reduction video transcoding. The evaluation of proposed strategies is performed using a message passing interface based video transcoder, which uses a coarse-grain parallel processing approach where video is segmented at group of pictures level.

Sammanfattning

Video omkodning avser processen att omvandla en digital video från ett format till ett annat. Eftersom omkodning är en beräkningsintensiv uppgift, kräver omkodning av ett stort antal samtidiga videoströmmar en stor mängd datorresurser. Dessutom, för att hantera olika belastningsförhållanden påett kostnadseffektivt sätt, bör en video omkodning tjänst vara dynamiskt skalbar. Infrastruktur som en tjänst (IaaS) datormoln erbjuder idag datorresurser, såsom virtuella maskiner, inom ramen för betala-per-användning affärsmodellen. Således kan IaaS moln utnyttjas för att ge en kostnads effektivt, dynamiskt skalbar video omkodning tjänst.

För att använda datorresurser på ett effektivt sätt i en molnmiljö , krävs kostnadseffektiv virtuell maskin provisionering för att undvika överutnyttjande och underutnyttjande av virtuella maskiner . Denna avhandling presenterar proaktiv virtuell maskin resurs allokering och deallokeringsalgoritmer för videoomkodning i datormoln. Eftersom användarnas efterfrågan påfilmer kan ändras vid olika tidpunkter krävs en kontroll för att se om nuvarande datorresurser är tillräckliga för de begärda videon. Därför är arbete med åtkomstkontroll också inkluderat. Förutom åtkomstkontroll, andvänds tidsupplösningsminskning för att undvika jitter i videon . Dessutom , i en molnmiljö som Amazon EC2, är datorresurser är dyrare jämfört med lagringsresurse . För att undvika upprepning av omkodningsoperationer behöver en omkodad video lagras under en viss tid. Att lagra alla filmer lika lång tid är inte heller kostnadseffektivt eftersom populära omkodade videoklipp har hög åtkomsttakt medan mindre populära omkodade videoklipp sällan nås. Denna avhandling presenterar en kostnadseffektiv beräknings- och lagrings- avvägningsstrategi , som lagrar filmer i videoförråd så länge det är kostnadseffektivt att lagra dem . Avhandlingen föreslår också videosegmenteringstrategier fär omkodning med minskad överföringshastighet och rumslig upplösning. Utvärderingen av föreslagna strategier utförs med hjälp av en video omkodare som är konstruerad enligt ett message-passing gränssnit , som använder en grovkorning parallell beräknings strategi där videon är segmenterad enligt bilggrupp (group-of-pictures) nivå.

Acknowledgements

This thesis would not have been completed without the support of several peoples and I am thankful to all of them who helped me during my doctoral studies.

First of all I would like to express my deep and sincere gratitude to my supervisor Professor Johan Lilius and advisor Dr. Sébastien Lafond for their patient and guidance. It is an honour for me to work under their supervision.

I would also like to thank the reviewers of my thesis, Dr. Tuan Anh Trinh and Associate Professor Lionel Morel for their time and valuable comments and useful suggestions to improve the quality of this thesis. I am also very thankful to Dr. Tuan Anh Trinh for accepting to be the opponent in the public defense of my thesis.

I am also thankful to my co-authors Tewodros Deneke, Adnan Ashraf and Professor Ivan Porres and other colleagues at the Information Technologies Department at Åbo Akademi University, Finland for their support and time. I am also thankful to Dr. Leonidas Tsiopoulos and Johan Ersfolk who spent their precious time and gave me valuable suggestions during the writing process of the summary part of this thesis.

For the partial financial support I am thankful to Higher Education Commission of Pakistan, Åbo Akademi University and Nokia Foundation Finland.

I owe my deepest gratitude to my late father Mr. Atta Muhammad Jokhio and other family members for their love and support.

Publication reprints

1. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In Parallel, Distributed and Network-Based Processing (PDP), 21st Euromicro International Conference, pp. 254-261, 27th February - 1st March 2013.

2. Adnan Ashraf, Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Stream-based admission control and scheduling for video transcoding in cloud computing. In Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium, pp.482-489, May 13-16, 2013.

3. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius. A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud. In 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2013) Santander, Spain, pp. 365-372, September 4-6, 2013.

4. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Cost-efficient dynamically scalable video transcoding in cloud computing. Technical report 1098. Turku Centre for Computer Science (TUCS), 2013

5. Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Bit rate reduction video transcoding with distributed computing. In Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference, pp. 206-212, February 15-17, 2012.

6. Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Analysis of video segmentation for spatial resolution reduction video transcoding. In Intelligent Signal Processing and Communications Systems (ISPACS), 2011 International Symposium, pp.1-6, December 07-09, 2011.

Contents

List of Figures

Glossary

ACES Admission Control based on Estimation of Service.

API Application Programming Interface.

CBAC Cost-Based Admission Control.

CBR Constant Bit Rate.

CPU Central Processing Unit.

EMA Exponential Moving Average.

FIFO First In, First Out.

GOP Group of Pictures.

HD High Definition.

HEVC High Efficiency Video Coding.

IaaS Infrastructure as a Service.

LP Load Predictor.

LT Load Tracker.

MPI Message Passing Interface.

PaaS Platform as a Service.

PDDS Proportional Delay Differentiated Service.

QoS Quality of Service.

SaaS Software as a Service.

SBACS Stream Based Admission Control and Scheduling.

SPMD Single Program Multiple Data.

VBR Variable Bit Rate.

VBV Video Buffering Verifier.

VM Virtual Machine.

Part I

Research Summary

Chapter 1

Introduction

The use of digital videos is common in our daily life and due to the massive amount of data present in a digital video, it is challenging to store and transmit uncompressed videos [27]. For cost-efficiency, a video is stored and transmitted in a certain compressed format. Video compression is a mature field and there are several techniques available to compress digital video. The compression is possible by utilizing both spatial and temporal redundancies. Several algorithms such as transform coding, entropy coding and frame prediction are used for video compression [32]. There are several video coding standards such as H.26x [26, 2, 4, 50, 71], MPEG [1, 3, 5] and HEVC [69] based on these algorithms to ensure correct functioning between encoders and decoders [39].

Users access videos with a variety of devices having different processing and storage capabilities such as cell phones, digital televisions, desktop machines, laptops, tablets etc. These devices may gain access to a video with different communication networks operating on different ranges of bandwidth. With the passage of time, the number of video compression formats is growing. Due to limited processing power and storage space, a device may support only a subset of the existing video formats. To solve this problem, a video can be either converted and stored in all possible formats at the server side or it can be converted into another format as it is requested. However, due to storage cost, it is not practically possible to store a video in all possible formats to fulfill the end-user requirements. A possible solution is to convert the unsupported video as it is requested and store it as long as it is cost-efficient to store. The process of converting a compressed video from one format into another is called video transcoding.

Video transcoding can change the bit rate [72, 47], frame resolution [66, 74], frame rate [67, 46], format [44], or any combination of these [17]. The goal of the bit-rate reduction video transcoding is to drop the bit-rate of a video stream by applying coarser quantization while maintaining low

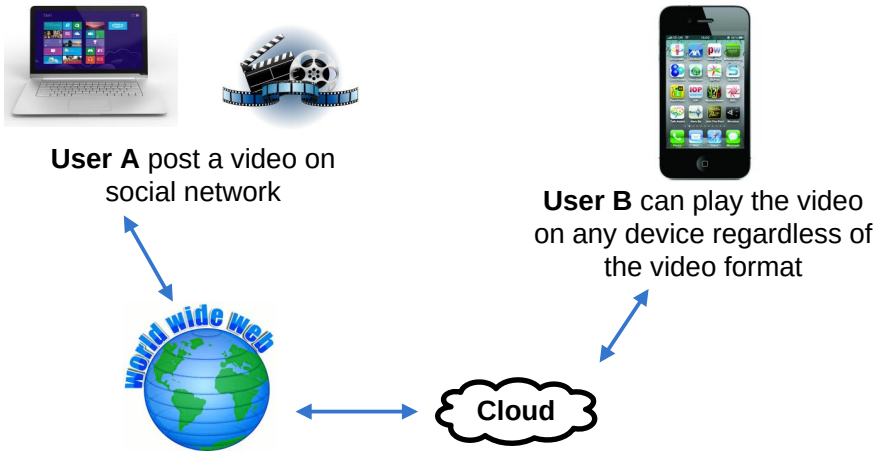

Figure 1.1: Cloud-Based Video Transcoding

complexity and meet the highest possible quality. In temporal resolution reduction transcoding, usually some frames are dropped to change the size of the video. The spatial resolution reduction video transcoding is required when a device with small screen resolution cannot play a high-resolution video. The format conversion is needed when the device at user-end does not support the video format.

Video transcoding is a computationally intensive operation and devices at user-end may not be suitable to do video transcoding. Therefore, it is usually performed at the server-side. It may be done in real-time or in batch processing. However, for an on-demand video streaming service, if the required video is not available in the desired format, the transcoding needs to be done on-the-fly in real-time. One of the main challenges of a real-time video transcoding operation is that it must avoid over and underflow of the output video buffer, which temporarily stores the transcoded videos at the server-side. The overflow occurs if the video transcoding rate exceeds the video play rate and the capacity of the buffer. Likewise, the buffer underflow may occur when the play rate exceeds the transcoding rate, while the buffer does not contain enough frames to avoid the underflow situation.

The number of user requests for video transcoding may vary at different times. To perform video transcoding in such an environment, a system, which can give computing resources on demand is needed. Cloud computing can offer such an environment in which resource provisioning is possible as they are needed [14, 49]. A cloud-based video transcoding use case is shown in Figure 1.1. The figure shows that User 'A' uploads a video on a social network, User 'B' can get access to the transcoded video and can play it regardless of the original video format.

Infrastructure as a Service (IaaS) clouds currently offer computing resources, such as Virtual Machines (VMs), storage space, and network bandwidth [59], which can be used to create a dynamically scalable cluster of video transcoding servers.

In a cloud environment, a video transcoding operation can be performed in several different ways. One possible way is to map an entire video stream on a dedicated VM. This way of mapping video streams on transcoding servers is not cost-efficient. Moreover, transcoding of High Definition (HD) video streams can take more time, which may violate the client-side Quality of Service (QoS) requirements of desired play rate [19]. Another approach is to split the video streams into smaller segments and transcode them independently [58, 35, 21, 63]. In this approach, one VM can be used to transcode a large number of video segments belonging to different video streams. Moreover, video segments of one particular stream can be transcoded on multiple VMs.

In a cloud-based video transcoding environment, the main task is to perform video transcoding in such a way so that a user can play video smoothly without video freezes and with the shortest start-up time [87]. The video freezes occur due to unavailability of video frames. The video startup time is the time at which the user will be able to watch video after selecting a video link.

1.1 Goals of this Thesis

The overall goal of this thesis is to contribute to the discussion on how the cloud computing can be used to provide a video transcoding service, so that a user can play a video smoothly without video freezes on his devices regardless of the original video format. In this work, the video transcoding is assumed to be a service, which is provided by a service provider. The sub-goals of this work are:

1. Analysis of video segmentation methods for video transcoding in a cloud and distributed computing environment.

2. Cost-efficient virtual machine provisioning algorithms to handle varying amounts of loads.

3. Admission control on server side to avoid overloading of servers.

4. A computation and storage cost trade-off strategy to estimate an equilibrium point on the time axis, which indicates duration for which a transcoded video could be stored.

1.2 Research Problems and Contributions of the Thesis

In this thesis, the main focus is on different research issues, which are related to perform distributed video transcoding in a cloud computing environment. To distribute a video among different machines efficiently, video segmentation is required. However, due to dependencies among different types of frames video segmentation is not possible at every point [33, 60, 7]. Furthermore, to use computing resources efficiently in a cloud computing environment, cost-efficient virtual machine provisioning is also required. Over-utilization and under-utilization of virtual machines needs to be avoided for better user experiences and lower cost. Since the number of user requests for videos may change at different times, a check is required to see if current computing resources are adequate for the video request. Furthermore, it is not practically possible to transcode a video for each request. To avoid repetition of transcoding operation for the same video, a transcoded video needs to be stored for a certain time. However, to store all videos for the same amount of time is also not cost-efficient. A cost-efficient computation and storage trade-off strategy is required.

1.2.1 Video Segmentation for Distributed Video Transcoding

This research work focuses on the computation, parallelization and data distribution among computing units. For data distribution a coarse grain approach is adopted in which video is segmented at Group of Pictures (GOP) level. The work presented in papers V and VI of this thesis focuses on different video segmentation strategies such as static segmentation and dynamic segmentation. A message passing programming model for distributed video transcoding is implemented to evaluate the static video segmentation strategies for different mechanisms of video transcoding. Video segmentation for distributed video transcoding is described in papers V,VI and section 3.1 of this thesis.

1.2.2 Resource Allocation and De-allocation

Video transcoding for an on-demand video streaming service needs to be done on-the-fly in real-time [68]. Since transcoding is a computationally intensive operation, video transcoding of a large number of video streams requires a large scale cluster-based distributed system. To handle varying load in a cost-efficient manner, the cluster should be dynamically scalable. IaaS Cloud is a suitable platform for video transcoding. It can provide computing resources, storage space, and network bandwidth. This the-

sis presents prediction-based dynamic resource allocation and de-allocation algorithms to scale video transcoding service on a given IaaS cloud in a horizontal fashion. The proposed algorithms allocate and de-allocate VMs to a dynamically scalable cluster of video transcoding servers. A two-step load prediction method [11] is used, which predicts a few steps ahead in the future to allow proactive resource allocation. For cost-efficiency, VM resources are shared among multiple video streams. The sharing of the VM resources is based on the video segmentation, which splits the streams into smaller segments that can be transcoded independently of one another. Resource allocation and de-allocation is described in papers I, IV and section 3.2 of this thesis.

1.2.3 Admission Control

It is possible that the existing servers may not be able to handle the incoming user load. To avoid overloading of servers, admission control on server side is used to restrict new incoming user load. Instead of directly rejecting requests in overloaded situations, the admission control uses a deferment policy and provides transcoding service for new requests after scaling the servers. In addition to admission control for new incoming video requests, the existing workload may have variable computational requirements and may cause the server overloading which is undesirable. The server overloading results in degraded performance and may lead to *transcoding jitters*. Paper II of this thesis presents admission control and a transcoding jitter preventation scheme.

1.2.4 Computation and Storage Trade-off Strategy

In an on-demand video streaming service, the source videos are usually high quality videos that are the primary datasets. Therefore, irrespective of their storage costs, they are never deleted from the *video repository*. The transcoded videos, on the other hand, are the derived datasets that can be regenerated from their source videos. Therefore, they should only be stored in the *video repository* when it is cost-efficient to store them. The computation cost of a transcoded video depends on its transcoding time and on how often the video is re-transcoded. Thus, if the same video is re-transcoded often, the computation cost would increase rapidly. On the other hand, the storage cost of a transcoded video depends upon the length of the storage duration and the video size. Therefore, it increases gradually with the passage of time. Thus, the proposed strategy estimates an equilibrium point on the time axis, which indicates the minimum duration for which the video should be stored in the *video repository*. Computation and storage trade-off strategy is described in papers III, IV of this thesis.

1.3 Organization of the Thesis

The rest of the summary part of this thesis is structured as follows. Chapter 2 discusses cloud computing and a distributed video transcoding architecture in a cloud. Chapter 3 provides the main contributions of this thesis, which include video segmentation, resource allocation and de-allocation algorithms for video transcoding in a cloud, stream based admission control algorithm, job scheduling and jitter avoidance algorithm, computation and storage trade-off strategy. Chapter 4 provides a summary of the papers included in this thesis. In chapter 5, achieved results are discussed. Finally, chapter 6 provides conclusions and future works.

Chapter 2

Video Transcoding in Cloud Computing

2.1 Cloud Computing

A cloud computing environment consists of a set of hardware, software, storage space, networks, and interfaces that are available on the Internet to deliver services. In a public cloud the physical infrastructure including computing resources, storage resources and applications are delivered on a per-use basis [1]. In a cloud computing environment, scale-up and scale-down of resources is possible. It is suitable for an on-demand video transcoding service, which requires flexibility. In a video transcoding service, the demand of resources varies with the fluctuations of load. A cloud computing environment can provide the required computing resources on a per-use basis.

A cloud supports the virtualiztion in which all software run within a VM. A VM is a computing environment, which supports operating system and application programs. Virtualization allows sharing computing resources among multiple users. There are three popular cloud computing service models including:

Software as a Service (SaaS) is the model in which a user accesses an application via the Internet. The customer is not responsible to maintain the application and uses it out of the box. The service provider is responsible to maintain the application.

Platform as a Service (PaaS) provides all the resources, which developers require to build applications and services. Some of the common solutions provided in PaaS services include database integration, storage, web service integration, versioning and security [77]. Although PaaS provides resource provisioning and load balancing, but usually users have to

[1] http://www.esri.com/news/arcwatch/0110/feature.html

pay on an hourly basis for getting those services. Therefore, the PaaS will have very high cost and it is not cost-efficient to use it for video transcoding service in cloud computing. Moreover, a PaaS provider may offer only some specific Application Programming Interface (API) features and support a limited number of programming languages, which might not fulfil the requirements.

IaaS provides on-demand services including hardware, storage, servers, operating systems, networking components and other infrastructure. The user typically pays for the resources on per-use basis while the service provider is responsible for providing the required resources and maintenance. The IaaS model is more suitable for a video transcoding service because it provides services, such as data storage and virtual servers for deploying and running applications. The users have to perform the resource allocation and load balancing. The IaaS is a low cost solution as compared with the PaaS. A user can install any programming language and API according to his requirements.

2.2 A Proposed Distributed Video Transcoding Architecture in Cloud Computing

The proposed distributed video transcoding architecture in cloud computing consists of multiple software components on multiple VMs. The VMs can be connected by a local network or a wide area network. It can provide both scalability and redundancy. Scalability means that the system can be expanded by requesting more VMs as needed. Redundancy means that the system provides the same service regardless of the failure of individual virtual machines. Figure 2.1 shows the proposed system architecture of the distributed video transcoding in cloud computing environment. It consists of a *streaming server*, a *video splitter*, a *video merger*, a *video repository*, a dynamically scalable cluster of *transcoding servers*, a *load balancer*, a *master controller*, and a *load predictor*.

2.2.1 Streaming Server

To transfer multimedia contents, such as video, to viewers over the Internet in real-time, media streaming technology is used in which parts of a video are downloaded, decoded and played [2]. The video playing and downloading happen at the same time. A buffer is used to store additional video contents from the streaming server. The overall process is invisible to the viewer.

The end-users or clients may send requests for videos, which are routed through the *streaming server*. Since the main focus of this research work

[2] http://www.wimpyplayer.com/docs/faqs/docs/general_streaming_definition.html

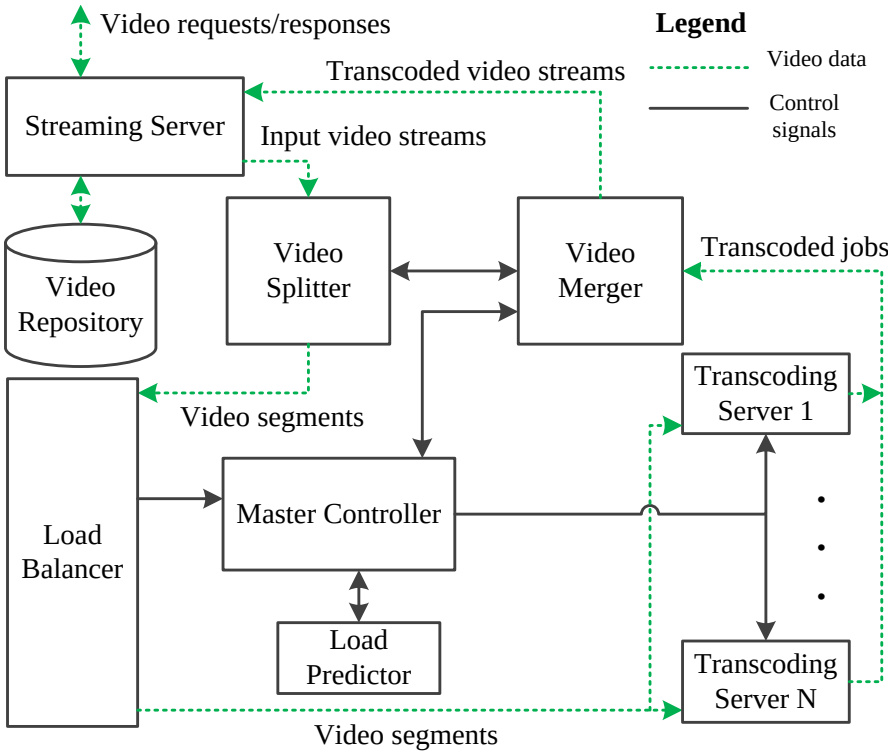

Figure 2.1: Proposed Video Transcoding Architecture in Cloud Computing

is on video transcoding, it is assumed that the *streaming server* is not a bottleneck.

Figure 2.2 shows a video streaming architecture in which a video is produced by a video camera and then after which it is encoded in a suitable format. The streaming server works as a media server, which sends streamed video to users connected with different networks.

At present there are several streaming technologies available such as progressive streaming, true streaming and adaptive/ live streaming [3]. Here is a brief overview of these technologies.

Progressive Streaming

In this technology, a video is delivered to the user device progressively where it is buffered locally and played after decoding. The technology is suitable for a short duration video only. It may have longer start-up time due to the initial buffer filling. The main disadvantage of this technology is bandwidth consumption, as the user may download more than what is actually played.

[3] http://www.rambla.be/state-play-overview-streaming-protocols

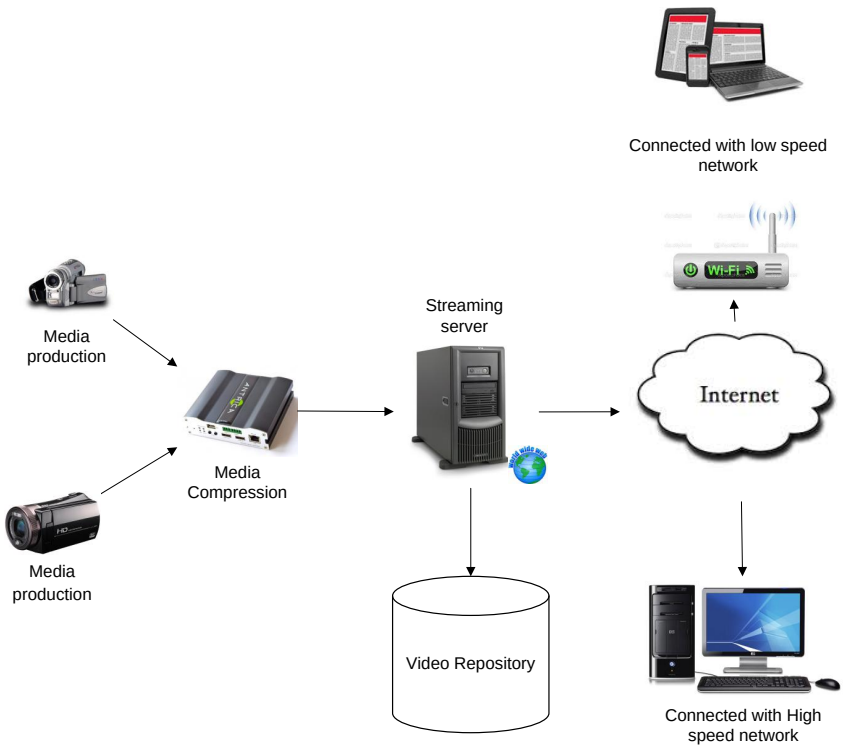

Figure 2.2: Video Streaming Architecture

True Streaming

In this technology, a video is not locally buffered at the user side. This may also be called live streaming or on-demand streaming. The advantages of this technology are very short start-up time and full control over fast forward, pause and rewind operations. This may require special media servers and firewalls. It also has higher cost as compared with the progressive streaming technology.

Adaptive Streaming

In this technology, contents of a video are adjusted with the network bandwidth conditions and processing capabilities of the devices at user side. It provides streaming service for both live streams and on-demand streams. The user has full 'seek' control over the video. It utilizes network bandwidth efficiently and sends the video segments to end users, which are requested for playing. It has a shorter start-up time as compared with progressive streaming. It may use the existing web servers. Therefore, it has lower cost and less performance as compared with true streaming technology.

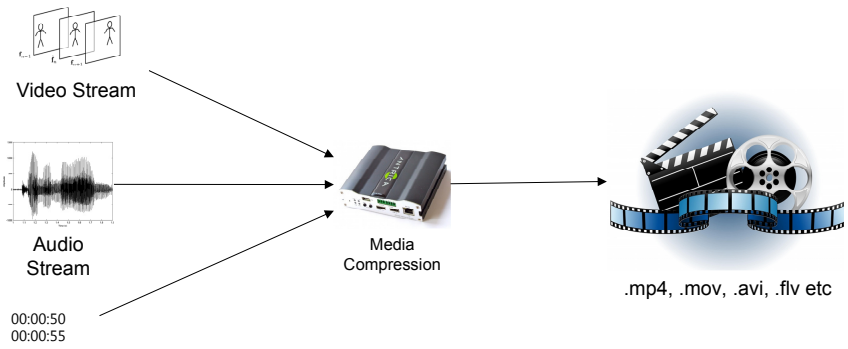

Figure 2.3: Media Compression

2.2.2 Video Repository

The video streams in certain compressed formats are stored in the *video repository*. The compressed videos can be either source videos or transcoded videos. The source videos are the original videos and transcoded videos are obtained from source videos after applying the transcoding process. The *streaming server* accepts video requests from users and checks if the required video is available in the *video repository*. If it finds the video in the desired format and resolution, it starts streaming the video. However, if it finds that the requested video is stored only in another format or resolution than the one desired by the user, it sends the video for segmentation and subsequent transcoding. Then, as soon as it receives the transcoded video from the *video merger*, it starts streaming the video.

A compressed video may consist of a video stream, an audio stream and other metadata information such as time stamps etc. Figure 2.3 shows a high level block diagram of media compression in which a video is compressed in a certain file format.

2.2.3 Video Splitter

The *video splitter* splits the video streams into smaller segments called jobs, which are placed into the job queue. Due to dependencies among different types of frames, video segmentation can be performed at certain points only. The main problem is how to perform the segmentation of source video so that parts of video can be distributed among transcoding machines. Compressed video files contain different types of frames, which have different compression rates and dependencies among them. Therefore one cannot split a given video at any particular frame or point. The *Intra(I)*-frame is independent and can be decoded without any other reference frame and is used as a reference frame for the other frames. In a given video sequence

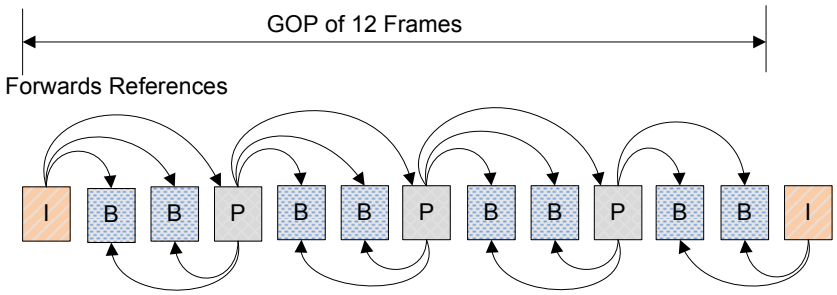

Figure 2.4: Group of Pictures (GOP)

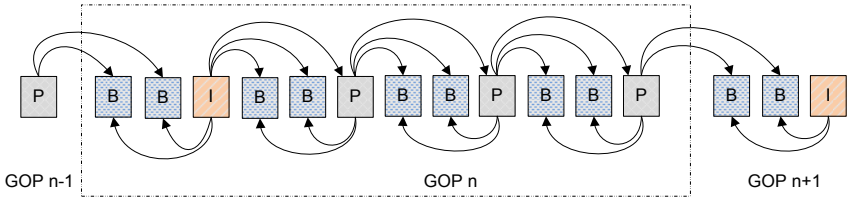

Figure 2.5: Open GOP Structure

a group of frames that constitute one *I*-frame and a number of other *Bi-directional predicted(B)* and *Predicted(P)* frames is called a GOP. Figure 2.4 shows a GOP of length 12. The video sequence partitioning algorithm utilizes this concept to divide a video stream in to smaller parts. The splitter divides the incoming video file into parts, which contain a number of GOPs and sends these parts to transcoding machines. There are two types of GOP: Open-GOP and closed-GOP.

Open-GOP

In open-GOPs, a reference frame can be from any other GOP. Due to dependencies among them, the transcoding process requires frames from both GOPs. In open-GOP the last *B*-frames are dependent to the next *I*-frame. This makes the video harder to segment. The possible solution is to add redundant frames when a video is segmented and remove those redundant frames while merging. It adds complexity, but segmentation is still possible. Another possible solution is to remove the last two *B*-frames and segment video between *P* and *I*-frames. Figure 2.5 shows, the structure of the Open-GOP where the first two *B*-frames of GOP *n*, reference the *P*-frames of GOP *n-1* and the *I*-frame of GOP *n*. The first two frames of GOP *n+1* will reference the last *P*-frame from GOP *n* and the *I*-frame from GOP *n+1*.

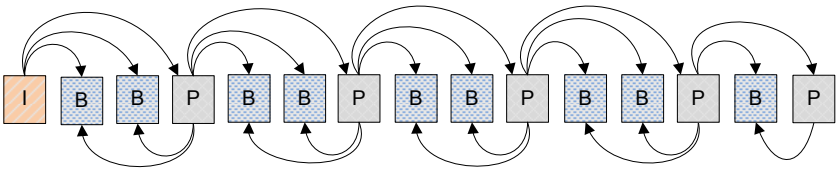

Figure 2.6: Closed GOP Structure

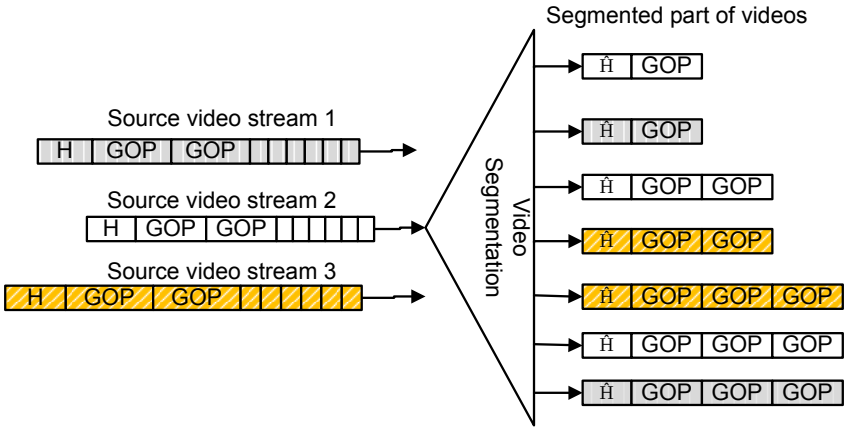

Figure 2.7: Video Segmentation

Closed-GOP

In closed-GOPs, all reference frames belong to the same GOP. Therefore, it represents an independent unit, which can be transcoded without having any frames of other GOPs. However, closed GOP will make compression less efficient. If all GOPs in a video are closed, it is possible to perform segmentation between the B and I-frame. Figure 2.6 shows the structure of closed-GOP where the last frame is a P-frame.

Figure 2.7 shows the segmentation of a source video. Here 'H' in *source video stream* indicates the sequence header of the whole stream and '$\hat{H}$' indicates the segment header. The sequence header 'H' from the source video is slightly different than the '$\hat{H}$' in segmented video. While performing video segmentation a segment header '$\hat{H}$' is attached to each video segment, which contains information such as the stream ID, the segment ID, the number of frames in the segment etc. The stream ID is attached to identify to which stream this segment belongs to and segment ID is attached to place the transcoded segments in order during the merging.

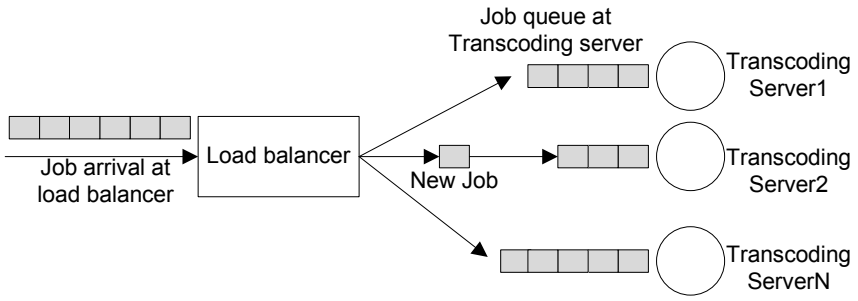

Figure 2.8: Load Balancer

2.2.4 Load Balancer

As shown in Figure 2.8 the *load balancer* employs a task assignment policy, which distributes load on the transcoding servers. In other words, it decides when and to which *transcoding servers* a transcoding job should be routed. It maintains a configuration file, which contains information about *transcoding servers* performing the transcoding operations. As a result of dynamic resource allocation and de-allocation operations, the configuration file is often updated with new information. The *load balancer* serves the jobs in First In, First Out (FIFO) order and has only one input queue. It implements Join-the-Shortest-Queue scheduling policy [38]. The joining decision can be based on the *shortest queue length* or *shortest queue waiting time*. The *shortest queue length* policy selects a *transcoding server* with the shortest queue length while the *shortest queue waiting time* policy selects a *transcoding server* with the least queue waiting time.

2.2.5 Transcoding Server

A transcoding server performs the actual transcoding operations such as bit rate reduction video transcoding, spatial resolution reduction video transcoding, temporal resolution reduction transcoding, and format conversion. The transcoding servers get video segments from the load balancer. Each transcoding server has a queue for new incoming jobs. A transcoding server uses the FIFO scheduling policy to serve the jobs in its queue. It sends the transcoded video segments to the merger.

2.2.6 Master Controller

The *master controller* is a control unit, which is used to control other components in the system architecture. The master controller also acts as a resource allocator. It implements prediction-based dynamic resource allocation and de-allocation algorithms and one or more computation and

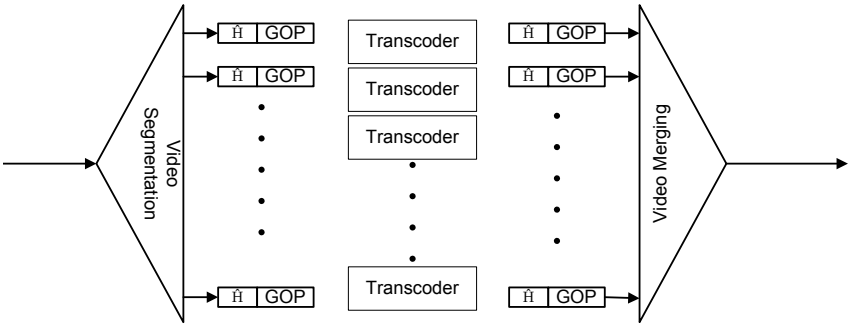

Figure 2.9: A Fork Join Queueing Node

storage trade-off strategies. The resource allocation and de-allocation are mainly based on the target play rate of the video streams and the predicted transcoding rate of the transcoding servers. The resource allocation and de-allocation algorithms are described in detail in papers I and IV of this thesis. The computation and storage trade-off strategy is provided in papers III and IV of this thesis.

2.2.7 Load Predictor

For load prediction, the *master controller* uses the *load predictor* [16]. To perform efficient prediction of the system load, a load tracker is used as a monitoring module that gets the actual play rate of the video streams and the actual transcoding rate of the transcoding servers. It sends this information to a prediction module that estimates the workload characteristics in the near future.

2.2.8 Video Merger

The main purpose of the video merger is to place the transcoded video segments of a video stream in right order. It also modifies the segment header and removes extra information such as the stream ID and the segment ID from the header. The merger and the splitter work like a fork-join queuing node [20] as shown in Figure 2.9.

Chapter 3

Contributions of the Thesis

The following sections provide a description of the research problems and contributions of this thesis.

3.1 Video Segmentation for Distributed Video Transcoding

3.1.1 Video Segmentation

To get better compression, frame prediction is used in different video coding standards such as MPEG-4 and H.264. However, frame predictive coding can not be used indefinitely due to propagation of error, lower video quality, and random access. Furthermore, if some data packets of a reference frame are lost, then the decoding of the remaining frames is impossible. To address the above problems, video sequences usually consist of some independent frames which do not require any other reference frames. These independent frames are termed as Intra(I)-frames. An I-frame, followed by P and B-frames before the next I-frame is termed as GOP.

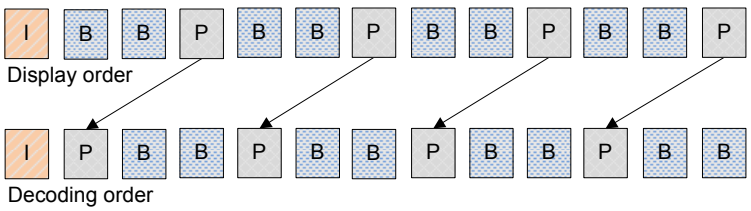

Figure 3.1: Frames Display and Decoding Order

In a GOP, the display order and decoding order of frames is usually different. Figure 3.1 shows that although the original frame sequence is IBBP..., it is decoded as IPBB.... This is due to the fact that B-frames are

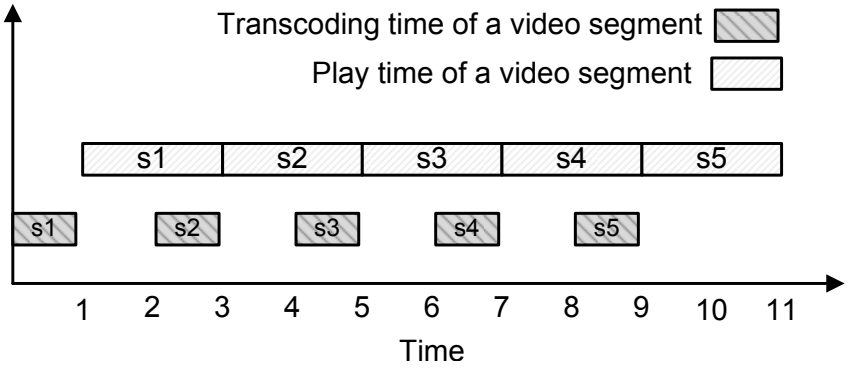

Figure 3.2: Transcoding Time and Play Time

predicted from both future and past frames. As shown in Figure 3.1, the P-frame which is a future frame is decoded first and then used for decoding the B-frame.

The video segmentation in a cloud is performed by the video splitter at the GOP level. The video splitter performs segmentation in such a way that each user gets a smooth video stream from the streaming server. It takes into account the transcoding time and the play time of the video segment. As shown in Figure 3.2, once a video segment is sent for transcoding, the next segment of the same stream is sent after some delay. In the figure, 's' stands for video segment.

The delay between two jobs during segmentation is based on the play time of a video segment and the number of transcoded video frames of the stream in the output buffer. If the transcoded frames are below certain predefined lower threshold, the stream segmentation is performed with zero delay. However, if the transcoded frames are above the threshold, the delay for the next video segment is set equal to the play time of the previous job.

3.1.2 Distributed Video Transcoding

To perform distributed video transcoding, it is possible to build a transcoder from scratch. However, using existing code library with some modifications is more practical. The distributed video transcoder used to perform different experiments in this work is based on the open source FFMPEG [1] library. The original transcoder is modified using Message Passing Interface (MPI) porgramming model. During video segmentation, a segment header is attached to each video segment. This sequence header consists of horizontal frame size, vertical frame size, aspect ratio, frame rate, bit rate, Video Buffering Verifier (VBV), buffer size, and other sequence header in-

[1] http://www.ffmpeg.org/

formation. All these parameters are required by the video transcoder. In addition to these parameters, the stream ID and the segment ID are also attached to help the video merger in identifying the segments. In paper V and VI, different static segmentation methods for bit rate reduction and spatial resolution reduction video transcoding are analysed.

3.1.3 Bit Rate Reduction Transcoding

A video encoder uses either Constant Bit Rate (CBR) or Variable Bit Rate (VBR). In both cases, a compressed video requires bit rate adjustment during transmission over a network. The bit rate adjustment may be required due to limited network bandwidth, due to network congestion, or when a user requires a video at lower bit rate. To reduce bit rate, video transcoding is used. In bit rate reduction video transcoding, video quality is usually degraded [18, 66, 72] while the frame rate and resolution remains the same. Bit rate reduction is obtained with coarser quantization [54, 73, 79]. The coarser quantization results in an increase of the zero quantized coefficients and hence, less bits are required to store the video. To speed-up the transcoding process, original motion vectors can be reused with motion vector refinement [19, 78]. In paper V, bit rate reduction video transcoding is analyzed using static segmentation methods. The work demonstrates that a MPI based transcoder is suitable to perform distributed video transcoding. The segmentation of a video stream with a) equal size having unequal number of intra frames and b) unequal size having equal number of intra frames is performed. The paper also provides results for start-up time of a video and the results indicate that, it is possible to get very short start-up times with suitable video segmentation.

3.1.4 Spatial Resolution Reduction Transcoding

In spatial resolution reduction transcoding, both the frame resolution and the bit rate are reduced. The main operation in this transcoding is to downscale the transform blocks using averaging or some other suitable method [48]. The spatial resolution reduction transcoding requires to calculate new motion vectors from existing motion vectors using either motion vector averaging, weighted mean, or median [48]. In paper VI, different methods of video segmentation are analyzed to perform spatial resolution reduction video transcoding. The main purpose is to analyze different static segmentation methods to obtain speed-up in the transcoding process. The segmentation methods which are analyzed include: (a) each segment has equal size, (b) each segment has equal number of frames, and (c) each segment has equal number of GOPs. To determine which segmentation method is better, the standard deviation of the transcoding time is com-

puted. The results indicate that equal number of GOPs approach for video segmentation is more suitable among these three segmentation methods.

3.1.5 Related Works

A video codec such as H.264 has high computational complexity and transcoding it into another codec requires high performance processing architecture. Future video codecs such as High Efficiency Video Coding (HEVC) requires more processing as compared with the H.264 [69]. Thus it is essential to perform video processing in a parallel or distributed computing environment. This problem has been studied in the past few years [42]. A video transcoding operation can be performed in parallel by using the data-level decomposition or the task-level methods. The data-level parallelism of video coding algorithms such as H.264 [70, 80] is further possible by using either coarse-grain parallelism at the slice-level [36], frame-level [61] or with fine-grain approach at macroblock level [51] and tile level [55]

Frame-level parallelism: In frame-level parallelism multiple frames can be decoded at the same time provided that the inter frames dependencies are satisfied [51]. It has been implemented in popular encoders and decoders such as FFMPEG [2] and x264 [3]. The frame level parallelism may require more memory. If there is high motion in a video then due to long motion vectors, there is little parallelism. In addition to that, due to different processing times of frames, the workload may be imbalanced. To perform frame level parallelism in a cloud computing environemnt, the high latency is a bottleneck. Therefore, this thesis did not used frame-level parallelism approach.

Slice-level parallelism: A video frame may consist of a number of slices. A slice is an independent unit in a frame [80]. The parallelism depends on the total number of slices in a frame. A bitstream having only one slice per frame will have no slice level parallelism [62]. The slice-level parallelism is useful while the transcoding of a single frame is under consideration. However, due to interframe dependencies, the slice-level parallelism has very limited scalability. Therefore, it is not suitable for video transcoding in a cloud computing environment.

Macroblock-level parallelism: Wavefront approach [75] can reconstruct macroblocks in parallel provided that the macroblocks are independent. Many-core architectures are suitable platforms for macroblock-level parallelism [10]. However, entropy coding can only be parallelized at frame or slice level [24]. As the macroblock-level parallelism has very high dependencies and it is only suitable for many-core archtectures. Therefore, it is not beneficial to use it in a distributed cloud computing environment.

[2] http://www.ffmpeg.org/
[3] http://x264.nl/

Tiles-level parallelism: The new video coding standard HEVC [4] [56, 57, 69, 81] has tiles which can provide parallelism. With tiles the video frame is divided in independent rectangular parts which can be coded in paralell [34]. The title-level parallelism is not available in several existing video codecs such as H.264 and MPEG-4. Therefore, this thesis did not used it.

In [8] equal-partition scheme is used to parallelize motion estimation of a video encoder using the Single Program Multiple Data (SPMD) programming paradigm. To reduce the communication overheads, the frames of a video are partitioned in overlapped fashion. However, this scheme may send some unnecessary information which is undesirable. Communication overheads in fine-grain parallel implementation of video coding algorithms are substantial. As the number of processors grows, it becomes even more challenging to keep scalability of the parallel video coding algorithms at the same level [76, 29]. In terms of load balancing, Independence, scalability, and the CPU utilization, GOP-level parallelism is suitable and it has many advantages over other methods. Therefore, all papers of this thesis used the coarse-grain approach for data distribution and segmented a video at GOP level. There exist several other works, which use the split and merge approach at GOP level for video processing [58, 21]. However, none of these works have compared different video segmentation mechanisms, while the paper VI of this thesis analyzes different types of video segmentation. In [58] authors propose a split and merge architecture to perform video processing that reduces video encoding time by using dynamic provisioning of virtual machines in a cloud. The paper provides a very general overview of the split and merge operations in video processing. It also provides some results, which indicate that the video encoding is possible by using split and merge approach and is faster as compared with the traditional video processing approaches. The paper did not provide any other technical description about the split operation for different types of GOPs such as open-GOP and closed-GOP. The paper also states that a frame cannot co-exist in more than one part of the video. This indicates that the authors are assuming that all videos have only closed-GOPs. However, this assumption does not hold in reality. Most video standards do have open-GOPs and due to interdependencies between frames, the splitting of such videos is only possible by placing redundant frames in the neighboring GOPs.

In [58, 21] authors refer to a term "video fragments" which refers to pieces of a video. While the papers of this thesis use the term "video segments" instead of "video fragments". Technically, a video fragment is a small piece of a video file which can not be further divided into smaller

[4] http://www.vcodex.com/h265.html

parts. While a video segment is a piece of video, which can be further divided into smaller parts. In other words a video segment can be divided further into fragments. In [35] a Hadoop-based cloud is used to perform video transcoding. This paper is rather experimental and does not discuss research issues such as resource allocation, admission control, storage and computation trade-off in the cloud.

3.2 Resource Allocation and De-allocation

Since video transcoding workloads are known to vary dynamically with time [28], it is challenging to have an accurate estimate of such workloads. The resource allocation using static methods may lead to either over provisioning or under provisioning of the resources. The over provisioning will increase the system cost and underutilization of resources, while under provisioning will cause poor performance with respect to QoS requirements. One possible solution is to allocate resources dynamically based on the workload. Paper I of this thesis presents dynamic resource allocation algorithms, which are essential to provide cost-efficient video transcoding service while maintaining satisfactory QoS.

In this work, an elastic cluster of video transcoding servers is used in which VMs are added and removed on-demand. To provision and terminate VMs, the target play rate and the video transcoding rate are taken into account. The VM provisioning takes some time [15], to avoid deteriorated performance, the resource allocation algorithms minimize oscillations in number of VMs [82, 43]. The minimum oscillations are possible with a delay in new VM allocation operations and terminating only those VMs whose renting period is near to completion.

3.2.1 Resource Allocation

The resource allocation algorithm calculates the target play rate of all streams and the total transcoding rate of all *transcoding servers*. Then it obtains the predicted total transcoding rate from the *load predictor*. It also takes into account the number of frames in the output buffer. Moreover, it checks if the target play rate exceeds the predicted transcoding rate while the buffer size falls below its lower threshold. In this case it chooses to allocate resources by provisioning one or more VMs. The resource allocation algorithm is provided as algorithm 1 in paper I and IV of this thesis.

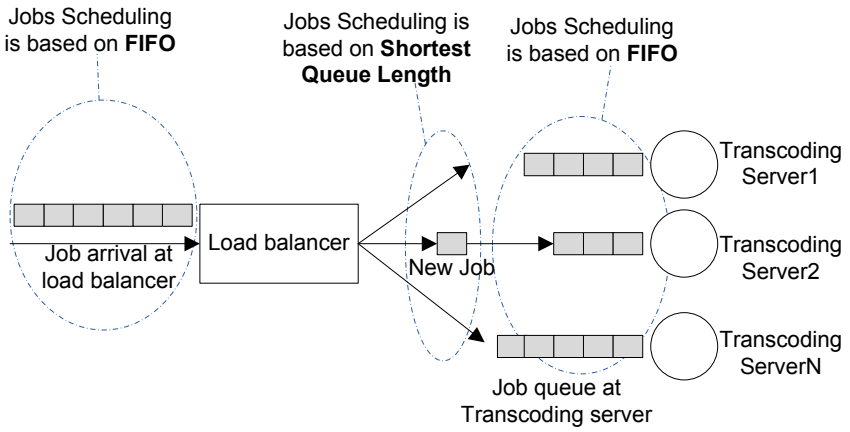

Figure 3.3: Job Scheduling Policies

3.2.2 Resource De-allocation

The main goal of resource de-allocation algorithm is to terminate VMs, which are underutilized. The algorithm takes the target play rate and total transcoding rate of all streams in the system to make VM termination decisions. It will terminate VMs if the predicted transcoding rate exceeds the play rate and if the buffer has more frames than the upper threshold value.

For cost-efficiency, the algorithm terminates only those servers, which are near the completion of their renting period. The resource de-allocation algorithm is provided as algorithm 2 in paper I and IV of this thesis.

3.2.3 Job Scheduling Based on the Shortest Queue Length

The proposed video transcoding architecture in the cloud computing environment uses video segmentation at GOP level. After a video is segmented, each segment is an independent unit and transcoded seperately. Consider n non-preemptive transcoding jobs J_i (i=1,2,....,n) with transcoding times T_i (i=1,2,....,n) to be transcoded on m identical virtual machines VM_1, VM_2,.....,VM_m. The jobs i=1,2,....,n also have precedence constraints. These jobs are created by the spliter and the job scheduling is performed based on queue length of individual transcoding servers. The precedence constraints are maintained by both the spliter and the merger while job scheduler and transcoding servers simply use FIFO policy. Figure 3.3 shows different jobs scheduling policies used for testing the resource allocation algorithms.

3.2.4 Load Prediction Models

The load prediction models are used to predict the system load in advance based on the past load. The load prediction models such as [11, 12, 64] are used to predict the load in web-based systems and can be used to predict load for a *transcoding service*.

The two-step load prediction approach was proposed by Andreolini and Casolari [11] to predict the load in real-time. It has a Load Tracker (LT) and a Load Predictor (LP). The LT keeps regular view of load behavior after filtering out the noise in the raw data [11]. The LT used in this work is based on the Exponential Moving Average (EMA) model. The LP gets input from LT and produces the future load value [11]. Andreolini and Casolari [11] and Saripalli et al. [64] used linear regression of only two LT values, which are the first and the last values in the past time window. Ashraf et al. [16] used simple linear regression model [52], which takes into account all LT values in the past time window. Paper I of this thesis adopts Ashraf et al. [16] approach for load prediction.

3.2.5 Related Works

The dynamic allocation of resources is used in the context of different fields of computer science such as operating systems, storage systems, network systems, web based systems, and data centers. Ardagna et al. [13] worked on capacity allocation algorithm and proposed a distributed algorithm to manage SaaS cloud system. The proposed algorithm is used for capacity allocation for different applications. The algorithm considers some parameters such as IaaS cost, SaaS provider's revenues, and QoS requirements of users in terms of response time of requests. The resource allocation algorithm is based on the predicted future load for applications and performance of VMs. The main aim of Ardagna's work is to maximize the profit. It considers the incoming workload and an average utiliztion of the VMs. For a video transcoding service, the incoming workload is not a suitable parameter to make VM provisioning decisions. It is the target play rate and the achieved transcoding rate which are more important. The work presented in paper I of this thesis considers both the target play rate and the achieved transcoding rate to make provisioning decisions. The work on performance model approaches include TwoSpot [82], which aims to reduce the platform dependency of applications in a PaaS cloud. The proposed architecture is having front-end, app-server, controller, and master. The front-end provides the similar services, which are provided by a web server. The app-server works as an application container. The controller is used to manage app-server instances and the master is used for load balancing. The work supports automatic scaling and hosting of web

applications. The scaling is based on Central Processing Unit (CPU) utilization, memory utilization, number of running applications, and number of requests per second. All these parameters are more suitable for web applications to make resource allocation decisions. For a video transcoding service, above parameters are not suitable to determine the number of VMs required at a time. Hu et al. [40] worked on server allocation and job scheduling algorithm. They proposed a heuristic based algorithm to determine the minimum number of servers required at a specific time, based on parameters such as predicted arrival rate and service rate. Chieu et al. [25] worked on an approach that scales servers for a web application based on user sessions. Their approach uses a monitoring agent, which periodically forwards number of active user sessions to the service monitor system. The servers are provisioned if the number of active user sessions exceeds an upper threshold. A server is removed if it does not have active sessions. To determine a suitable upper threshold value of the user sessions is the main problem with this approach. Iqbal et al. [43] proposed a VM resource allocation approach based on load prediction for read intensive multi-tier web applications. The VM provisioning decisions are based on the CPU utilization and response time of the requests. The proposed approach performs a check after a certain time interval for under-utilized VMs and terminates if it finds any under-utilized server. Dutreilh et al. [31] approach is based on a fixed gain controller. The controller is used to scale-up and scale-down VMs under varying loads. Their work is based on a comparison of static threshold-based and reinforcement learning techniques. None of these works covers the video transcoding in cloud computing while paper I of this thesis covers resource allocation for video transcoding in the cloud.

3.3 Admission Control

The VM resource allocation allows to scale the transcoding service in a cloud. However, the variation in transcoding load may cause servers to be overloaded resulting in deteriorated service to the users. To avoid such situations, paper II of this thesis provides a Stream Based Admission Control and Scheduling (SBACS) approach for elastic tier of video transcoding servers. The scheme provides admission control on new incoming streams. The admission control decisions are mainly based on the work load on servers. A two-step load prediction mechanism is also used to predict the load on servers [11]. The admission controller can accept new streams, reject new streams, and defer new streams. The deferment is an act to delay or postpone for some time. For admitted streams, the load balancer uses temporal resolution reduction to prevent transcoding *jitters*.

3.3.1 Stream-Based Admission Control with Per Stream Admission

SBACS implements a per stream admission control to prevent over-admission of incoming streams. A server can have *open*, *closed* and *overloaded* states for new incoming streams based on the load. The admission control algorithm is provided as algorithm 1 in paper II of this thesis. The admission algorithm is activated on the arrival of a new stream or when there is at least one deferred transcoding request. If there is at least one *open* server available then the stream is accepted for transcoding. A server is open if it has less workload in its queue than a predefined lower threshold value.

3.3.2 Stream Deferment Mechanism

The stream deferment mechanism can be used in a scalable cluster of video transcoding servers [16]. The admission controller uses deferment for new incoming streams if it cannot find an *open* server. An *entertainment server* accommodates the deferred streams and sends a wait message to the user. All deferred streams have higher priority as compared with the new incoming streams. The admission controller accepts the deferred streams as soon as a server become less overloaded or new VMs are provisioned. The *entertainment server* also has a maximum capacity limit for deferred streams and if it cannot handle the streams then new streams are rejected.

3.3.3 Jitter Prevention

The admission control can restrict new incoming transcoding requests. However, the server overloading is also possible with existing streams in case of failure of VMs or variable computational requirements of video transcoding operations. The proposed approach uses temporal resolution reduction to prevent *transcoding jitters* in the existing admitted video streams. The proposed algorithm for jitter prevention is provided as algorithm 2 in paper II of this thesis. The jitter prevention algorithm uses the deadline and the response time of individual jobs and takes appropriate actions to further prevent starvation of jobs and overloading of servers. The job starvation is encountered when a process is denied necessary resources. The outcome of jobs starvation in a video transcoding service is *jitter*. The *jitter* in video transcoding is the unavailability of video frames at user-end.

The *jitter* prevention is possible by dropping of frames, which significantly reduces the transcoding time. It is also termed as temporal resolution reduction [78]. Figure 3.4 shows the temporal resolution reduction transcoding, where some B-type frames are dropped.

To prevent *jitter* in a stream, it computes the estimated delivery deadline ED_j, the estimated response time ER_j, and the estimated transcoding

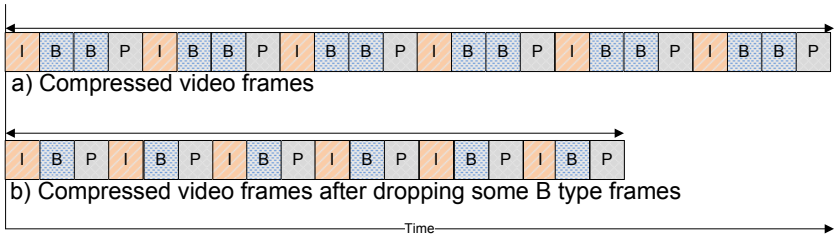

Figure 3.4: Temporal Resolution Reduction Video Transcoding

time ET_j, of each transcoding job j. In case of deadline violation, a certain number of frames is dropped. The number of dropped frames is in proportion to the degree of violation DV_j. The degree of violation DV_j is computed by adding the current clock time $current_{time}$ and ER_j and then subtracting ED_j

$$DV_j = ER_j + current_{time} - ED_j \tag{3.1}$$

The dropping of frames is applied in two situations. First when a server is overloaded and second when there is a deadline violation.

3.3.4 Related Works

Cherkasova and Phaal proposed an admission control approach [23] that is based on the *on-off* control. It computes the server's predicted load for the next time interval from the server's measured loads during a predefined time interval. It then makes a decision to admit or reject new requests during the time interval. Almeida et al. [9] proposed a joint resource allocation and admission control approach for a cloud based environment having virtualized platform. The VMs run web server applications. The main objective is to dynamically adjust the VM capacity limit for new load to get the maximum profit and to satisfy the customers' QoS requirements. Chen et al. [22] proposed Admission Control based on Estimation of Service (ACES) approach. The proposed approach admits requests on the basis of processing time required by a request. In ACES, admission is based on comparison of the available computation capacity and the delay bound. Shaaban and Hillston [65] presented a Cost-Based Admission Control (CBAC) approach, which is based on discount-charge model postponing user requests or charging more cost in a high load period. Ashraf et al. [16] proposed a session-based adaptive admission control for VM, which uses deferment mechanism and provides per session admission control [53]. Huang et al. [41] proposed two admission control schemes to enable Proportional Delay Differentiated Service (PDDS) at the application level. Both admission control schemes are augmented with a prediction mechanism.

The prediction is used to predict the total maximum average waiting time and maximum arrival rate for each priority class. The admission of incoming requests is based on average arrival rate and average waiting time of the user requests. All the above admission control approaches and other related works do not focus on admission control for video transcoding. In this thesis, the proposed work on admission control mainly focuses on video transcoding service in the cloud computing.

3.4 Computation and Storage Trade-off Strategy

In section II of paper I of this thesis, it is stated that the video repository is used to store compressed videos. Furthermore, it states that after each transcoding operation, a copy of transcoded video is stored for a certain amount of time, which is typically two to three weeks. Although, transcoding a video as it is requested the first time and then to store it for subsequent requests, will have low cost as compared with transcoding it on each request. However, to store all transcoded videos for the same amount of time is not a cost-efficient approach. To store an unpopular video for longer times is wastage of disk space. There exists a trade-off between computation cost and storage cost for a video transcoding service.

A cloud such as Amazon provides both computing and storage resources. The computing resources include VMs, which are rented and charged on hourly basis. The storage resources are rented and charged on a monthly basis. The renting cost of computing resources is usually high as compared with the storage resources.

Figure 3.5 shows storage cost of a video. Initially it is much lower than the computation cost. However, with the passage of time, it becomes higher than the computation cost. The proposed strategy in paper III of this thesis estimates the equilibrium point for computation cost and the storage cost, which indicates the time period for which a transcoded video can be stored in the *video repository*.

3.4.1 Proposed Strategy

In this subsection, the proposed computation and storage cost trade-off strategy is presented. Papers III and IV of this thesis also present the strategy. The proposed computation and storage trade-off strategy relies on simple calculations that give, for each transcoded video, a storage cost based on the storage duration and a transcoding cost. It also takes into account the popularity of individual transcoded videos. The estimated computation cost and the estimated storage cost are used to calculate a cost and popularity score S_{τ_i} for each transcoded video τ_i.

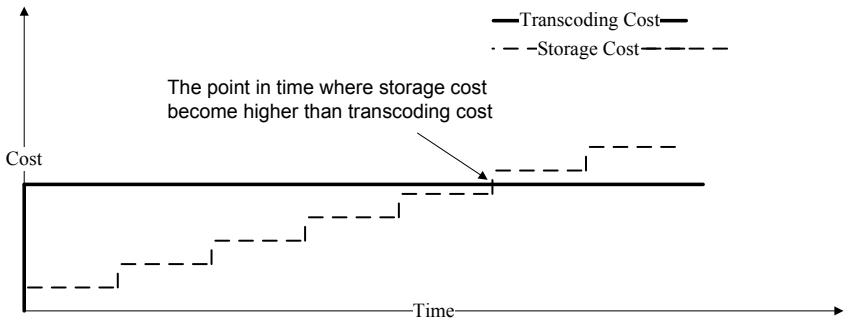

Figure 3.5: Storage vs Transcoding Cost

In the proposed strategy, the storage cost SC_{τ_i} of a transcoded video τ_i is calculated as

$$SC_{\tau_i} = \frac{VSmb_{\tau_i}}{GB_{mb}} * \frac{SC_m}{RP_S} * SD_{\tau_i} \tag{3.2}$$

where $VSmb_{\tau_i}$ is the size of the transcoded video τ_i in megabytes, GB_{mb} is the megabytes to gigabytes conversion factor, SC_m is the monthly storage cost per 1 gigabytes of storage, RP_S is the month to desired time unit conversion factor, and SD_{τ_i} is the length of the storage duration for the transcoded video τ_i.

Similarly, the transcoding cost TC_{τ_i} of a transcoded video τ_i is calculated as

$$TC_{\tau_i} = TT_{\tau_i} * \frac{RC_T}{H_{sec}} \tag{3.3}$$

Where TT_{τ_i} is the transcoding time of τ_i, RC_T is the renting cost of a transcoding server per renting hour, and H_{sec} is the hour to seconds conversion factor, which is used to normalize the computation cost to a per second basis.

Whenever a transcoded video request arrives, the popularity score and video cost are updated. The NS_{τ_i} is the new cost and popularity score, which is the ratio of transcoding cost and storage cost

$$NS_{\tau_i} = \frac{TC_{\tau_i}}{SC_{\tau_i}} \tag{3.4}$$

Finally, the total cost and popularity score S_{τ_i} of a video τ_i is calculated by accumulating the new cost and popularity score NS_{τ_i} of the said video over time. On certain time intervals, the proposed strategy checks if the storage cost is higher than the product of transcoding cost and S_{τ_i}.

$$SC_{\tau_i} > TC_{\tau_i} * S_{\tau_i} \tag{3.5}$$

If the storage cost is higher, then the transcoded video is removed from the video repository.

3.4.2 Related Works

The work on computation and storage trade-off strategy for video transcoding in a cloud computing environment is new. However, there exists some related works in the area of cost-efficient computation and storage trade-off analysis. Adams et al. [6] worked on trading storage for computation to get maximum efficiency. Some result specific factors such as odds and frequency of reuse, reuse lead time, rebuild time, and miss penalty are highlighted in their work. Furthermore, their work also highlights some marginal cost factors including cost of storage, cost of network and cost of computation. These factors can be used to obtain the computation and storage trade-offs. However, their work did not propose any computation and storage cost trade-off strategy. Deelman et al. [30] studied the computation cost and storage cost trade-off for a data-intensive astronomy application using the Amazon EC2 [5] and the Amazon S3 [6] fee structure. Their work showed that the cloud offers a cost-efficient solution for data-intensive applications. This is to store popular datasets in the cloud while re-compute unpopular data-sets on-demand. Their work did not explore several aspects such as the startup cost of application in a cloud, as well as security and data privacy in a cloud. Furthermore, they did not provided any strategy for computation cost and storage cost in the cloud. Nectar system [37] automates the computation and data management in a data-center. In Nectar system, derived datasets are produced as the results of computations. A derived dataset can be stored for future use or can be regenerated on-demand. To avoid unnecessary computation for regenerating a derived dataset, all derived datasets are cached. When the storage space reaches a predefined upper threshold, some less important datasets are removed to free disk space. Nectar system considers the elapsed time of the dataset, the size, the number of times it has been used, and its cumulative computation time. Those datasets are removed, which have the largest cost-to-benefit ratio. Nectar system's proposed computation and storage trade-off strategy is not designed to reduce the total cost of storage and computation. Yuan et al. [85] proposed a Cost Transitive Tournament Shortest Path (CTT-SP) algorithm to obtain the best computation and storage cost trade-off. Yuan et al. [84] also proposed a cost rate based storage strategy [83, 86], which compares storage cost rate and computation cost rate of datasets to decide storage status of a dataset. In contrast, the work in this thesis estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal. Moreover, it estimates video popularity of the individual transcoded videos to differentiate popular videos. The existing strategies were originally proposed for scientific

[5] http://aws.amazon.com/ec2/
[6] http://aws.amazon.com/s3/

datasets and to the best of my knowledge, there is currently no existing computation and storage trade-off strategy for video transcoding service in the cloud. The difference of application domain may play a vital role when determining cost-efficiency of the existing strategies. Therefore, some of the existing strategies may have limited efficacy and little cost-efficiency for video transcoding. . Kathpa et al. [45] analyzed Compute vs. Storage Trade-off for transcoded videos and developed an elimination metric. Their work mainly focuses on the effect of elimination metric by different parameters such as video resolution, codec and container. In the experiments, they used relatively short duration videos and assumed that popularity is the same across the videos. The assumption for same popularity does not hold in reality for different videos. The work in papers III and IV of this thesis is based on the realistic load pattern and do not make such assumptions.

Chapter 4

Description of Papers

The contribution of the thesis is described in the context of the individual research papers, which are presented in part II of this thesis. In this chapter, a summary of the original publications along with the description of the author's contribution is provided.

4.1 Overview of the Papers

4.1.1 Paper I: Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing

In this paper, a novel approach to perform video transcoding in cloud computing is demonstrated along with VM provisioning algorithms. A cloud has VMs, which represent computing resources in a distributed computing environment. The proposed resource allocation approach for video transcoding is based on video segmentation at the GOP level. The video segmentation at GOP level is used to share VM resources among several video streams. As the user load may vary at different times, the approach provides a resource allocation and a resource de-allocation algorithm. The resource allocation algorithm is mainly used to provision VMs at peak load hours in order to provide efficient transcoding service. The resource de-allocation algorithm is used to terminate the VM resources at off-peak hours while there are less transcoding requests. Both resource allocation and de-allocation algorithms use a two-step load prediction method. The two-step load prediction method is based on load tracking using EMA method and load prediction by using a simple linear regression method. To demonstrate the proposed approach, a discrete-event simulation is used involving two different load patterns.

Author's contribution The basic idea presented in this paper was initially developed jointly by both co-authors Fareed Jokhio and Adnan Ashraf while Fareed Jokhio was working as a PhD student under the supervision

of Professor Johan Lilius and Dr. Sébastien Lafond. Both co-authors were responsible for developing this idea. Fareed Jokhio is the main author of this paper. The paper was written jointly by both co-authors Fareed Jokhio and Adnan Ashraf.

4.1.2 Paper II: Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing

This paper provides an admission control approach for new streams in a cloud based video transcoding service. The approach uses load on servers' queues to determine their states. The possible states of servers are open, closed and overloaded. In open state, the servers can accept new streams while in the other case, new streams are *deferred* for some time. To handle *deferred* streams, an entertainment server is used, which also has a maximum capacity. In case the entertainment server is unable to handle the *deferred* streams, the new streams are *rejected*. The work also includes a job scheduling approach, which uses temporal resolution reduction video transcoding to avoid server overloading and transcoding jitter. The video segmentation at GOP level is used for better load balancing and efficient sharing of VMs in a cloud based environment. The results indicate that the proposed approach provides a trade-off between cost and QoS. By having a check on stream admission, the server overloading is avoided and transcoding jitter is also reduced.

Author's contribution The basic idea presented in this paper was initially developed jointly by co-authors Adnan Ashraf, Fareed Jokhio and Tewodros Deneke. Fareed Jokhio was working as a PhD student under the supervision of Professor Johan Lilius and Dr. Sébastien Lafond. All co-authors were responsible for developing this idea. The paper was written jointly by co-authors Adnan Ashraf and Fareed Jokhio.

4.1.3 Paper III: A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud

In a cloud based video transcoding service, there are original videos and transcoded videos. Transcoded videos are obtained after applying the transcoding operation. A transcoded video can be stored for future requests or can be transcoded on-demand. The amount of data available in a video is huge and storing an unpopular transcoded video for a longer time may not be cost-efficient. As video transcoding is a compute-intensive operation, transcoding a popular video for each request will have a very high computation cost. This work presents a computation and storage cost trade-off strategy for transcoded videos in a cloud computing environment. The strategy is based on the computation and storage cost and the popular-

ity of the videos. It finds an optimal point for each transcoded video, which tells how long a transcoded video should be stored in the video repository. The proposed approach is demonstrated in a discrete-event simulation using realistic and semi-synthetic load patters. The proposed strategy is also compared with the existing intuitive strategies and the simulation results indicate that the proposed strategy is more cost-efficient as compared with two other intuitive strategies.

Author's contribution The basic idea presented in this paper was initially developed jointly by co-authors Fareed Jokhio and Dr. Sébastien Lafond. Fareed Jokhio was working as a PhD student under the supervision of Professor Johan Lilius and Dr. Sébastien Lafond. Fareed Jokhio and Adnan Ashraf were responsible for developing the idea and developed software simulations and conducted experiments. Fareed Jokhio is the main author of this paper. The paper was written jointly by both co-authors Fareed Jokhio and Adnan Ashraf.

4.1.4 Paper IV: Cost-efficient dynamically scalable video transcoding in cloud computing

This paper expands the work presented in the papers I and III of this thesis. The resource allocation algorithm presented in the paper I is improved. In addition to transcoding rate and play rate of video streams, the queue loads of transcoding servers are also taken into account to make resource allocation decisions. The job scheduling policy for transcoding servers is also replaced with an improved job scheduling policy. The work presented in the paper III is expanded and the calculation of cost and popularity score is demonstrated by an algorithm. The algorithm is provided as algorithm 3 in the paper IV of this thesis. The popularity score reduction and the procedure of the removal of a video is also demonstrated using an algorithm. The algorithm is provided as algorithm 4 in paper IV.

Author's contribution The basic idea presented in this paper was initially developed jointly by co-authors Fareed Jokhio, Dr. Sébastien Lafond and Adnan Ashraf. Fareed Jokhio was working as a PhD student under the supervision of Professor Johan Lilius and Dr. Sébastien Lafond. Fareed Jokhio and Adnan Ashraf were responsible for developing the idea and developed software simulations and conducted experiments. Fareed Jokhio is the main author of this paper. The paper was written jointly by both co-authors Fareed Jokhio and Adnan Ashraf.

4.1.5 Paper V: Bit Rate Reduction Video Transcoding with Distributed Computing

This paper presents an approach to perform bit rate reduction transcoding by video segmentation. A MPI-based programming model is used to create a video transcoder, which can work in a distributed computing environment and on a multi-core system as well. The video segmentation is performed at GOPs level, which provides significant gain in terms of execution speed-up. The proposed approach is evaluated on a multi-core system with two different video segmentation strategies. The results indicate that the proposed parallelization approach provides a gain in execution time with acceptable video quality.

Author's contribution The basic idea presented in this paper was initially developed jointly by both co-authors Fareed Jokhio and Tewodros Deneke while Fareed Jokhio was working as a PhD student under the supervision of Professor Johan Lilius and Dr. Sébastien Lafond. Both co-authors were responsible for developing this idea. Fareed Jokhio is the main author of this paper. The paper was written jointly by both co-authors Fareed Jokhio and Tewodros Deneke.

4.1.6 Paper VI: Analysis of Video Segmentation for Spatial Resolution Reduction Video Transcoding

In this paper, three different video segmentation strategies are analyzed to perform spatial resolution reduction video transcoding. A MPI-based distributed transcoder is used in which video segmentation is performed at GOPs level. The segmentation strategies used in this work are: (1) each segment has equal size, (2) each segment has equal number of frames, and (3) each segment has equal number of GOPs. The results indicates that the performance of *equal number of GOPs* segmentation strategy is better as compared with two other segmentation strategies.

Author's contribution The basic idea presented in this paper was initially developed jointly by both co-authors Fareed Jokhio and Tewodros Deneke while Fareed Jokhio was working as a PhD student under the supervision of Professor Johan Lilius and Dr. Sébastien Lafond. Both co-authors were responsible for developing this idea. Fareed Jokhio is the main author of this paper. The paper was written jointly by both co-authors Fareed Jokhio and Tewodros Deneke.

Chapter 5

Discussion

As mentioned in section 1.1, the first goal of this thesis was to analyze video segmentation methods for video transcoding in a distributed cloud computing environment. In papers V and VI, different video segmentation methods are analyzed for various video transcoding mechanisms. The analysis of video start-up time is provided in paper V. The paper states that a dynamic video segmentation method can be used to achieve very short video start-up time.

The second goal of this thesis was to analyze cost-efficient VM provisioning algorithms. Paper I of this thesis present prediction based dynamic resource allocation and de-allocation algorithms. The algorithms take into account the accumulative transcoding rate and play rate of all video streams. To obtain the achievable throughput, underutilized servers are terminated by the resource de-allocation algorithm.

The third goal of this thesis was to avoid server overloading by applying admission control. Paper II presents a stream based admission control algorithm with deferment mechanism. If all the servers are in closed states and unable to accept new transcoding requests then the admission control uses an entertainment server, which suspends the new incoming transcoding requests. New requests are rejected if the entertainment server is also in closed state.

The results in paper II of this thesis indicate that there were zero rejected streams. The work in paper II of this thesis also includes a job scheduling algorithm, which is used to avoid transcoding jitter. The jitter avoidance is performed by dropping frames. Furthermore, the results in paper II of this thesis indicate that the transcoding jitter is reduced. If the jitter is avoided by dropping frames, then there is a trade-off between the number of dropped frames and the cost of server provisioning. In the experiments, a limited number of frames were dropped to get the reduced transcoding jitter.

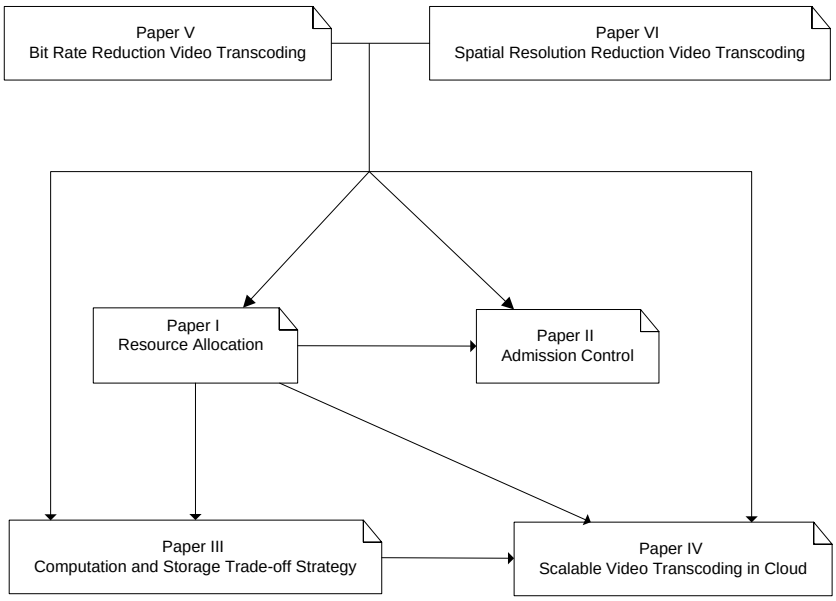

Figure 5.1: Relationship of the Previous Publications

The fourth and final goal of this thesis was to avoid repetition of transcoding operations by storing a transcoded video as long as it is cost-efficient to store it. Paper III provides a cost-efficient computation and storage trade-off strategy. The proposed strategy checks and removes a transcoded video if it is expensive to keep the transcoded video in the repository. Further improvements are possible by using heuristic techniques to determine which transcoded videos will be removed and removing those transcoded videos even before the point in time obtained by the proposed computation and storage trade-off strategy.

Figure 5.1 shows the relationship of the original publications that are attached in part II of this thesis. Papers I, II, III, and IV describe the video transcoding in the cloud computing environment while papers V and VI presents the work on video segmentation and distributed video transcoding using an MPI based video transcoder. Paper I includes the basic idea of the system architecture of a cloud based video transcoding service. It provides algorithms for allocation and de-allocation of VMs. The work presented in paper I is further extended in paper II, which includes the evaluation of stream level admission control algorithm to restrict new transcoding requests in order to avoid the overloading of servers during high load. A job scheduling algorithm is also provided to minimize transcoding jitters. Paper III of this thesis extends the work presented in paper I and provides a computation and storage cost trade-off strategy, which determines how long a transcoded video can be stored or how often it can be transcoded. Paper

IV is an extension of paper I and III. This paper includes improved resource allocation and de-allocation algorithms and the proposed computation and storage cost based trade-off strategy. Finally papers V and VI include the work on video segmentation methods for video transcoding mechanisms.

Chapter 6

Conclusion and Future Work

To perform efficient video transcoding in a cloud computing environment, a GOP level video segmentation approach was adopted. As the load conditions may change at different times, the proposed approach uses an elastic computing environment in a cloud where VM resources can be provisioned on-demand. The VM allocation algorithms are provided to scale video transcoding service in an IaaS cloud. The resource allocation algorithms use a two-step load prediction method. The load prediction uses a load tracker using EMA method, which provides a smooth trend of the load and a linear regression model for predicting the future load. The proposed VM allocation approach is demonstrated with a discrete-event simulation using two different experiments having synthetic load patterns. The experimental results indicate that the proposed approach provides cost-efficient VM allocation.

The resource allocation work is augmented with a stream-based admission control. The admission control is essential in bursty load conditions where servers overloading can degrade the QoS requirements of end-users. The admission control decisions are based on the transcoding servers' queue load. A server becomes closed for new streams if its queue loads exceeds a predefined upper-threshold value. As the resource allocation is possible in a cloud computing environment, the admission control approach uses a deferment mechanism instead of directly rejecting new incoming requests. To handle deferred requests, an entertainment server is used. The entertainment server also has an upper limit of handling deferred requests. If it exceeds that limit, the new streams are rejected. The admission control can restrict the new coming streams, but the existing load may also cause the server to overload in case of failure of one or more VM resources. To handle such situations, a jitter preventation algorithm is also provided. The transcoding *jitter* is avoided with a frame dropping scheme. The proposed admission control approach and job scheduling algorithm are demonstrated

with a discrete-event simulation using different synthetic load patterns. The results show that the proposed approach provides a good trade-off between cost and QoS. The approach prevents servers overloading and reduces transcoding *jitter*.

The work also includes a computation cost and storage cost trade-off strategy for video transcoding in a cloud. The proposed strategy is based on the computation cost of a transcoded video, its storage cost, and popularity. It is cost-efficient to store popular transcoded videos and perform on-demand transcoding for unpopular transcoded videos. The proposed strategy helps in making decisions on how long a transcoded video should be stored or how frequently it should be transcoded. A discrete-event simulation is used to demonstrate the proposed strategy. For the sake of comparison, the proposed strategy is compared with two existing intuitive strategies. The results show that the proposed strategy is more cost-efficient as compared with the two intuitive strategies.

This thesis also proposes video segmentation strategies for bit rate reduction and spatial resolution reduction video transcoding. The evaluation of the proposed strategies is performed using an MPI-based video transcoder.

The future work of this thesis includes a better computation cost and storage cost trade-off strategy, which uses either prediction or lotteries to remove a video even before the equilibrium point to save storage cost. The storage locality of the transcoded videos is also another open problem. A video, which is more frequently accessed in a certain region, needs to be stored on a nearby storage server. There is a clear trade-off between the network bandwidth and the storage of the multiple copies of the same transcoded video on different locations. A better resource allocation policy can also be developed by taking into account some parameters such as number of transcoding requests, CPU and memory utilization, I/O load of the system. Some other research issues such as failure of components, and network congestion can also be a potential future work of this thesis. In addition, the optimal size of a GOP for a system as a whole to provide the best quality and user experience under different load patterns and video qualities can also be analyzed.

Bibliography

[1] MPEG-l video group, Information technology-coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s: Part 2-Video, ISO/lEC 11172-2, International Standard, 1993.

[2] ITU-T experts group on very low bitrate visual telephony, ITU-T Recommendation H.263: Video coding for low bitrate communication, December 1995.

[3] MPEG-2 video group, Information technology-generic coding of moving pictures and associated audio: Part 2-Video, ISO/lEC 13818-2, International Standard, 1995.

[4] ITU-T experts group on very bitrate visual telephony, ITU-T Recommendation H.263 version 2: Video coding for low bitrate communication, January 1998.

[5] MPEG-4 video group, Generic coding of audio-visual objects: Part 2-Visual, ISO/IEC JTCI/SC29/WG11 N1902, FDIS of ISO/lEC 14496-2, Atlantic City, November 1998.

[6] Ian F. Adams, Darrell D. E. Long, Ethan L. Miller, Shankar Pasupathy, and Mark W. Storer. Maximizing efficiency by trading storage for computation. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[7] Gurhanli Ahmet, Chung-Ping Charlie, and Hung Shih-Hao. Coarse grain parallelization of h.264 video decoder and memory bottleneck in multi-core architectures. *International Journal of Computer Theory and Engineering*, 3(3):375–381, 2011.

[8] Shahriar M. Akramullah, Ishfaq Ahmad, and Ming L. Liou. Parallelization of mpeg-2 video encoder for parallel and distributed computing systems, 1996.

[9] Jussara Almeida, Virgílio Almeida, Danilo Ardagna, Italo Cunha, Chiara Francalanci, and Marco Trubian. Joint admission control and resource allocation in virtualized servers. *J. Parallel Distrib. Comput.*, 70(4):344–362, April 2010.

[10] M. Alvarez-Mesa, B. Juurlink, C.C. Chi, A. Azevedo, C. Meenderinck, and A. Ramirez. *Scalable Parallel Programming Applied to H.264/AVC Decoding*. SpringerBriefs in Computer Science. Springer, 2012.

[11] Mauro Andreolini and Sara Casolari. Load prediction models in web-based systems. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, valuetools '06, New York, NY, USA, 2006. ACM.

[12] Mauro Andreolini, Sara Casolari, and Michele Colajanni. Models and framework for supporting runtime decisions in web-based systems. *ACM Trans. Web*, 2(3):17:1–17:43, July 2008.

[13] Danilo Ardagna, Carlo Ghezzi, Barbara Panicucci, and Marco Trubian. Service provisioning on the cloud: Distributed algorithms for joint capacity allocation and admission control. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2010.

[14] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[15] Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres. Feedback control algorithms to deploy and scale multiple web applications per virtual machine. *In 38th Euromicro Conference on Software Engineering and Advanced Applications*, 2012.

[16] Adnan Ashraf, Benjamin Byholm, and Ivan Porres. A session-based adaptive admission control approach for virtualized application servers. In *Utility and Cloud Computing (UCC), 5th IEEE/ACM International Conference on*, pages 65–72, 2012.

[17] P. A.A. Assuncao and M. Ghanbari. A frequency-domain video transcoder for dynamic bit-rate reduction of mpeg-2 bit streams. *IEEE Trans. Cir. and Sys. for Video Technol.*, 8(8):953–967, December 1998.

[18] P.A.A. Assuncao and M. Ghanbari. Transcoding of single-layer mpeg video into lower rates. *Vision, Image and Signal Processing, IEE Proceedings* -, 144(6):377 –383, dec 1997.

[19] N. Bjork and C. Christopoulos. Transcoder architectures for video coding. *Consumer Electronics, IEEE Transactions on*, 44(1):88 –98, feb 1998.

[20] A. Bouillard and B. Gaujal. Perfect sampling for fork-join networks. Technical Report 0, LIP, 2005.

[21] Karin Breitman, Markus Endler, Rafael Pereira, and Marcello Azambuja. When tv dies, will it go to the cloud? *Computer*, 43:81–83, April 2010.

[22] Xiangping Chen, Huamin Chen, and Prasant Mohapatra. ACES: An efficient admission control scheme for QoS-aware web servers. *Computer Communications*, 26(14):1581 – 1593, 2003.

[23] L. Cherkasova and P. Phaal. Session-based admission control: a mechanism for peak load management of commercial web sites. *Computers, IEEE Transactions on*, 51(6):669 –685, jun 2002.

[24] Chi Ching Chi and Ben Juurlink. A qhd-capable parallel h.264 decoder. In *Proceedings of the international conference on Supercomputing*, ICS '11, pages 317–326, New York, NY, USA, 2011. ACM.

[25] T.C. Chieu, A. Mohindra, A.A. Karve, and A. Segal. Dynamic scaling of web applications in a virtualized cloud computing environment. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 281 –286, oct. 2009.

[26] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: video coding at low bit rates. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(7):849–866, 1998.

[27] R.V. Cox, B.G. Haskell, Y. LeCun, B. Shahraray, and L. Rabiner. On the applications of multimedia processing to communications. *Proceedings of the IEEE*, 86(5):755–824, 1998.

[28] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *IEEE/ACM Trans. Netw.*, 5(6):835–846, December 1997.

[29] Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35:1–35:44, October 2011.

[30] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: the montage

example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press.

[31] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck. From data center resource allocation to control theory and back. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 410 –417, july 2010.

[32] T. Ebrahimi and M. Kunt. Visual data compression for multimedia applications. *Proceedings of the IEEE*, 86(6):1109–1125, 1998.

[33] Jean-Claude Fernandez and Manuel P. Malumbres. A parallel implementation of h.26l video encoder (research note). In Burkhard Monien and Rainer Feldmann, editors, *Euro-Par*, volume 2400 of *Lecture Notes in Computer Science*, pages 830–833. Springer, 2002.

[34] A. Fuldseth, M. Horowitz, S. Xu, and M. Zhou. Tiles. Technical Report JCTVC-E408, March 2011.

[35] Adriana Garcia, Hari Kalva, and Borko Furht. A study of transcoding on cloud environments for video content delivery. In *Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing*, MCMC '10, pages 13–18, New York, NY, USA, 2010. ACM.

[36] Amit Gulati and George Campbell. Efficient mapping of the h.264 encoding algorithm onto multiprocessor dsps. 5683:94–103, 2005.

[37] Pradeep Kumar Gunda, Lenin Ravindranath, Chandramohan A. Thekkath, Yuan Yu, and Li Zhuang. Nectar: automatic management of data and computation in datacenters. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[38] Varun Gupta, Mor Harchol-balter, Karl Sigman, and Ward Whitt. Analysis of join-the-shortest-queue routing for web server farms. In *In PERFORMANCE 2007. IFIP WG 7.3 International Symposium on Computer Modeling, Measurement and Evaluation*, 2007.

[39] B.G. Haskell, P.G. Howard, Y.A. LeCun, A. Puri, J. Ostermann, M.R. Civanlar, L. Rabiner, L. Bottou, and P. Haffner. Image and video coding-emerging standards and beyond. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(7):814–837, 1998.

[40] Ye Hu, Johnny Wong, Gabriel Iszlai, and Marin Litoiu. Resource provisioning for cloud computing. In *Proceedings of the 2009 Conference*

of the Center for Advanced Studies on Collaborative Research, CASCON '09, pages 101–111, New York, NY, USA, 2009. ACM.

[41] Chenn-Jung Huang, Chih-Lun Cheng, Yi-Ta Chuang, and Jyh-Shing Roger Jang. Admission control schemes for proportional differentiated services enabled internet servers using machine learning techniques. *Expert Systems with Applications*, 31(3):458 – 471, 2006.

[42] Zixia Huang, Chao Mei, Li Erran Li, and Thomas Woo. Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 201–205. IEEE, 2011.

[43] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871 – 879, 2011.

[44] Kou-Sou Kan and Kuo-Chin Fan. Video transcoding architecture with minimum buffer requirement for compressed mpeg-2 bitstream. *Signal Processing*, 67(2):223–235, 1998.

[45] Atish Kathpal, Mandar Kulkarni, and Ajay Bakre. Analyzing compute vs. storage tradeoff for video-aware storage efficiency. In *Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'12, pages 13–13, Berkeley, CA, USA, 2012. USENIX Association.

[46] H. Kato, Y. Takishima, and Y. Nakajima. A fast dv to mpeg-4 transcoder integrated with resolution conversion and quantization. *IEEE Trans. Cir. and Sys. for Video Technol.*, 17(1):111–119, January 2007.

[47] Gertjan Keesman, Robert Hellinghuizen, Fokke Hoeksema, and Geert Heideman. Transcoding of mpeg bitstreams. *Signal Processing: Image Communication*, 8(6):480–500, 1996.

[48] Zhijun Lei and Nicolas D. Georganas. H.263 video transcoding for spatial resolution downscaling. In *ITCC*, pages 425–430. IEEE Computer Society, 2002.

[49] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi-Li Zhang, and Yafei Dai. Cloud transcoder: Bridging the format and resolution gap

between internet videos and mobile devices. In *The 22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2012.

[50] Ming Liou. Overview of the px64 kbit/s video coding standard. *Commun. ACM*, 34(4):59–63, April 1991.

[51] Cor Meenderinck, Arnaldo Azevedo, Ben Juurlink, Mauricio Alvarez Mesa, and Alex Ramirez. Parallel scalability of video decoders. *J. Signal Process. Syst.*, 57(2):173–194, November 2009.

[52] D.C. Montgomery, E.A. Peck, and G.G. Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2012.

[53] Sireesha Muppala and Xiaobo Zhou. Coordinated session-based admission control with statistical learning for multi-tier internet applications. *Journal of Network and Computer Applications*, 34(1):20 – 29, 2011.

[54] Y. Nakajima, H. Hori, and T. Kanoh. Rate conversion of mpeg coded video by re-quantization process. In *Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3 - Volume 3*, ICIP '95, pages 3408–, Washington, DC, USA, 1995. IEEE Computer Society.

[55] Jongho Nang and Junwha Kim. An effective parallelizing scheme of mpeg-1 video encoding on ethernet-connected workstations. In *Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference (APDC '97)*, APDC '97, pages 4–, Washington, DC, USA, 1997. IEEE Computer Society.

[56] Jens-Rainer Ohm and Gary J. Sullivan. High efficiency video coding: The next frontier in video compression [standards in a nutshell]. *IEEE Signal Process. Mag.*, 30(1):152–158, 2013.

[57] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. Comparison of the coding efficiency of video coding standards - including high efficiency video coding (hevc). *IEEE Trans. Circuits Syst. Video Techn.*, 22(12):1669–1684, 2012.

[58] Rafael Pereira, Marcello Azambuja, Karin Breitman, and Markus Endler. An architecture for distributed high performance video processing in the cloud. In *IEEE CLOUD*, pages 482–489. IEEE, 2010.

[59] J. Rhoton and R. Haukioja. *Cloud Computing Architected: Solution Design Handbook*. Recursive Press, 2011.

[60] A. Rodrguez, A. Gonzlez, and Manuel P. Malumbres. Hierarchical parallelization of an h.264/avc video encoder. In *PARELEC*, pages 363–368. IEEE Computer Society, 2006.

[61] A. Rodriguez, A. Gonzalez, and M. P. Malumbres. Hierarchical parallelization of an h.264/avc video encoder. In *Proceedings of the international symposium on Parallel Computing in Electrical Engineering*, PARELEC '06, pages 363–368, Washington, DC, USA, 2006. IEEE Computer Society.

[62] Michael Roitzsch. Slice-balancing h.264 video encoding for improved scalability of multicore decoding. In *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, EMSOFT '07, pages 269–278, New York, NY, USA, 2007. ACM.

[63] S Sankaraiah, H S Lam, C Eswaran, and Junaidi Abduallah. Gop level parallelism on h.264 video encoder for multicore architecture. In *International Proceedings of Computer Science and Information Technolog*, volume 7, pages 127–132. IACSIT Press, 2011.

[64] P. Saripalli, G.V.R. Kiran, R.R. Shankar, H. Narware, and N. Bindal. Load prediction and hot spot detection models for autonomic cloud computing. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 397 –402, dec. 2011.

[65] Yussuf Abu Shaaban and Jane Hillston. Cost-based admission control for internet commerce QoS enhancement. *Electronic Commerce Research and Applications*, 8(3):142 – 159, 2009.

[66] T. Shanableh and M. Ghanbari. Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. *Multimedia, IEEE Transactions on*, 2(2):101 –110, jun 2000.

[67] Guobin Shen, Y. He, Wanyong Cao, and Shipeng Li. Mpeg-2 to wmv transcoder with adaptive error compensation and dynamic switches. *IEEE Trans. Cir. and Sys. for Video Technol.*, 16(12):1460–1476, December 2006.

[68] K. Stuhlmuller, N. Farber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *IEEE Journal on Selected Areas in Communications*, 18:1012–1032, 2000.

[69] Gary J. Sullivan, Jens-Rainer Ohm, Woojin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Trans. Circuits Syst. Video Techn.*, 22(12):1649–1668, 2012.

[70] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra. The h.264/avc advanced video coding standard: Overview and introduction to the fidelity range extensions. In *SPIE conference on Applications of Digital Image Processing XXVII*, pages 454–474, 2004.

[71] Gary J. Sullivan and Thomas Wiegand. Video compression – from concepts to the h.264/avc standard. In *PROCEEDINGS OF THE IEEE*, pages 18–31, 2005.

[72] Huifang Sun, W. Kwok, and J.W. Zdepski. Architectures for mpeg compressed bitstream scaling. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(2):191 –199, apr 1996.

[73] M.-T. Sun, T.-D. Wu, and J.-N. Hwang. Dynamic bit allocation in video combining for multipoint conferencing. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 45(5):644 –648, may 1998.

[74] Yap-Peng Tan and Haiwei Sun. Fast motion re-estimation for arbitrary downsizing video transcoding using h.264/avc standard. *IEEE Trans. on Consum. Electron.*, 50(3):887–894, 2004.

[75] Erik B. Van Der Tol, Egbert G. T. Jaspers, and Rob H. Gelderblom. Mapping of h.264 decoding on a multiprocessor architecture. pages 707–718, 2003.

[76] Bharadwaj Veeravalli, Xiaolin Li, and Chi Chung Ko. On the influence of start-up costs in scheduling divisible loads on bus networks. *IEEE Trans. Parallel Distrib. Syst.*, 11(12):1288–1305, December 2000.

[77] Toby Velte, Anthony Velte, and Robert Elsenpeter. *Cloud Computing, A Practical Approach*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2010.

[78] A. Vetro, C. Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *Signal Processing Magazine, IEEE*, 20(2):18 – 29, mar 2003.

[79] O. Werner. Requantization for transcoding of mpeg-2 intraframes. *Image Processing, IEEE Transactions on*, 8(2):179 –191, feb 1999.

[80] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[81] Thomas Wiegand, Jens-Rainer Ohm, Gary J. Sullivan, Woojin Han, Rajan L. Joshi, Thiow Keng Tan, and Kemal Ugur. Special section on

the joint call for proposals on high efficiency video coding (hevc) standardization. *IEEE Trans. Circuits Syst. Video Techn.*, 20(12):1661–1666, 2010.

[82] Andreas Wolke and Gerhard Meixner. TwoSpot: A cloud platform for scaling out web applications dynamically. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin / Heidelberg, 2010.

[83] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12, 2010.

[84] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. Computation and storage trade-off for cost-effectively storing scientific datasets in the cloud. In Borko Furht and Armando Escalante, editors, *Handbook of Data Intensive Computing*, pages 129–153. Springer New York, 2011.

[85] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A local-optimisation based strategy for cost-effective datasets storage of scientific applications in the cloud. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 179–186, 2011.

[86] Dong Yuan, Yun Yang, Xiao Liu, Gaofeng Zhang, and Jinjun Chen. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience*, 24(9):956–976, 2012.

[87] Wenwu Zhu, Chong Luo, Jianfeng Wang, and Shipeng Li. Multimedia cloud computing. *Signal Processing Magazine, IEEE*, 28(3):59–69, 2011.

Complete List of Original Publications

1. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In Parallel, Distributed and Network-Based Processing (PDP), 21st Euromicro International Conference, pp. 254-261, 27th February - 1st March 2013.

2. Adnan Ashraf, Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Stream-based admission control and scheduling for video transcoding in cloud computing. In Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium, pp.482-489, May 13-16, 2013.

3. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius. A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud. In 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2013) Santander, Spain, pp. 365-372, September 4-6, 2013.

4. Fareed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Cost-efficient dynamically scalable video transcoding in cloud computing. Technical report 1098. Turku Centre for Computer Science (TUCS), 2013

5. Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Bit rate reduction video transcoding with distributed computing. In Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference, pp. 206-212, February 15-17, 2012.

6. Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Analysis of video segmentation for spatial resolution reduction video transcoding. In Intelligent Signal Processing and Communications

Systems (ISPACS), 2011 International Symposium, pp.1-6, December 07-09, 2011.

7. Fareed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Analysis of Video Transcoding on Multi-Core Platform. In Fourth Swedish Workshop on Multi-Core Computing (ACM), November 23-25, 2011

8. Fareed Jokhio, Andreas Dahlin, Johan Ersfolk, and Johan Lilius. Analysis of an RVC-CAL MPEG-4 Simple Profile Decoder. Technical Report 1018, TUCS, pp. 1-22, 2011.

9. Andreas Dahlin, Fareed Jokhio, Johan Lilius, Jérôme Gorin, and Mickaël Raulet. Interfacing and scheduling legacy code within the Canals framework. In Design and Architectures for Signal and Image Processing (DASIP), pp. 1-8, November 02-04, 2011.

10. Johan Ersfolk, Ghislain Roquier, Fareed Jokhio, Johan Lilius, and Marco Mattavelli. Scheduling of Dynamic Dataflow Programs with Model Checking. In 2011 IEEE Workshop on Signal Processing Systems (SiPS), pp. 1-6, October 4-7, 2011.

Part II

Original Publications

Paper I

Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing (PDP 2013)

Fareed Jokhio and Adnan Ashraf and Sébastien Lafond and Ivan Porres and Johan Lilius

Originally published in proceedings of *the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. IEEE Computer Society, pp.254-261, 27th-February - 1st March 2013, Belfast, Northern Ireland.

Paper II

Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing

Adnan Ashraf and Fareed Jokhio and Tewodros Deneke and Sébastien Lafond and Ivan Porres and Johan Lilius

Originally published in proceedings of *the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2013)*. IEEE Computer Society, pp.482-489, MAY 13-16, 2013, Delft, The Netherlands

Paper III

A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud

Fareed Jokhio and Adnan Ashraf and Sébastien Lafond and Johan Lilius

Published in proceedings of *the 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2013)*. IEEE Computer Society, pp. 365-372, September 4-6, 2013, Santander, Spain.

Paper IV

Cost-efficient dynamically scalable video transcoding in cloud computing

Fareed Jokhio and Adnan Ashraf and Sébastien Lafond and Ivan Porres and Johan Lilius

Originally published as a Technical Report 1098 in *Turku Centre for Computer Science(TUCS) 2013*

This work is based on the following publications.

[1] Fareed Jokhio and Adnan Ashraf and Sébastien Lafond and Ivan Porres and Johan Lilius, "Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing (PDP 2013)", Published in proceedings of *the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*. IEEE Computer Society, pp.254-261, 27th-February - 1st March 2013, Belfast, Northern Ireland.

[2] Fareed Jokhio and Adnan Ashraf and Sébastien Lafond and Johan Lilius, "A Computation and Storage Trade-off Strategy for Cost-Efficient Video Transcoding in the Cloud", Published in proceedings of *the 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2013)*. IEEE Computer Society, pp. 365-372, September 4-6, 2013, Santander, Spain.

Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing

Fareed Jokhio
Adnan Ashraf
Sébastien Lafond
Ivan Porres
Johan Lilius

Department of Information Technologies
Åbo Akademi University
Joukahaisenkatu 3-5A, 20520, Turku, Finland
{fjokhio, aashraf, slafond, iporres, jolilius}@abo.fi

TUCS Technical Report
No 1098, December 2013

Abstract

Video transcoding of a large number of on-demand videos requires a large scale cluster of transcoding servers. Moreover, storage of multiple transcoded versions of each source video requires a large amount of disk space. Infrastructure as a Service (IaaS) clouds provide virtual machines (VMs) for creating a dynamically scalable cluster of servers. Likewise, a cloud storage service may be used to store a large number of transcoded videos. Moreover, it may be possible to reduce the total IaaS cost by trading storage for computation, or vice versa. In this paper, we present prediction-based dynamic resource allocation algorithms to scale on-demand video transcoding service on a given IaaS cloud. The proposed algorithms provide mechanisms for allocation and deallocation of VMs to a dynamically scalable cluster of video transcoding servers in a horizontal fashion. We also present a computation and storage trade-off strategy for cost-efficient video transcoding in the cloud called cost and popularity score based strategy. The proposed strategy estimates computation cost, storage cost, and video popularity of individual transcoded videos and then uses this information to make decisions on how long a video should be stored or how frequently it should be re-transcoded from a given source video. The proposed algorithms and the trade-off strategy are demonstrated in a discrete-event simulation and are empirically evaluated using a realistic load pattern.

Keywords: Video transcoding, dynamic resource allocation, computation and storage trade-off, cost-efficiency, cloud computing

TUCS Laboratory
Embedded Systems Laboratory
Software Engineering Laboratory

1 Introduction

With an ever increasing number of digital videos delivered everyday via the Internet, the number of video formats and video codecs used for digital video representation are also increasing rapidly. Moreover, since video streaming of a large number of videos requires a lot of server-side resources, digital videos are often stored and transmitted in compressed formats to conserve storage space and communication bandwidth. With the emergence of a large number of video compression techniques and packaging formats, such as MPEG-4 [32] and H.264 [33], the diversity of digital video content representation has grown even faster. However, for a client-side device, it is practically impossible to support all the existing video formats. Therefore, an unsupported format needs to be converted into one of the supported formats before the video could be played on the device.

The process of converting a compressed digital video from one format to another format is termed as video transcoding [31]. It may involve extracting video and audio tracks from the file container, decoding the tracks, down-scaling frame-size, dropping of frames, reducing bit-rate by applying coarser quantization, encoding the audio and video tracks into a suitable format, and packing those tracks into a new container. Since video transcoding is a compute-intensive operation, transcoding of a large number of on-demand videos requires a large scale cluster of transcoding servers. Similarly, storage of multiple transcoded versions of each source video requires a large amount of disk space. Moreover, in order to be able to handle different load conditions in a cost-efficient manner, the cluster of transcoding servers should be dynamically scalable.

Cloud computing provides theoretically infinite computing and storage resources, which can be provisioned in an on-demand fashion under the pay-per-use business model [4]. Infrastructure as a Service (IaaS) clouds, such as Amazon Elastic Compute Cloud (EC2)[1], provide Virtual Machines (VMs) for creating a dynamically scalable cluster of servers. Likewise, a cloud storage service may be used to store a large number of transcoded videos. Determining the number of VMs and the amount of storage to provision from an IaaS cloud is an important problem. The exact number of VMs and the exact amount of storage needed at a specific time depend on the incoming load from service users and their performance requirements.

In a cloud environment, a video transcoding operation can be performed in several different ways. For example, it is possible to map an entire video stream on a dedicated VM. However, it requires a large number of VMs to transcode several simultaneous streams. Moreover, transcoding of high-definition (HD) video streams may require a lot of time, which may violate the client-side performance requirements of the desired play rate [9]. Another approach is to split the video streams into smaller segments and then transcode them independently of one another [19]. In this approach, one VM can be used to transcode a large number of

[1] http://aws.amazon.com/ec2/

video segments belonging to different video streams. Moreover, video segments of a particular stream can be transcoded on multiple VMs.

In this paper, we present prediction-based dynamic resource allocation and deallocation algorithms [22] to scale video transcoding service on a given IaaS cloud in a horizontal fashion. The proposed algorithms allocate and deallocate VMs to a dynamically scalable cluster of video transcoding servers. We use a two-step load prediction method [2], which predicts the video transcoding rate a few steps ahead in the future to allow proactive resource allocation under soft realtime constraints. For cost-efficiency, we share VM resources among multiple video streams. The sharing of the VM resources is based on video segmentation, which splits the streams into smaller segments that can be transcoded independently of one another [22]. We also investigate the computation and storage cost trade-off for video transcoding in the cloud and present a cost-efficient strategy called cost and popularity score based strategy [21]. The proposed strategy estimates computation cost, storage cost, and video popularity of individual transcoded videos and then uses this information to make decisions on how long a video should be stored or how frequently it should be re-transcoded from its source video. The objective is to reduce the total IaaS cost by trading storage for computation, or vice versa. Thus, the paper makes two contributions: (1) proactive resource allocation and deallocation algorithms to scale video transcoding service on a given IaaS cloud; and (2) a computation and storage cost trade-off strategy for video transcoding in cloud computing. It extends the works published in [20], [21], and [22] and provides an extended evaluation. The proposed algorithms and the trade-off strategy are demonstrated in discrete-event simulations and are empirically evaluated using a realistic load pattern.

We proceed as follows. Section 2 presents the system architecture of an on-demand video transcoding service and sets the context for the proposed dynamic resource allocation algorithms and the proposed trade-off strategy. Section 3 describes the proposed algorithms. The proposed trade-off strategy is presented in Section 4. Section 5 describes experimental design and presents the results of the experimental evaluation. In Section 6, we discuss important related works before concluding in Section 7.

2 System Architecture

The system architecture of the cloud-based on-demand video transcoding service is shown in Figure 1. It consists of a *streaming server*, a *video splitter*, a *video merger*, a *video repository*, a dynamically scalable cluster of *transcoding servers*, a *load balancer*, a *master controller*, and a *load predictor*. The video requests and responses are routed through the streaming server. It uses an output video buffer, which temporarily stores the transcoded videos at the server-side. Our resource allocation algorithms are designed to avoid over and underflow of the video buffer.

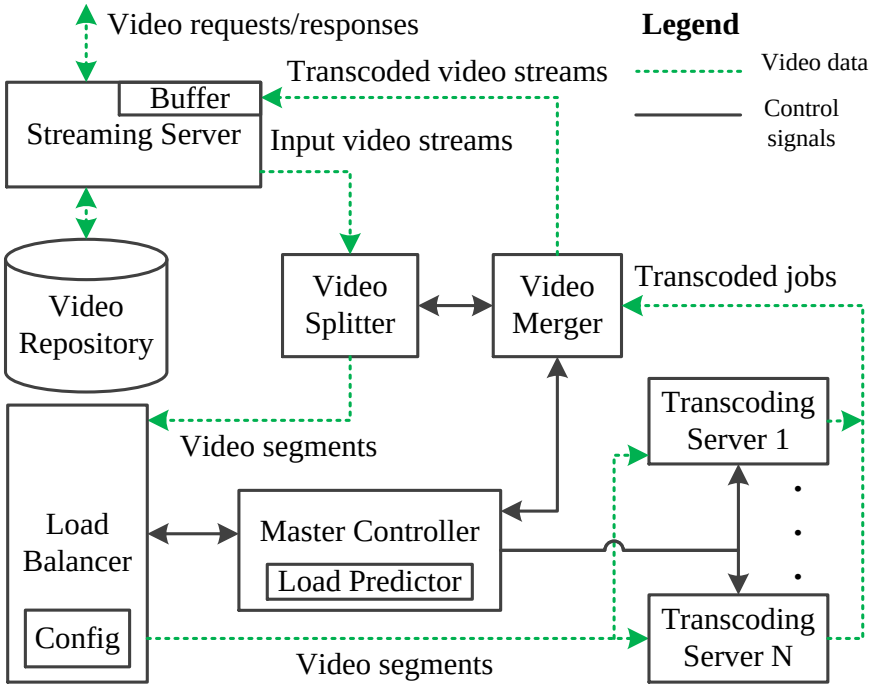

Figure 1: System architecture of the cloud-based on-demand video transcoding service

The overflow occurs if the video transcoding rate exceeds the video play rate and the capacity of the buffer. Likewise, the buffer underflow may occur when the play rate exceeds the transcoding rate, while the buffer does not contain enough frames either to avoid the underflow situation. Since the main focus of this paper is on video transcoding, we assume that the streaming server is not a bottleneck.

The video streams in certain compressed formats are stored in the video repository. The streaming server accepts video requests from users and checks if the required video is available in the video repository. If it finds the video in the desired format and resolution, it starts streaming the video. However, if it finds that the requested video is stored only in another format or resolution than the one desired by the user, it sends the video for segmentation and subsequent transcoding. Then, as soon as it receives the transcoded video from the video merger, it starts streaming the video.

After each transcoding operation, the computation and storage trade-off strategy determines if the transcoded video should be stored in the video repository or not. Moreover, if a transcoded video is stored, then the trade-off strategy also determines the duration for which the video should be stored. Therefore, it allows us to trade computation for storage or vice versa in order to reduce the total operational cost and to improve performance of the transcoding service.

The video splitter splits the video streams into smaller segments called jobs, which are placed into the job queue. A compressed video consists of three

different types of frames namely, *I*-frames (intracoded frames), *P*-frames (predicted frames), and *B*-frames (bi-directional predicted frames). Due to interdependencies among different types of frames, the video splitting or segmentation is performed at the key frames, which are always I-frames. An I-frame followed by P and B frames is termed as a group of pictures (GOP). GOPs represent atomic units that can be transcoded independently of one another [22]. Video segmentation at GOP level is discussed in more detail in [19] and [23].

The load balancer employs a task assignment policy, which distributes load on the transcoding servers. In other words, it decides when and to which transcoding server a transcoding job should be sent. It maintains a configuration file, which contains information about transcoding servers that perform the transcoding operations. As a result of the dynamic resource allocation and deallocation operations, the configuration file is often updated with new information. The load balancer serves the jobs in FIFO (First In, First Out) order. It implements one or more job scheduling policies, such as, the *shortest queue length* policy, which selects a transcoding server with the shortest queue length and the *shortest queue waiting time* policy, which selects a transcoding server with the least queue waiting time.

The actual transcoding is performed by the transcoding servers. They get compressed video segments, perform the required transcoding operations, and return the transcoded video segments for merging. A transcoding server runs on a dynamically provisioned VM. Each transcoding server processes one or more simultaneous jobs. When a transcoding job arrives at a transcoding server, it is placed in the server's queue from where it is subsequently processed.

The master controller acts as the main controller and the resource allocator. It implements prediction-based dynamic resource allocation and deallocation algorithms, as described in Section 3. It also implements one or more computation and storage trade-off strategies, such as the proposed cost and popularity score based strategy, which is presented in Section 4. In our approach, the resource allocation and deallocation is mainly based on the target play rate of the video streams and the predicted transcoding rate of the transcoding servers. For load prediction, the master controller uses load predictor, which predicts future load on the transcoding servers. The video merger merges the transcoded jobs into video streams, which form video responses. Our load prediction approach is described in detail in [7] and [22]. It consists of a load tracker and a load predictor [2]. We use exponential moving average (EMA) for the load tracker and a simple linear regression model [26] for the load predictor.

3 Proactive VM Allocation Algorithms

In this section, the proposed dynamic VM allocation and deallocation algorithms for video transcoding in the cloud are presented. The objective is to reduce the over and under allocation of resources while satisfying the client-side performance

requirements. For the sake of clarity, the concepts used in the algorithms and their notation are summarized in Table 1. The algorithms implement proactive control, which uses a two-step load prediction approach [2] in which the current and the past system load is tracked to predict the future system load. The predicted system load is then used to make decisions on the allocation and deallocation of VMs to a dynamically scalable cluster of transcoding servers. Moreover, a fixed minimum number of transcoding servers is always maintained, which represents the base capacity N_B.

On discrete-time intervals, the master controller obtains the play rate of all video streams and adds them together to get the total target play rate $PR(t)$. It then obtains the video transcoding rate from each transcoding server and calculates the total transcoding rate $TR(t)$. Moreover, for proactive VM allocation, it uses load predictor to predict the total transcoding rate $\hat{TR}(t)$ a few steps ahead in the future.

The algorithms are designed to be cost-efficient while minimizing potential oscillations in the number of VMs [34]. This is desirable because, in practice, provisioning of a VM takes a few minutes [5], [6]. Therefore, oscillations in the number of VMs may lead to deteriorated performance. Moreover, since some contemporary IaaS providers, such as Amazon EC2, charge on hourly basis, oscillations will result in a higher provisioning cost. Therefore, the algorithms counteract oscillations by delaying new VM allocation operations until previous VM allocation operations have been realized [18]. Furthermore, for cost-efficiency, the deallocation algorithm terminates only those VMs whose renting period approaches its completion.

3.1 VM Allocation Algorithm

The VM allocation algorithm is given as Algorithm 1. The first two steps deal with the calculation of the target play rate $PR(t)$ of all streams and the total transcoding rate $TR(t)$ of all transcoding servers (lines 3–7). The algorithm then obtains the predicted total transcoding rate $\hat{TR}(t)$ from the load predictor (line 8). Moreover, to avoid underflow of the output video buffer that temporarily stores transcoded jobs at the server-side, it considers the size of the output video buffer $B_S(t)$. If the target play rate exceeds the predicted transcoding rate while the buffer size $B_S(t)$ falls below its lower threshold B_L (line 9), the algorithm chooses to allocate resources by provisioning one or more VMs (line 10). The number of VMs to provision $N_P(t)$ is calculated as follows

$$N_P(t) = \left\lceil \frac{PR(t) - \hat{TR}(t)}{\frac{TR(t)}{|S(t)|}} \right\rceil \tag{1}$$

where $|S(t)|$ is the number of transcoding servers at time t. The VM allocation algorithm also takes into account the number of jobs waiting in the servers' queues. It checks the average queue length of all servers $avgQJobs(t)$ and if the average queue length is above a predefined maximum upper threshold $MAXQL_{UT}$

Table 1: Summary of concepts and their notation for VM allocation algorithms

Notation	Description
$avgQJobs(t)$	average queue length of all servers at discrete-time t
$count_{over}(t)$	over allocation count at t
$N_P(t)$	number of servers to provision at t based on $PR(t)$ and $\hat{TR}(t)$
$N_{P_Q}(t)$	number of servers to provision at t based on $avgQJobs(t)$
$N_T(t)$	number of servers to terminate at t
$PR(t)$	sum of target play rates of all streams at t
$S(t)$	set of transcoding servers at t
$S_p(t)$	set of newly provisioned servers at t
$S_c(t)$	servers close to completion of renting period at t
$S_t(t)$	servers selected for termination at t
$TR(t)$	total transcoding rate of all servers at t
$\hat{TR}(t)$	predicted total transcoding rate of all servers at t
$RT(s,t)$	remaining time of server s at t with respect to renting hour
$V(t)$	set of video streams at t
B_L	buffer size lower threshold in megabytes
$B_S(t)$	size of the output video buffer in megabytes
B_U	buffer size upper threshold in megabytes
C_T	over allocation count threshold
$jobCompletion$	job completion delay
$MAXQL_{UT}$	maximum queue length upper threshold
N_B	number of servers to use as base capacity
RT_L	remaining time lower threshold
RT_U	remaining time upper threshold
$startUp$	server startup delay
$calcN_P()$	calculate the value of $N_P(t)$
$calcN_T()$	calculate the value of $N_T(t)$
$calcQN_P()$	calculate the value of $N_{P_Q}(t)$ based on queue length
$calRT(s,t)$	calculate the value of $RT(s,t)$
$delay(d)$	delay for duration d
$getPR()$	get $PR(t)$ from video merger
$getTR(s)$	get transcoding rate of server s
$get\hat{TR}()$	get $\hat{TR}(t)$ from load predictor
$provision(n)$	provision n servers
$select(n)$	select n servers for termination
$sort(S)$	sort servers S on remaining time
$terminate(S)$	terminate servers S

(line 12), it chooses to provision one or more servers (line 13). In this case, the number of VMs to provision $N_{P_Q}(t)$ is calculated as follows

$$N_{P_Q}(t) = \left\lceil \frac{avgQJobs(t)}{MAXQL_{UT}} \right\rceil \tag{2}$$

The algorithm then provisions $N_P(t) + N_{P_Q}(t)$ VMs, which are added to the cluster of transcoding servers (lines 20–21). To minimize potential oscillations due to unnecessary VM allocations, the algorithm adds a delay for the VM startup time (line 22). Furthermore, it ensures that the total number of VMs $|S(t)|$ does not exceed the total number of video streams $|V(t)|$. The algorithm adjusts the number of VMs to provision $N_P(t)$ if $|S(t)| + N_P(t)$ exceeds $|V(t)|$ (lines 16–18). This is desirable because the transcoding rate of a video on a single VM is usually higher than the required play rate.

Algorithm 1 VM allocation algorithm

```
1:  while true do
2:      N_P(t) := 0, N_PQ(t) := 0
3:      PR(t) := getPR()
4:      TR(t) := 0
5:      for s∈S(t) do
6:          TR(t) := TR(t) + getTR(s)
7:      end for
8:      T̂R(t) := getT̂R(TR(t))
9:      if T̂R(t) < PR(t) ∧ B_S(t) < B_L then
10:         N_P(t) := calcN_P()
11:     end if
12:     if avgQJobs(t) > MAXQL_UT then
13:         N_PQ(t) := calcQN_P()
14:     end if
15:     N_P(t) := N_P(t) + N_PQ(t)
16:     if |S(t)| + N_P(t) > |V(t)| then
17:         N_P(t) := |V(t)| − |S(t)|
18:     end if
19:     if N_P(t) ≥ 1 then
20:         S_p(t) := provision(N_P(t))
21:         S(t) := S(t) ∪ S_p(t)
22:         delay(startUp)
23:     end if
24: end while
```

3.2 VM Deallocation Algorithm

The VM deallocation algorithm is presented in Algorithm 2. The main objective of the algorithm is to minimize the VM provisioning cost, which is a function of the number of VMs and time. Thus, it terminates any redundant VMs as soon as possible. Moreover, to avoid overflow of the output video buffer, it considers the size of the output video buffer $B_S(t)$. After obtaining the target play rate $PR(t)$ and the predicted total transcoding rate $\hat{TR}(t)$ (lines 2–7), the algorithm makes a comparison. If $\hat{TR}(t)$ exceeds $PR(t)$ while the buffer size $B_S(t)$ exceeds its upper threshold B_U (line 8), it may choose to deallocate resources by terminating one or more VMs. However, to minimize unnecessary oscillations, it deallocates resources only when the buffer overflow situation persists for a predetermined minimum amount of time.

Algorithm 2 VM deallocation algorithm

```
 1: while true do
 2:   PR(t) := getPR()
 3:   TR(t) := 0
 4:   for sϵS(t) do
 5:     TR(t) := TR(t) + getTR(s)
 6:   end for
 7:   T̂R(t) := getT̂R(TR(t))
 8:   if T̂R(t) > PR(t) ∧ B_S(t) > B_U ∧ count_over(t) > C_T then
 9:     for sϵS(t) do
10:       RT(s,t) := calRT(s,t)
11:     end for
12:     S_c(t) := {∀sϵS(t)|RT(s,t) < RT_U ∧ RT(s,t) > RT_L}
13:     if |S_c(t)| ≥ 1 then
14:       N_T(t) := calcN_T()
15:       N_T(t) := min(N_T(t), |S_c(t)|)
16:       if N_T(t) ≥ 1 then
17:         sort(S_c(t))
18:         S_t(t) := select(N_T(t))
19:         S(t) := S(t) \ S_t(t)
20:         delay(jobCompletion)
21:         terminate(S_t(t))
22:       end if
23:     end if
24:   end if
25: end while
```

In the next step, the algorithm calculates the remaining time of each transcoding server $RT(s,t)$ with respect to the completion of the renting period (lines 9–11). It then checks if there are any transcoding servers whose remaining time is

less than the predetermined upper threshold of remaining time RT_U and more than the lower threshold of remaining time RT_L (line 12). The objective is to terminate only those servers whose renting period is close to the completion, while excluding any servers that are extremely close to the completion of their renting period. Therefore, it is not practically feasible to complete all running and pending jobs on them before the start of the next renting period. If the algorithm finds at least one such server $S_c(t)$ (line 13), it calculates the number of servers to terminate $N_T(t)$ as

$$N_T(t) = \left\lceil \frac{\hat{TR}(t) - PR(t)}{\frac{TR(t)}{|S(t)|}} \right\rceil - N_B \tag{3}$$

Then, it sorts the transcoding servers in $S_c(t)$ on the basis of their remaining time (line 17), and selects the servers with the lowest remaining time for termination (line 18). The rationale of sorting of servers is to ensure cost-efficiency by selecting the servers closer to completion of their renting period. A VM that has been selected for termination might have some pending jobs in its queue. Therefore, it is necessary to ensure that the termination of a VM does not abandon any jobs in its queue. One way to do this is to migrate all pending jobs to other VMs and then terminate the VM [5], [6]. However, since transcoding of video segments takes relatively less time to complete, it is more reasonable to let the jobs complete their execution without requiring them to migrate and then terminate a VM when there are no more running and pending jobs on it. Therefore, the deallocation algorithm terminates a VM only when the VM renting period approaches its completion and all jobs on the server complete their execution (line 20). Finally, the selected servers are terminated and removed from the cluster (line 21).

4 Computation and Storage Trade-off Strategy

In this section, we present the proposed computation and storage trade-off strategy. For the sake of clarity, we provide a summary of the notations in Table 2. The proposed cost and popularity score based strategy estimates the computation cost, the storage cost, and the video popularity of individual transcoded videos and then uses this information to make decisions on how long a video should be stored or how frequently it should be re-transcoded from a given source video. In an on-demand video streaming service, the source videos are usually high quality videos that comprise the primary datasets. Therefore, irrespective of their computation and storage costs, they are never deleted from the video repository. The transcoded videos, on the other hand, are the derived datasets that can be regenerated on-demand from their source videos. Therefore, they should only be stored in the video repository when it is cost-efficient to store them. Thus, the proposed strategy is only applicable to the transcoded videos. In other words, since the computation and the storage costs of the source videos are not relevant, the proposed

Table 2: Summary of concepts and their notation for trade-off strategy

Notation	Description
τ	set of transcoded videos
τ_i	i^{th} transcoded video
NS_{τ_i}	new cost and popularity score of τ_i
RC_T	renting cost of a transcoding server per renting hour
S_{τ_i}	total cumulative cost and popularity score of τ_i
SC_{τ_i}	storage cost of τ_i per time unit
SC_m	monthly storage cost per 1 gigabytes
SD_{τ_i}	storage duration for transcoded video τ_i
TC_{τ_i}	transcoding cost of τ_i
TT_{τ_i}	transcoding time of τ_i
$VSmb_{\tau_i}$	transcoded video τ_i size in megabytes
DC	decrement in S_{τ_i}
GB_{mb}	megabytes to gigabytes conversion factor
H_{sec}	hour to seconds conversion factor
RP_S	month to desired time unit conversion factor
$calcNS(\tau_i)$	calculate NS_{τ_i}
$calcSC(\tau_i)$	calculate SC_{τ_i}
$calcTC(\tau_i)$	calculate TC_{τ_i}
$delay(SD_{\tau_i})$	delay for SD_{τ_i}
$getS(\tau_i)$	get S_{τ_i}
$getSC(\tau_i)$	get SC_{τ_i}
$getTC(\tau_i)$	get TC_{τ_i}
$removeVideo(\tau_i)$	remove video τ_i

strategy is based only on the computation and storage costs of the transcoded videos.

In cloud computing, the computation cost is essentially the cost of using VMs, which is usually calculated on an hourly basis. The storage cost, on the other hand, is often computed on a monthly basis. The computation cost of a transcoded video depends on its transcoding time and on how often the video is re-transcoded. Thus, if a video is frequently re-transcoded, the computation cost would increase rapidly. On the other hand, the storage cost of a transcoded video depends on the length of the storage duration and the video size on disk. Therefore, it increases gradually with the passage of time. The longer the duration, the higher the cost. Thus, our proposed strategy estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal. This estimated equilibrium point indicates the minimum duration for which the video should be stored in the video repository. Figure 2 shows that if a video

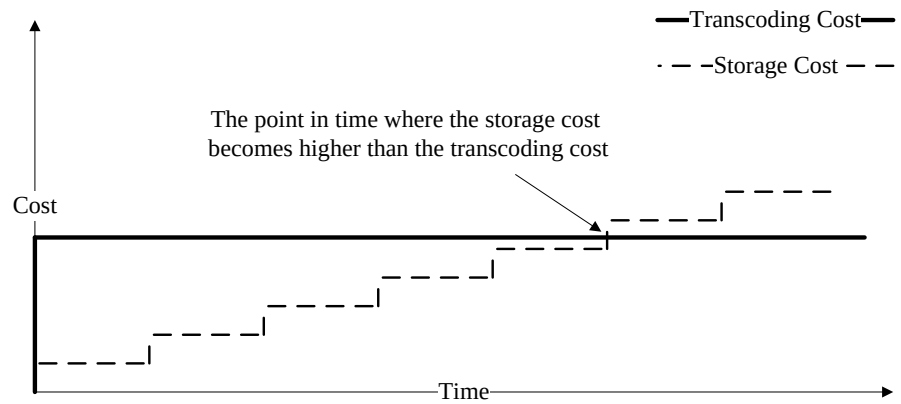

Figure 2: The estimated equilibrium point between the storage cost and the transcoding cost of a transcoded video

is transcoded once and stored in the video repository, then initially the computation cost is higher than the storage cost. However, with the passage of time, the storage cost continues to increase until it becomes equal to the computation cost and then it grows even further unless the video is removed from the video repository. Thus, if the video is deleted before its estimated equilibrium point and then it is subsequently requested, the computation cost will increase due to unnecessary re-transcoding. Likewise, if the video is stored beyond its estimated equilibrium point and then it does not receive a subsequent request, the storage cost will increase unnecessarily.

In an on-demand video streaming service, each transcoded video may be requested and viewed a number of times. Frequently viewed, popular videos get a lot of requests. While, sporadically viewed, less popular videos get only a few requests. For cost-efficient storage, it is essential to use an estimate of the popularity of the individual transcoded videos. This information can then be used to determine the exact duration for which a video should be stored in the video repository. Therefore, the proposed strategy accounts for the popularity of individual transcoded videos. It uses the estimated computation cost, the estimated storage cost, and the video popularity information to calculate a cost and popularity score S_{τ_i} for each transcoded video τ_i. The higher the score the longer the video is stored in the video repository. Thus, with the incorporation of the video cost and popularity score, it becomes justifiable to store popular transcoded videos beyond their estimated equilibrium point. In other words, it differentiates popular videos that should be stored for a longer duration.

In our proposed strategy, the storage cost SC_{τ_i} of a transcoded video τ_i is calculated as

$$SC_{\tau_i} = \frac{VSmb_{\tau_i}}{GB_{mb}} \cdot \frac{SC_m}{RP_S} \cdot SD_{\tau_i} \tag{4}$$

where $VSmb_{\tau_i}$ is the size of the transcoded video τ_i in megabytes, GB_{mb} is the

megabytes to gigabytes conversion factor, SC_m is the monthly storage cost per 1 gigabytes of storage, RP_S is the month to desired time unit conversion factor, and SD_{τ_i} is the length of the storage duration for the transcoded video τ_i. Similarly, the transcoding cost TC_{τ_i} of a transcoded video τ_i is calculated as

$$TC_{\tau_i} = TT_{\tau_i} \cdot \frac{RC_T}{H_{sec}} \tag{5}$$

where TT_{τ_i} is the transcoding time of τ_i, RC_T is the renting cost of a transcoding server per renting hour, and H_{sec} is the hour to seconds conversion factor, which is used to normalize the computation cost to a per second basis.

Whenever a new request for a transcoded video τ_i arrives at the streaming server, the video cost and popularity score S_{τ_i} is updated to reflect the new costs and the new popularity information. The new cost and popularity score NS_{τ_i} represents the estimated equilibrium point where the computation cost and the storage cost of τ_i become equal. Therefore, it indicates the minimum duration for which the video should be stored. The new cost and popularity score NS_{τ_i} of a video τ_i is calculated as the ratio of the transcoding cost TC_{τ_i} and the storage cost SC_{τ_i}

$$NS_{\tau_i} = \frac{TC_{\tau_i}}{SC_{\tau_i}} \tag{6}$$

Finally, the total cost and popularity score S_{τ_i} of a video τ_i is calculated by accumulating the new cost and popularity score NS_{τ_i} of the said video over time. That is, for each new request of a transcoded video τ_i, we obtain the previous value of the total cost and popularity score S_{τ_i} of the transcoded video, calculate NS_{τ_i}, and then add them together to produce the new value of the S_{τ_i}. Moreover, the total cost and popularity score of a video that was not stored previously is set to NS_{τ_i}. The total cost and popularity score S_{τ_i} determines the exact duration for which a video τ_i should be stored. The pseudocode for score calculation is presented in Algorithm 3.

Algorithm 3 Calculation of cost and popularity score

while true do
  if τ_i is requested **then**
    $SC_{\tau_i} := calcSC(\tau_i)$
    $TC_{\tau_i} := calcTC(\tau_i)$
    $NS_{\tau_i} := calcNS(\tau_i)$
    $S_{\tau_i} := \begin{cases} S_{\tau_i} + NS_{\tau_i}, & \text{if } \tau_i \text{ was stored previously} \\ NS_{\tau_i}, & \text{otherwise} \end{cases}$
  end if
end while

Each transcoded video τ_i should be stored in the video repository for as long as it is cost-efficient to store it. However, when a video loses its popularity, it

should be subsequently deleted to avoid unnecessary storage cost. Therefore, on certain time intervals, the proposed strategy performs the following steps for each transcoded video τ_i. It obtains the storage cost SC_{τ_i}, the cost and popularity score S_{τ_i}, and the transcoding cost TC_{τ_i}. Then, it multiplies S_{τ_i} and TC_{τ_i} and compares it with SC_{τ_i} as follows

$$SC_{\tau_i} > TC_{\tau_i} \cdot S_{\tau_i} \tag{7}$$

If the inequality holds, it implies that it is cost-efficient to delete the transcoded video. Therefore, the video is removed from the video repository. However, if the inequality does not hold, it indicates that it is not cost-efficient to delete the video. Therefore, the video is not removed. Moreover, the cost and popularity score S_{τ_i} is decremented in accordance with the length of the time interval to reflect the passage of time. In this way, when a popular video loses its popularity, it starts losing its cost and popularity score as well until it is removed from the video repository or it gets some new requests to regain its popularity. The pseudocode to decrement cost and popularity score S_{τ_i} and to remove a video is given as Algorithm 4.

Algorithm 4 Decrementing score and removing a video

```
1:  while true do
2:    for τ_i ϵ τ do
3:      SC_τi := getSC(τ_i)
4:      TC_τi := getTC(τ_i)
5:      S_τi := getS(τ_i)
6:      if SC_τi > TC_τi · S_τi then
7:        removeVideo(τ_i)
8:      else
9:        S_τi := S_τi − DC
10:     end if
11:   end for
12:   delay(SD_τi)
13: end while
```

5 Experimental Evaluation

Software simulations are often used to test and evaluate new approaches and strategies involving complex environments [10], [8]. For our proposed resource allocation algorithms and trade-off strategy, we have developed a discrete-event simulation in the Python programming language. It is based on the SimPy simulation framework [25]. Also, for a comparison of the results with the alternative existing approaches, we have developed discrete-event simulations for two intuitive computation and storage trade-off strategies, which are the store all strategy

and the usage based strategy [36]. The store all strategy stores all transcoded videos irrespective of their costs and popularity. While the usage based strategy stores only popular videos and removes the rest. That is, it does not account for the computation and storage costs.

5.1 Experimental Design and Setup

For the computation and storage costs, we used the Amazon EC2 and the Amazon S3[2] cost models. The computation cost in Amazon EC2 is based on an hourly charge model. Whereas, the storage cost of Amazon S3 is based on a monthly charge model. In our experiment, we used only small instances. As of writing of this paper, the cost of a small instance in Amazon EC2 is $\$0.06$ per hour. Whereas, the cost of storage space in Amazon S3 is based on a nonlinear cost model as shown in Table 3.

The experiment used HD, SD (Standard-Definition), and mobile video streams. Since SD videos currently have a higher demand than the HD and mobile videos, we considered 20% HD, 30% mobile, and 50% SD video streams. The GOP size for different types of videos was different. For HD videos, the average size of a video segment was 75 frames with a standard deviation of 7 frames. Likewise, for SD and mobile videos, the average size of a segment was 250 frames with a standard deviation of 20 frames.

In an on-demand video transcoding service, a source video is usually transcoded in many different formats. Therefore, we assumed that a source video can be transcoded into a maximum of 30 different formats. Likewise, since in an on-demand video streaming service, the number of source videos always continue to grow, we used a continuously increasing number of source videos in our experiment. However, since the number of the newly uploaded source videos is usually only a small fraction of the total number of downloaded videos, the video upload rate in our experiment was assumed to be 1% of the total number of the video download requests. The desired time unit for storage, as used in the month to desired time unit conversion factor RP_S, was assumed to be one day. Therefore, RP_S was 30. Moreover, the minimum storage duration for a transcoded video SD_{τ_i} was also assumed to be one day.

The objective of the experiment was to evaluate the proposed algorithms and trade-off strategy for a realistic load pattern. Therefore, it used a real load pattern, which constitutes real video access data from Bambuser AB[3]. The load pattern consists of approximately 40 days of real video access data. The total number of frames in a video stream was in the range of 18000 to 90000, which represents an approximate play time of 10 to 50 minutes with the frame rate of 30 frames per second.

[2]http://aws.amazon.com/s3/

[3]http://bambuser.com/

Table 3: Amazon S3 storage pricing

	Standard Storage
First 1 TB per month	$ 0.095 per GB
Next 49 TB per month	$ 0.080 per GB
Next 450 TB per month	$ 0.070 per GB
Next 500 TB per month	$ 0.065 per GB
Next 4000 TB per month	$ 0.060 per GB
Over 5000 TB per month	$ 0.055 per GB

5.2 Results and Analysis

In this section, we compare the experimental results of the proposed strategy with that of the store all strategy and the usage based strategy. Each result in Figure 3 to Figure 5 consists of seven different plots, which are number of user requests, number of transcoding servers, transcoding cost, storage cost, storage size, number of source videos, and number of transcoded videos. The number of user requests plot represents the load pattern of the video access data. In other words, it is the user load on the streaming server. Due to data confidentiality, the exact volume of the load can not be revealed. Therefore, we have omitted the scale of this plot from all the results. The number of transcoding servers plot shows the total number of transcoding servers being used at a particular time. The transcoding cost plot represents the total computation cost of all transcoded videos in US dollars. Similarly, the storage cost plot shows the storage cost in US dollars of all transcoded videos, which are stored in the video repository. The storage size plot represents the total size of the cloud storage used to store the transcoded videos. The number of source videos plot shows the total number of source videos in the video repository. Likewise, the number of transcoded videos is the total number of transcoded videos in the video repository. The results are also summarized in Table 4.

Figure 3 presents the simulation results of the store all strategy. The results span over a period of 40 days. At the end of the simulation, the total number of transcoded videos in the video repository was 206590, while the total number of source videos was 20902. The average number of transcoding servers was 102, the total transcoding cost was $\$4458.42$, the total storage cost was $\$4911.36$, and the total storage size was 42.16 terabytes. Since the store all strategy stores all transcoded videos irrespective of their computation and storage costs, the storage cost was very high due to a large number of transcoded videos stored in the video repository. Therefore, the results indicate that the store all strategy is not cost-efficient.

Figure 4 presents the results of the usage based strategy. At the end of the simulation, the total number of transcoded videos in the video repository was 190734 for the same number of source videos as used in the store all strategy.

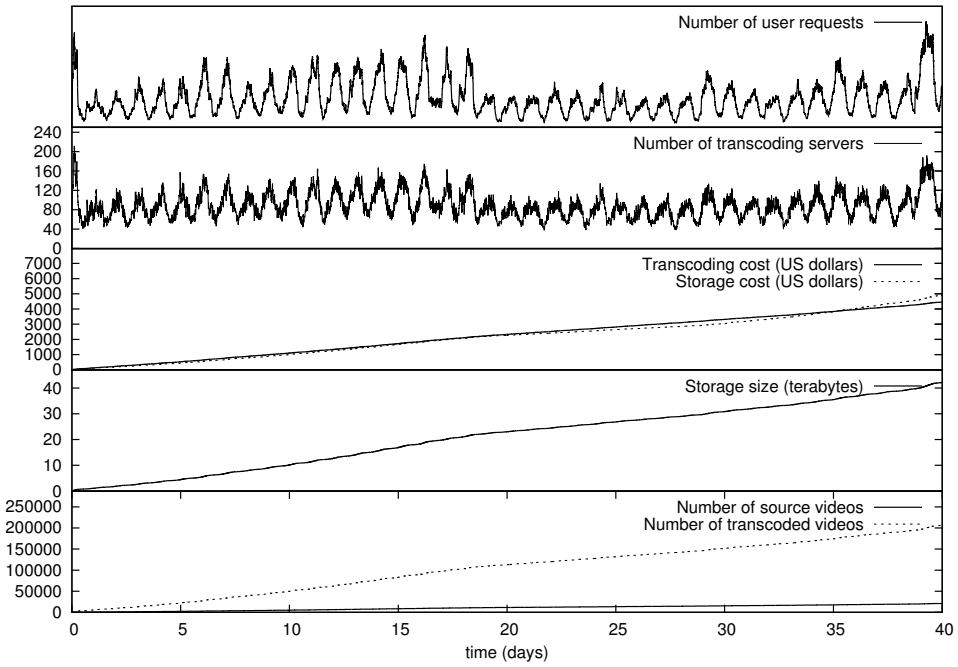

Figure 3: Store all strategy

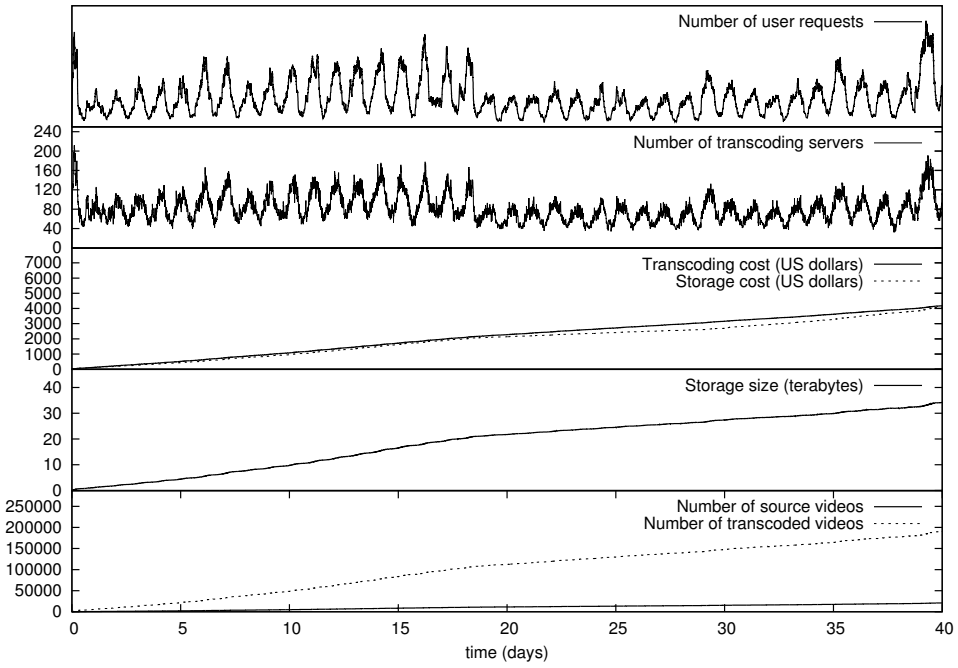

Figure 4: Usage based strategy

Table 4: Summary of results

Strategy	Avg. servers	Transcoding cost	Storage cost	Total cost
Store all	102	$4458.42	$4911.36	$9369.78
Usage based	94	$4179.12	$4090.56	$8269.68
Score based	107	$4893.60	$2307.84	$7201.44

The average number of transcoding servers was 94, the total transcoding cost was $4179.12, the total storage cost was $4090.56, and the total storage size was 34.19 terabytes. Since the usage based strategy stores only popular videos, the storage cost of the usage based strategy was slightly less than that of the store all strategy. Therefore, the results indicate that the usage based strategy is cost-efficient when compared to the store all strategy. However, since it does not account for the computation and the storage costs, it may remove some videos that have a high transcoding cost.

Figure 5 presents the results of the proposed score based strategy. At the end of the simulation, the total number of transcoded videos in the video repository was 64392 for the same number of source videos as used in the store all strategy and the usage based strategy. The average number of transcoding servers was 107, the total transcoding cost was $4893.60, the total storage cost was $2307.84, and the total storage size was 14.93 terabytes. Since the proposed strategy accounts for the computation cost, the storage cost, and the video popularity information, the storage cost was much less than that of the store all strategy and the usage based strategy.

Figure 6 presents a comparison of the total costs, which consists of the computation cost and the storage cost. The results show that the store all strategy has the highest total cost. The usage based strategy has slightly less total cost than the store all strategy. Moreover, the proposed storage has the least total cost among all the three strategies. Therefore, the results indicate that the proposed strategy is cost-efficient when compared to the store all and the usage based strategies.

6 Related Work

Distributed video transcoding with video segmentation was proposed in [19] and [23]. Jokhio et al. [19] presented bit rate reduction video transcoding using multiple processing units, while [23] analyzed different video segmentation methods to perform spatial resolution reduction video transcoding. Huang et al. [17] presented a cloud-based video proxy to deliver transcoded videos for streaming. The main contribution of their work is a multilevel transcoding parallelization framework. Li et al. [24] proposed a cloud transcoder, which uses a compute cloud as an intermediate platform to provide transcoding service. Shin and Koh [30]

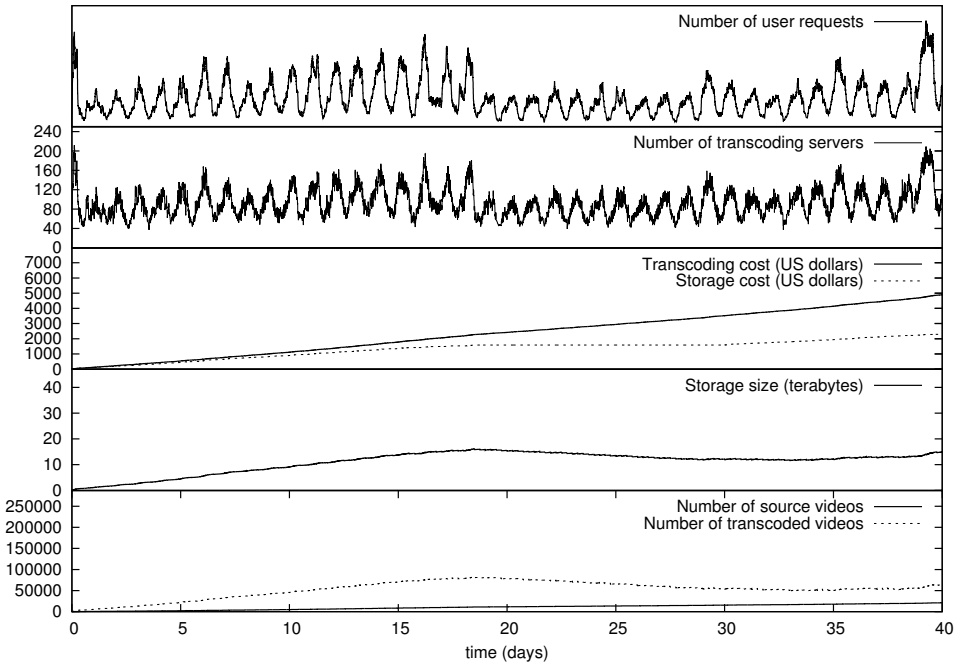

Figure 5: Proposed score based strategy

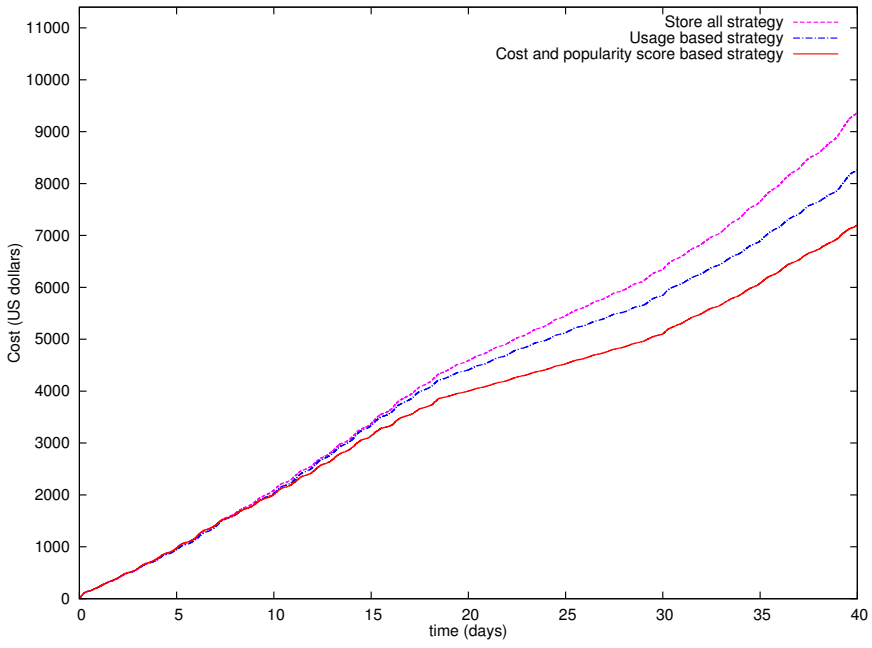

Figure 6: Cost comparison

presented a hybrid scheme to determine an optimal threshold between the static and dynamic transcoding. Ashraf et al. [8] proposed an admission control and job scheduling approach for video transcoding in the cloud. None of these papers addressed the VM allocation problem for video transcoding in cloud computing.

6.1 VM Allocation Approaches

The existing works on dynamic VM allocation can be classified into two main categories: Plan-based approaches and control theoretic approaches. The plan-based approaches can be further classified into workload prediction approaches and performance dynamics model approaches. One example of the workload prediction approaches is Ardagna et al. [3], while TwoSpot [34], Hu et al. [16], Chieu et al. [11], Iqbal et al. [18] and Han et al. [15] use a performance dynamics model. Similarly, Dutreilh et al. [13], Pan et al. [27], Patikirikorala et al. [28], and Roy et al. [29] are control theoretic approaches. One common difference between all of these works and our proposed approach is that they are not designed specifically for video transcoding in cloud computing. In contrast, our proposed approach is based on the important performance and VM allocation metrics for video transcoding service, such as video play rate and server transcoding rate. Moreover, it is cost-efficient as it uses a reduced number of VMs for a large number of video streams, it provides proactive VM allocation under soft real-time constraints, and it does not depend upon performance and dynamics of the underlying system. A more detailed analysis of the VM allocation approaches can be found in [22].

6.2 Computation and Storage Trade-off Strategies

There are currently only a few works in the area of computation and storage trade-off analysis for cost-efficient usage of cloud resources. One of the earlier attempts include Adams et al. [1], who highlighted some of the important issues and factors involved in constructing a cost-benefit model, which can be used to analyze the trade-offs between computation and storage. However, they did not propose a strategy to find the right balance between computation and storage resources. Deelman et al. [12] studied cost and performance trade-offs for an astronomy application using Amazon EC2 and Amazon S3 cost models. The authors concluded that, based on the likelihood of reuse, storing popular datasets in the cloud can be cost-effective. However, they did not provide a concrete strategy for cost-effective computation and storage of scientific datasets in the cloud.

Nectar system [14] is designed to automate the management of data and computation in a data center. It initially stores all the derived datasets when they are generated. However, when the available disk space falls below a threshold, all obsolete or least-valued datasets are garbage collected to improve resource utilization. Although Nectar provides a computation and storage trade-off strategy, it is

not designed to reduce the total cost of computation and storage in a cloud-based service that uses IaaS resources.

Yuan et al. [36] proposed two strategies for cost-effective storage of scientific datasets in the cloud, which compare the computation cost and the storage cost of the datasets. They also presented a Cost Transitive Tournament Shortest Path (CTT-SP) algorithm to find the best trade-off between the computation and the storage resources. Their strategies are called cost rate based storage strategy [35], [38] and local-optimization based storage strategy [37]. The cost rate based storage strategy compares computation cost rate and storage cost rate to decide storage status of a dataset. Whereas, the local-optimization based storage strategy partitions a data dependency graph (DDG) of datasets into linear segments and applies the CTT-SP algorithm to achieve a localized optimization. In contrast to the cost rate based storage strategy [35], [38], our proposed trade-off strategy estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal. Moreover, it estimates video popularity of the individual transcoded videos to differentiate popular videos. The DDG-based local-optimization based storage strategy of Yuan et al. [37] is not much relevant for video transcoding because video transcoding does not involve a lot of data dependencies.

Most of the existing computation and storage trade-off strategies described above were originally proposed for scientific datasets. To the best of our limited knowledge, there are currently no existing computation and storage trade-off strategies for video transcoding. The difference of application domain may play a vital role when determining cost-efficiency of the existing strategies. Therefore, some of the existing strategies may have limited efficacy and little cost-efficiency for video transcoding.

7 Conclusion

In this paper, we presented proactive VM allocation algorithms to scale video transcoding service in a cloud environment. The proposed algorithms provide a mechanism for creating a dynamically scalable cluster of video transcoding servers by provisioning VMs from an IaaS cloud. The prediction of the future user load is based on a two-step load prediction method, which allows proactive VM allocation under soft real-time constraints. For cost-efficiency, we used video segmentation which splits a video stream into smaller segments that can be transcoded independently of one another. This helped us to perform video transcoding of multiple simultaneous streams on a single server.

We also proposed a cost-efficient computation and storage trade-off strategy for video transcoding in the cloud. The proposed strategy estimates the computation cost, the storage cost, and the video popularity information of individual transcoded videos and then uses this information to make decisions on how long a

video should be stored or how frequently it should be re-transcoded from a given source video. The objective is to reduce the total IaaS cost by trading storage for computation, or vice versa.

The proposed approach is demonstrated in a discrete-event simulation and an experimental evaluation involving a realistic load pattern. Also, for the sake of comparison, we simulated two intuitive computation and storage trade-off strategies and compared their results with that of the proposed strategy. The results show that the proposed algorithms provide cost-efficient VM allocation for transcoding a large number of video streams while minimizing oscillations in the number of servers. The results also indicate that our proposed trade-off strategy is more cost-efficient than the two intuitive strategies as it provided a good trade-off between the computation and storage resources.

References

[1] Ian F. Adams, Darrell D. E. Long, Ethan L. Miller, Shankar Pasupathy, and Mark W. Storer. Maximizing efficiency by trading storage for computation. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[2] Mauro Andreolini, Sara Casolari, and Michele Colajanni. Models and framework for supporting runtime decisions in web-based systems. *ACM Trans. Web*, 2(3):17:1–17:43, July 2008.

[3] Danilo Ardagna, Carlo Ghezzi, Barbara Panicucci, and Marco Trubian. Service provisioning on the cloud: Distributed algorithms for joint capacity allocation and admission control. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2010.

[4] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[5] Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres. Feedback control algorithms to deploy and scale multiple web applications per virtual machine. In *Software Engineering and Advanced Applications (SEAA), 38th EUROMICRO Conference on*, pages 431–438, September 2012.

[6] Adnan Ashraf, Benjamin Byholm, and Ivan Porres. CRAMP: Cost-efficient resource allocation for multiple web applications with proactive scaling. *4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 581–586, 2012.

[7] Adnan Ashraf, Benjamin Byholm, and Ivan Porres. A session-based adaptive admission control approach for virtualized application servers. In *Utility and Cloud Computing (UCC), 5th IEEE/ACM International Conference on*, pages 65–72, 2012.

[8] Adnan Ashraf, Fareed Jokhio, Tewodros Deneke, Sebastien Lafond, Ivan Porres, and Johan Lilius. Stream-based admission control and scheduling for video transcoding in cloud computing. *in Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium on*, pages 482–489, 2013.

[9] N. Bjork and C. Christopoulos. Transcoder architectures for video coding. *Consumer Electronics, IEEE Transactions on*, 44(1):88 –98, February 1998.

[10] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[11] T.C. Chieu, A. Mohindra, A.A. Karve, and A. Segal. Dynamic scaling of web applications in a virtualized cloud computing environment. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 281 –286, October 2009.

[12] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: the Montage example. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press.

[13] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck. From data center resource allocation to control theory and back. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 410 –417, July 2010.

[14] Pradeep Kumar Gunda, Lenin Ravindranath, Chandramohan A. Thekkath, Yuan Yu, and Li Zhuang. Nectar: automatic management of data and computation in datacenters. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[15] Rui Han, Li Guo, M.M. Ghanem, and Yike Guo. Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644 –651, May 2012.

[16] Ye Hu, Johnny Wong, Gabriel Iszlai, and Marin Litoiu. Resource provisioning for cloud computing. In *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '09, pages 101–111, New York, NY, USA, 2009. ACM.

[17] Zixia Huang, Chao Mei, Li Erran Li, and Thomas Woo. CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy. In *INFOCOM, 2011 Proceedings IEEE*, pages 201–205, 2011.

[18] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871 – 879, 2011.

[19] F. Jokhio, T. Deneke, S. Lafond, and J. Lilius. Bit rate reduction video transcoding with distributed computing. In *Parallel, Distributed and*

Network-Based Processing (PDP), 2012 20th Euromicro International Conference on, pages 206 –212, February 2012.

[20] Fareed Jokhio, Adnan Ashraf, Tewodros Deneke, Sebastien Lafond, Ivan Porres, and Johan Lilius. *Developing Cloud Software: Algorithms, Applications, and Tools*, chapter Proactive Virtual Machine Allocation for Video Transcoding in the Cloud, pages 113–143. Turku Centre for Computer Science (TUCS) General Publication Number 60, October 2013.

[21] Fareed Jokhio, Adnan Ashraf, Sebastien Lafond, and Johan Lilius. A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. In *Software Engineering and Advanced Applications (SEAA), 39th Euromicro Conference on*, pages 365–372, 2013.

[22] Fareed Jokhio, Adnan Ashraf, Sebastien Lafond, Ivan Porres, and Johan Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In *Parallel, Distributed and Network-Based Processing (PDP), 21st Euromicro International Conference on*, pages 254–261, 2013.

[23] Fareed Ahmed Jokhio, Tewodros Deneke, Sébastien Lafond, and Johan Lilius. Analysis of video segmentation for spatial resolution reduction video transcoding. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2011 International Symposium on*, pages 1–6, December 2011.

[24] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi-Li Zhang, and Yafei Dai. Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices. In *The 22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2012.

[25] Norman Matloff. *A Discrete-Event Simulation Course Based on the SimPy Language*. University of California at Davis, 2006.

[26] D.C. Montgomery, E.A. Peck, and G.G. Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2012.

[27] W. Pan, D. Mu, H. Wu, and L. Yao. Feedback control-based QoS guarantees in web application servers. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 328 –334, September 2008.

[28] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. A multi-model framework to implement self-managing control systems for QoS management. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, pages 218–227, 2011.

[29] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500 –507, July 2011.

[30] Ilhoon Shin and Kern Koh. Hybrid transcoding for QoS adaptive video-on-demand services. *IEEE Trans. on Consum. Electron.*, 50(2):732–736, May 2004.

[31] A. Vetro, C. Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *Signal Processing Magazine, IEEE*, 20(2):18 – 29, March 2003.

[32] J. Watkinson. *The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4*. Broadcasting and communications. Elsevier/Focal Press, 2004.

[33] T. Wiegand, G. J. Sullivan, and A. Luthra. Draft ITU-T recommendation and final draft international standard of joint video specification. Technical report, 2003.

[34] Andreas Wolke and Gerhard Meixner. TwoSpot: A cloud platform for scaling out web applications dynamically. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin / Heidelberg, 2010.

[35] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12, 2010.

[36] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. Computation and storage trade-off for cost-effectively storing scientific datasets in the cloud. In Borko Furht and Armando Escalante, editors, *Handbook of Data Intensive Computing*, pages 129–153. Springer New York, 2011.

[37] Dong Yuan, Yun Yang, Xiao Liu, and Jinjun Chen. A local-optimisation based strategy for cost-effective datasets storage of scientific applications in the cloud. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 179–186, 2011.

[38] Dong Yuan, Yun Yang, Xiao Liu, Gaofeng Zhang, and Jinjun Chen. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience*, 24(9):956–976, 2012.

Paper V

Bit Rate Reduction Video Transcoding with Distributed Computing

Fareed Jokhio and Tewodros Deneke and Sébastien Lafond and Johan Lilius

Originally published in proceedings of *the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2012)*. IEEE Computer Society, pp. 206-212, 15-17 February 2012, Garching, Germany.

Paper VI

Analysis of Video Segmentation for Spatial Resolution Reduction Video Transcoding

Fareed Jokhio and Tewodros Deneke and Sébastien Lafond and Johan Lilius

Originally published in proceedings of *the Internation Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2011)*, IEEE, pp. 1-6, December 7-9, 2011, Chiang Mai, Thailand.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Division for Natural Sciences and Technology
- Department of Information Technologies

ISBN 978-952-12-3004-2
ISSN 1239-1883

Fareed Ahmed Jokhio

Video Transcoding in a Distributed Cloud Computing Environment