Adnan Ashraf

Cost-Efficient Virtual Machine Management

Provisioning, Admission Control, and Consolidation

Turku Centre *for* Computer Science

TUCS Dissertations
No 183, October 2014

Cost-Efficient Virtual Machine Management

Provisioning, Admission Control, and Consolidation

Adnan Ashraf

To be presented, with the permission of the Department of Information Technologies at Åbo Akademi University, for public criticism in Auditorium Alpha in the ICT Building, on October 09, 2014, at 11:00 a.m.

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

2014

Supervisor

Professor Ivan Porres
Department of Information Technologies
Åbo Akademi University
Joukahaisenkatu 3-5, 20520 Turku
Finland

Reviewers

Dr. Radu Calinescu
Senior Lecturer in Large-Scale Complex IT Systems
Department of Computer Science
University of York
Deramore Lane, York YO10 5GH
United Kingdom

Professor Kari Systä
Department of Pervasive Computing
Tampere University of Technology
Korkeakoulunkatu 1, 33720 Tampere
Finland

Opponent

Dr. Radu Calinescu
Senior Lecturer in Large-Scale Complex IT Systems
Department of Computer Science
University of York
Deramore Lane, York YO10 5GH
United Kingdom

ISBN 978-952-12-3111-7
ISSN 1239-1883

To my mother and late father.

Abstract

One of the main challenges in Software Engineering is to cope with the transition from an industry based on software as a product to software as a service. The field of Software Engineering should provide the necessary methods and tools to develop and deploy new cost-efficient and scalable digital services. In this thesis, we focus on deployment platforms to ensure cost-efficient scalability of multi-tier web applications and on-demand video transcoding service for different types of load conditions.

Infrastructure as a Service (IaaS) clouds provide Virtual Machines (VMs) under the pay-per-use business model. Dynamically provisioning VMs on demand allows service providers to cope with fluctuations on the number of service users. However, VM provisioning must be done carefully, because over-provisioning results in an increased operational cost, while under-provisioning leads to a subpar service. Therefore, our main focus in this thesis is on cost-efficient VM provisioning for multi-tier web applications and on-demand video transcoding. Moreover, to prevent provisioned VMs from becoming overloaded, we augment VM provisioning with an admission control mechanism. Similarly, to ensure efficient use of provisioned VMs, web applications on the under-utilized VMs are consolidated periodically. Thus, the main problem that we address is cost-efficient VM provisioning augmented with server consolidation and admission control on the provisioned VMs. We seek solutions for two types of applications: multi-tier web applications that follow the request-response paradigm and on-demand video transcoding that is based on video streams with soft realtime constraints.

Our first contribution is a cost-efficient VM provisioning approach for multi-tier web applications. The proposed approach comprises two sub-approaches: a reactive VM provisioning approach called ARVUE and a hybrid reactive-proactive VM provisioning approach called Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling. Our second contribution is a prediction-based VM provisioning approach for on-demand video transcoding in the cloud. Moreover, to prevent virtualized servers from becoming overloaded, the proposed VM provisioning approaches are augmented with admission control approaches. Therefore, our third contribution is a session-based admission control approach for multi-tier web

applications called adaptive Admission Control for Virtualized Application Servers. Similarly, the fourth contribution in this thesis is a stream-based admission control and scheduling approach for on-demand video transcoding called Stream-Based Admission Control and Scheduling. Our fifth contribution is a computation and storage trade-off strategy for cost-efficient video transcoding in cloud computing. Finally, the sixth and the last contribution is a web application consolidation approach, which uses Ant Colony System to minimize the under-utilization of the virtualized application servers.

Sammanfattning

En av de största utmaningarna i programvaruproduktion är hanteringen av övergången från en industri baserad på mjukvara som produkt till mjukvara som tjänst. Forskningsområdet för programvaruproduktion bör tillhandahålla de nödvändiga metoderna och verktygen för att utveckla och ta nya kostnadseffektiva och skalbara digitala tjänster i bruk. Denna avhandling fokuserar på utplaceringsplattformer för säkerställandet av kostnadseffektiv skalning av webbapplikationer i flera lager och efterfrågansstyrda videotranskodningstjänster för olika sorters belastning.

Datormoln som tillhandahåller infrastruktur enligt en tjänstemodell erbjuder virtuella maskiner enligt en affärsmodell där kunden betalar enligt bruk. Dynamisk anskaffning av virtuella maskiner enligt behov låter tjänsteleverantörerna hantera fluktuationer i antalet användare. Anskaffning av virtuella maskiner måste dock göras försiktigt, eftersom överanskaffning leder till ökad verksamhetskostnad, medan underanskaffning leder till försämrad tjänstekvalitet. Därför ligger huvudfokus i denna avhandling på kostnadseffektiv anskaffning av virtuella maskiner för webbapplikationer i flera lager och efterfrågansstyrd videotranskodning. För att förhindra att virtuella maskiner blir överbelastade utökas anskaffningen med en tillträdeskontrollsmekanism. För att säkerställa effektiv användning av de virtuella maskinerna konsolideras applikationer på underbelastade sådana periodvis. Således är det främsta problemet kostnadseffektiv anskaffning av virtuella maskiner utökad med tillträdeskontroll och serverkonsolidering. Denna avhandling söker lösningar för två tillämpningar: Webbapplikationer i flera lager som följer begäran-svarsparadigmen och efterfrågansstyrd videotranskodning baserad på videoströmmar med mjuka realtidsbegränsningar.

Den första kontributionen är ett kostnadseffektivt tillvägagångssätt för anskaffning av virtuella maskiner för webbapplikationer i flera lager. Det föreslagna tillvägagångssättet består av två delar: Ett reaktivt tillvägagångssätt för anskaffning av virtuella maskiner kallat ARVUE och en reaktiv-prediktiv hybridmetod kallad Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling. Den andra kontributionen är ett prediktionsbaserat tillvägagångssätt för anskaffning av virtuella maskiner för efterfrågansstyrd videotranskodning i ett datormoln.

För att förhindra överbelastning av servrarna har tillvägagångssätten för anskaffning utökats med tillträdeskontroll. Således är den tredje kontributionen sessionsbaserad tillträdeskontroll för webbapplikationer i flera lager kallad Adaptive Admission Control for Virtualized Application Servers. På liknande sätt är den fjärde kontributionen ett tillvägagångssätt för strömbaserad tillträdeskontroll och schemaläggning för efterfrågansstyrd videotranskodning kallat Stream-Based Admission Control and Scheduling. Den femte kontributionen är en beräknings- och lagringsavvägningsstrategi för kostnadseffektiv videotranskodning i ett datormoln. Den sjätte och sista kontributionen är ett tillvägagångssätt för konsolidering av webbapplikationer som använder Ant Colony System-metoden för att minimera graden av underbelastning hos de virtuella maskinerna.

Acknowledgments

First and foremost, I thank Allah (God), the Almighty, for giving me an excellent opportunity to pursue the highest degree in my field of study and providing me with the means to successfully complete my thesis.

I am glad that this day has arrived when I am writing this final part of my thesis. This thesis is the result of the efforts and support of many people and it gives me an immense pleasure to express my sincere and deepest gratitude to all those who helped me complete my thesis.

I want to thank Professor Ivan Porres for his guidance, encouragement, and kind supervision. Ivan is an excellent supervisor, an outstanding researcher, and a wonderful person. It has been a privilege and an honor for me to work closely with him. During the course of my PhD, Ivan provided his full support through regular weekly meetings and fruitful discussions. He not only gave me some excellent and concrete ideas to work on, but also encouraged my own ideas and helped me in all aspects of research and publishing.

I also wish to thank Dr. Radu Calinescu and Professor Kari Systä for their time and efforts to review my thesis and for providing valuable feedback and constructive comments, which helped me prepare the final manuscript of my thesis. I am also grateful to Dr. Radu Calinescu for his kind acceptance to act as the opponent at my doctoral defence.

I would like to thank all of my coauthors for their efforts, many interesting discussions, and fruitful collaborations. In particular, I am grateful to Benjamin Byholm, Fareed Ahmed Jokhio, Sébastien Lafond, Professor Johan Lilius, Tewodros Deneke, and Niclas Snellman at Åbo Akademi University, Fahimeh Farahnakian at University of Turku, Joonas Lehtinen and Marc Englund at Vaadin Ltd., and Professor Tommi Mikkonen and Timo Aho at Tampere University of Technology. I am also thankful to Benjamin Byholm and Fareed Ahmed Jokhio for their useful feedback on an initial draft of my thesis.

I would like to extend my gratitude to my teachers at the Department of Information Technologies, especially Dragos Truscan, Marta Olszewska, Elena Troubitsyna, Professor Barbro Back, and Professor Pirkko Walden. I also wish to thank my current and past colleagues in the Software En-

gineering Laboratory, particularly Ali Hanzala Khan, Benjamin Byholm, Irum Rauf, Dragos Truscan, Fredrik Abbors, Niclas Snellman, Thomas Fors, Kristian Nybom, Marta Olszewska, Tanwir Ahmad, Max Weijola, Mehdi Nobakht, and Jeanette Heidenberg. Furthermore, I want to acknowledge the support of the administrative and technical personnel at the Department of Information Technologies.

I am highly grateful and honored to receive generous scholarships from the Higher Education Commission (HEC) of Pakistan, Nokia Foundation, Åbo Akademi University, and Ulla Tuominen Foundation. I would also like to express my gratitude to Turku Centre for Computer Science (TUCS) for providing travel grants to support my conference and educational trips.

I also wish to thank my teachers and my current and past colleagues in Pakistan. In particular, I am grateful to Dr. Naveed Ikram, Dr. Muhammad Jaffar-ur-Rehman (late), Dr. Aamer Nadeem, Dr. Muhammad Sher, Muhammad Usman, Usman Nasir, and Sameer Akram.

Last but not least, I thank my family for their endless love, incredible support, and prayers. I wish to express my deepest gratitude to my mother and late father for their selfless love and profound dedication, my both sisters and both brothers for their constant encouragement and support, my wife for her unwavering love and understanding, and my daughter and son for filling my days with joy and smiles.

Adnan Ashraf
Turku, October 2014

List of Included Publications

This thesis is based on the following original publications. The publication reprints of the included publications are presented in Part II of the thesis.

I Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres. Feedback Control Algorithms to Deploy and Scale Multiple Web Applications per Virtual Machine. In *Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2012)*, pp. 431–438, September 2012, Cesme, Izmir, Turkey.

II Adnan Ashraf, Benjamin Byholm, and Ivan Porres. A Session-Based Adaptive Admission Control Approach for Virtualized Application Servers. In *Proceedings of the 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012)*, pp. 65–72, November 2012, Chicago, IL, USA.

III Adnan Ashraf, Benjamin Byholm, and Ivan Porres. CRAMP: Cost-Efficient Resource Allocation for Multiple Web Applications with Proactive Scaling. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pp. 581–586, December 2012, Taipei, Taiwan.

IV Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing. In *Proceedings of the 21st EUROMICRO International Conference on Parallel, Distributed and Network-based Processing (PDP 2013)*, pp. 254–261, February 2013, Belfast, UK.

V Adnan Ashraf, Fareed Ahmed Jokhio, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, pp. 482–489, May 2013, Delft, the Netherlands.

VI Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius. A Computation and Storage Trade-Off Strategy for Cost-Efficient Video Transcoding in the Cloud. In *Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2013)*, pp. 365–372, September 2013, Santander, Spain.

VII Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing. *Turku Centre for Computer Science (TUCS) Technical Reports*, number 1098, pp. 1–25, December 2013.

VIII Adnan Ashraf and Ivan Porres. Using Ant Colony System to Consolidate Multiple Web Applications in a Cloud Environment. In *Proceedings of the 22nd EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP 2014)*, pp. 482–489, February 2014, Turin, Italy.

Contents

List of Figures

List of Tables

Part I

Research Summary

Chapter 1

Introduction

One of the main challenges and opportunities in the field of Software Engineering is to cope with the transition from an industry based on software as a product to Software as a Service (SaaS) [7]. The field of Software Engineering should provide the necessary methods and tools to develop and deploy new cost-efficient, scalable, reliable, and secure digital services. In this thesis, we focus on deployment platforms to ensure scalability [60] of multi-tier web applications and on-demand soft realtime video transcoding service for different types of load conditions.

Web applications are often deployed in a three-tier computer architecture that consists of client, application, and database tiers [10]. The client tier runs within the user web browser, while the application server and the database server tiers run in the remote server infrastructure. Both the application and the database tiers are implemented using a computer cluster to be able to process many user requests simultaneously. In this configuration, a load balancing subsystem distributes the user requests among the computers in the cluster. Traditionally, these clusters are composed of a fixed number of computers and are dimensioned to serve a predetermined maximum number of concurrent users.

A web-based video streaming service is also implemented using a cluster-based distributed system, which may consist of different types of servers, such as, video streaming servers and video transcoding servers. A video transcoding server converts a compressed video from one format to another [93]. It may change video format, bit rate, frame resolution, frame rate, or any combination of these [66]. Video transcoding is a compute-intensive operation. For an on-demand video streaming service, it may be necessary to transcode a large number of videos on-the-fly under soft realtime constraints. Transcoding of a large number of simultaneous videos necessitates a cluster of video transcoding servers.

Infrastructure as a Service (IaaS) clouds, such as Amazon Elastic Compute Cloud (EC2)[1], provide Virtual Machines (VMs) under the pay-per-use business model. Dynamically provisioning VMs on demand allows IaaS users to deploy and scale their web applications and video transcoding service without requiring to invest into large-scale Information Technology (IT) infrastructures. With cloud elasticity, it is possible to create a dynamically scalable cluster of servers consisting of a varying number of VMs. However, VM provisioning must be done carefully because over-provisioning results in an increased operational cost, while under-provisioning leads to a sub-par service, which may violate users' Quality of Service (QoS) requirements concerning performance, resulting in a loss of revenue.

Determining the number of VMs to provision for a cluster is an important problem as the exact number of VMs needed at a specific time depends upon the user load and the QoS requirements, which are specified in the Service Level Agreements (SLAs). In this thesis, our main goal is cost-efficient VM provisioning for multi-tier web applications and video transcoding. Moreover, to prevent provisioned VMs from becoming overloaded, we augment VM provisioning with an admission control mechanism. For cost-efficiency, it is also necessary to reduce under-utilization of servers in a cluster. As a recent study on physical servers showed that the under-utilization of servers in enterprises is a matter of concern [94]. Under-utilization of VMs can be reduced by using server consolidation techniques similar to those used in data centers for power-efficiency [42, 77, 94].

1.1 Research Questions

The main problem that we intend to tackle is cost-efficient VM provisioning augmented with server consolidation and admission control on the provisioned VMs. We seek solutions for two types of applications: multi-tier web applications that follow the request-response paradigm and on-demand video transcoding that is based on video streams with soft realtime constraints. Although there are many similarities between VM provisioning for web applications and VM provisioning for video transcoding, each one of them also has its own challenges [9]. Some of the common challenges for web applications and video transcoding include: ensuring dynamic scaling under different load conditions, handling VM provisioning delay, preventing servers from becoming overloaded, making load predictions under soft real-time constraints, reducing oscillations in the number of provisioned VMs, and reducing under-utilization of VMs. Moreover, for web application consolidation, the main challenge is to reduce both the total number of VMs and the number of application migrations. Similarly, important challenges

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

for video transcoding include: preventing transcoding jitters in the admitted video streams and providing a good trade-off between the computation cost and the storage cost.

The existing VM provisioning approaches for web-based systems [6, 26, 31,39,48,51,52,57,62,78,79,81,83,99,107,108] tend to use dedicated hosting on the VM level, where each VM is used exclusively for one particular web application. However, the main drawback of dedicated hosting is that it prohibits sharing of VM resources among multiple concurrent web applications. In contrast, the shared hosting [91] of web applications provides improved VM resource utilization by allowing deployment of multiple web applications on a VM [13]. Similarly, the existing admission control approaches for web-based systems [3,29,30,58,76,82,85,95] tend to use the traditional on-off control and request-based admission. Moreover, most of them rely on rejection of requests to prevent servers from becoming overloaded. All of the existing VM provisioning and admission control approaches discussed in this thesis were originally proposed for web-based systems. To the best of our knowledge, there are currently no existing VM provisioning and admission control approaches for video transcoding in cloud computing. Similarly, the existing server consolidation approaches [18, 19, 33, 40–46, 54, 61, 71, 72, 77, 94, 96, 100] are used in data centers to consolidate VMs on Physical Machines (PMs). To the best of our knowledge, there is currently no existing work on consolidating multi-tier web applications on VMs. The Research Questions (RQs) that motivated this thesis are as follows:

- RQ1: How to ensure scalability of multi-tier web applications and on-demand video transcoding service for different types of load conditions while providing a good trade-off between performance and cost?

- RQ2: How to cost-efficiently prevent servers from becoming overloaded?

- RQ3: How to provide a good trade-off between the computation cost and the storage cost when using a public IaaS cloud for on-demand video transcoding?

- RQ4: How to consolidate multi-tier web applications on under-utilized VMs to reduce under-utilization of the virtualized application servers in a cloud-based shared hosting environment?

The above RQs are addressed in detail in the original publications in Part II of the thesis.

1.2 Overview of Research Contributions

In this thesis, we propose a cost-efficient VM provisioning approach for multi-tier web applications [2,10,11] and on-demand video transcoding [66]. Moreover, to prevent virtualized servers from becoming overloaded, the proposed VM provisioning approach is augmented with an admission control mechanism [12, 15]. Similarly, the under-utilization of the virtualized application servers is minimized by providing a web application consolidation approach [16]. We also present a computation and storage trade-off strategy for cost-efficient video transcoding in cloud computing [64, 65]. These contributions are presented in detail in the original publications in Part II of the thesis. A brief overview of the main contributions is presented in the following subsections.

1.2.1 VM Provisioning for Multi-tier Web Applications

As mentioned earlier, determining the number of VMs to provision for a dynamic cluster of virtualized servers is an important problem as the exact number of VMs needed at a specific time depends upon the user load and the QoS requirements. In this thesis, our first contribution is a cost-efficient VM provisioning approach for multiple multi-tier web applications. The proposed approach comprises two sub-approaches: a reactive VM provisioning approach called ARVUE[2] [2, 10] and a hybrid reactive-proactive VM provisioning approach called Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling (CRAMP) [11]. ARVUE uses a reactive feedback control loop to scale multiple web applications on a given IaaS cloud. Since it is a reactive approach, the VM provisioning decisions are based on the current and past load conditions. CRAMP is similar to ARVUE, but it uses a hybrid reactive-proactive control loop.

In comparison to existing solutions, the proposed approach provides automatic deployment and scaling of multiple simultaneous web applications on a given IaaS cloud in a shared hosting [91] environment. It monitors and uses resource utilization metrics and does not require a performance model of the applications or the infrastructure dynamics. The shared hosting environment allows us to share VM resources among deployed applications, reducing the total number of required VMs. Performance under varying load conditions is guaranteed by automatic adjustment and tuning of the CRAMP parameters.

[2]The name ARVUE is not an acronym. It was invented by Marc Englund at Vaadin Ltd. (https://vaadin.com). According to him, ARVUE is a wordplay on the term *our view*.

1.2.2 VM Provisioning for Video Transcoding

The second contribution of this thesis is a novel VM provisioning approach for video transcoding in the cloud [66]. The proposed approach uses load prediction to proactively scale video transcoding service on a given IaaS cloud. It provides mechanisms for allocation and deallocation of VMs to a cluster of video transcoding servers in a horizontal fashion. We use a two-step load prediction method [4, 5], which allows proactive VM provisioning with high prediction accuracy under soft realtime constraints. For cost-efficiency, our work supports transcoding of multiple on-demand video streams concurrently on a single VM, resulting in a reduced number of required VMs. We use video segmentation at Group of Pictures (GOP) level, which splits video streams into smaller segments that can be transcoded independently of one another.

1.2.3 Admission Control for Multi-tier Web Applications

Our third contribution is a session-based admission control approach called adaptive Admission Control for Virtualized Application Servers (ACVAS) [12]. ACVAS uses measured and predicted resource utilizations of a server to make admission control decisions for new user sessions. Instead of using the traditional *on-off* control [30], it implements per-session admission control [76], which reduces the risk of over-admission. Moreover, instead of relying only on rejection of new sessions, ACVAS takes benefit of the cloud elasticity to implement a simple session deferment mechanism that reduces the number of rejected sessions while increasing session throughput. Thus, compared to existing approaches that tend to rely only on request rejection, each admission control decision in ACVAS has three possible outcomes: admit, defer, or reject. Performance under varying load conditions is guaranteed by automatic adjustment and tuning of the ACVAS parameters.

1.2.4 Admission Control and Scheduling for Video Transcoding

For video transcoding, we present a novel admission control approach called Stream-Based Admission Control and Scheduling (SBACS) [15]. SBACS uses queue waiting time of transcoding servers to make admission control decisions for incoming video streams. It implements stream-based admission control with per-stream admission. To ensure efficient utilization of the transcoding servers, video streams are segmented at the GOP level. In addition to the traditional rejection policy, SBACS also provides a stream deferment policy, which exploits cloud elasticity to allow temporary deferment of the incoming video streams. In other words, the admission controller can decide to admit, defer, or reject an incoming stream and hence reduce

the rejection rate. In order to prevent *transcoding jitters* in the admitted streams, we introduce a job scheduling mechanism, which may drop a small proportion of video frames from a video segment to ensure continued delivery of the video contents to the user.

1.2.5 Computation and Storage Trade-off Strategy

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. Moreover, storage of multiple transcoded versions of each source video requires a large amount of disk space. As mentioned earlier, IaaS clouds provide 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.

We present a novel computation and storage trade-off strategy for cost-efficient video transcoding in the cloud called cost and popularity score based strategy [64,65]. 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.

1.2.6 Web Application Consolidation using ACO

The under-utilization of VMs becomes more pertinent when a SaaS or a Platform as a Service (PaaS) provider wants to leverage an IaaS cloud to cost-efficiently deploy a large number of web applications of varying resource needs. The solution to this problem is to create a dynamically scalable application server tier that manages multiple applications simultaneously, while using shared hosting [91] to deploy multiple applications on a VM [10, 11]. In comparison to the traditional dedicated hosting of web applications where each VM is used exclusively for one particular web application, the shared hosting of web applications allows improved VM utilization by sharing VM resources among multiple concurrent web applications. However, in a shared hosting environment, dynamic scaling alone does not minimize over-provisioning of VMs.

We present a novel approach to consolidate multiple web applications in a cloud-based shared hosting environment [16]. The proposed approach uses Ant Colony Optimization (ACO) [37, 38] to build a web application migration plan, which is then used to minimize over-provisioning of VMs by consolidating web applications on under-utilized VMs.

1.3 Research Settings

The research work presented in this thesis was carried out within the context of the Cloud Software Program 2010-2013[3]. Cloud Software was a Finnish research program, whose goal was to significantly improve the competitive position of the Finnish software-intensive industry in the global markets in the field of cloud computing. More than 30 Finnish IT companies and research organizations participated in the program.

The results presented in this thesis are an outcome of the research collaborations between Åbo Akademi University, its academic partner Tampere University of Technology, and industry partners Vaadin Ltd.[4] and Bambuser AB[5].

1.4 Validation of the Research Work

A convenient and quick way of testing new algorithms and solutions involving complex environments is to write and run software simulations [12]. A special kind of simulations called discrete-event simulations are most appropriate for simulating and evaluating cluster, grid, and cloud computing environments and systems [24].

A discrete-event simulation [17] contains a relatively detailed representation of the internal components of a system and their interactions. It may represent state variables, events, resources, processes, objects and their attributes, sets, and queues, among others. In comparison to the traditional mathematical and analytical models that tend to represent a system at a fixed point in time, a discrete-event simulation is run by a mechanism that imitates the actual clock time. Moreover, the state variables in a discrete-event simulation change their values at discrete points in time at which some events occur. Therefore, discrete-event simulations model the dynamic behavior of a system, where the passage of time and the occurrences of events play important roles.

We have developed discrete-event simulations to validate our proposed VM provisioning approach for video transcoding (Section 1.2.2), admission control approach for multi-tier web applications (Section 1.2.3), admission control and scheduling approach for video transcoding (Section 1.2.4), computation and storage trade-off strategy for video transcoding (Section 1.2.5), and web application consolidation approach (Section 1.2.6). The evaluation comprises a series of experiments involving synthetic as well as realistic load patterns. Moreover, for our VM provisioning approach for multi-tier web

[3] http://www.cloudsoftwareprogram.org/
[4] https://vaadin.com
[5] http://bambuser.com/

applications (Section 1.2.1), we have developed a prototype implementation, which has been evaluated in another series of experiments that involve a synthetic load pattern.

A thorough and sound simulation study consists of a set of steps, such as problem formulation, model conceptualization, data collection, coding, verification and validation of the simulation model, experimental design and setup, simulation runs, and analysis of the results [17]. For the credibility and acceptability of the simulation results, it is important that these steps are performed carefully, correctly, and rigorously. Therefore, we took all necessary steps to ensure the credibility and acceptability of our simulation results. The problem formulation, model conceptualization, data collection, and validation of the simulation models were performed with an active participation from our industry partners Vaadin Ltd. and Bambuser AB. The data used in the simulations comprise synthetic as well as realistic load patterns. The realistic load patterns were used to provide representative results under real load conditions. Moreover, the synthetic load patterns were designed to simulate a richer set of scenarios. The experiments were repeated several times to ensure that they are deterministic. In addition, each experiment was preceded by a series of preliminary experiments that were performed to obtain appropriate values for the experiment parameters. Furthermore, in each simulation, the system components and their interactions were modeled at a relatively lower level of abstraction, which allowed detailed analyses of the important aspects pertaining to the dynamic behavior of the system.

1.5 Thesis Organization

The thesis consists of two parts. Part I provides a research summary, while Part II presents the original publications. Part I consists of five chapters. Chapter 1 builds motivation of this work and presents RQs, a brief overview of research contributions, research settings, and our approach to validate research work. Chapter 2 provides background and discusses important related works. Chapter 3 presents a summary of the main contributions, while focusing on the challenges that they address. Chapter 4 provides a description and organization of the original publications and provides a mapping between the publications and the RQs. Finally, we present our conclusions and some future directions in Chapter 5.

Chapter 2

Background and Related Work

In this chapter, we first provide a brief overview of the background concepts and technologies on which this thesis is based. These include cloud computing, multi-tier web applications, and video transcoding. Then, we present the most important related works on VM provisioning approaches, admission control approaches, computation and storage trade-off strategies, and consolidation approaches.

2.1 Cloud Computing

Cloud computing is a relatively new and emerging computing paradigm, which promises to deliver computing as the fifth utility [22]. It leverages several existing concepts and technologies such as data centers, clusters, grids, and hardware virtualization, among others and gives them a new perspective and identity. From a business perspective, cloud computing provides a pay-per-use business model which opens new avenues for the development, deployment, and scaling of digital services. The dynamic on-demand provisioning of computing resources allows companies to deploy their web applications and web services without requiring an upfront investment into large-scale IT infrastructures. Moreover, with cloud elasticity, it is possible to create a dynamically scalable cluster of servers, which scales up and down based on the prevailing load conditions. A frequently cited paper by Vaquero et al. [92] presented the following encompassing definition of the cloud after studying more than twenty different definitions:

> Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum

> resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs [92].

Perhaps, the main distinguishing characteristics of cloud computing are those enumerated in [14]:

- Computing resources can be purchased on-demand from a seemingly unlimited supply.
- The capital expenses needed to purchase computing resources upfront are changed to operational expenses, shifting the capital investment risk for under or over provisioning to the cloud computing vendor.
- Computing is priced with a pay-as-you-go pricing model, where capacity can be scaled up and down on a short term basis.

The National Institute of Standards and Technology (NIST) has also emphasized on the elasticity of computing resources in their definition of cloud computing [73]:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. [73]

The five essential characteristics in the NIST definition are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [73]. The service models are IaaS, PaaS, and SaaS. Similarly, the deployment models are private cloud, community cloud, public cloud, and hybrid cloud. Figure 2.1 summarizes the service and deployment models in cloud computing. It also shows that the three service models can operate on top of any of the four deployment models [75, 88]. Armbrust et al. [7] provided an excellent overview of the technology drivers behind cloud computing. They also discussed the main obstacles and opportunities in cloud computing.

An IaaS cloud, such as Amazon Web Services (AWS)[1], allows its users to provision processing, storage, network, and other fundamental computing resources. The users may deploy and run arbitrary software including operating systems, applications, and web-based services. A PaaS cloud allows its users to deploy and run applications created or tailored according

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

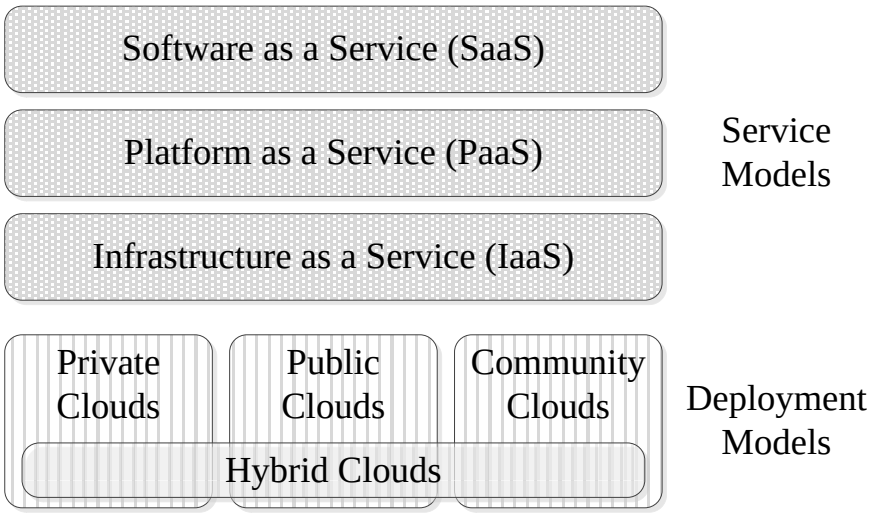

Figure 2.1: Service and deployment models in cloud computing

to the programming languages, frameworks, libraries, tools, and services supported by the provider, for example, Google App Engine (GAE)[2]. The users control their deployed applications. However, they do not control the underlying operating system, storage, servers, and networks. Similarly, a SaaS cloud allows its users to use certain applications provided by the cloud provider. The users may have only a limited control on some user-specific application configuration settings.

2.2 Multi-tier Web Applications

Web applications are often deployed in a three-tier computer architecture that consists of client, application, and database tiers [10]. Figure 2.2 presents an overview of the three-tier computer architecture for web application deployment. The client tier runs within the user web browser, while the application server and the database server tiers run in the remote server infrastructure. Both the application and the database tiers are implemented using a computer cluster to be able to process many user requests simultaneously. In this configuration, a load balancing subsystem distributes the user requests among the computers in a cluster. Traditionally, these clusters are composed of a fixed number of computers and are dimensioned to serve a predetermined maximum number of concurrent users.

With the advent of cloud elasticity, it is now possible to create a dynamically scalable cluster of servers consisting of a varying number of VMs.

[2]https://cloud.google.com/products/app-engine/

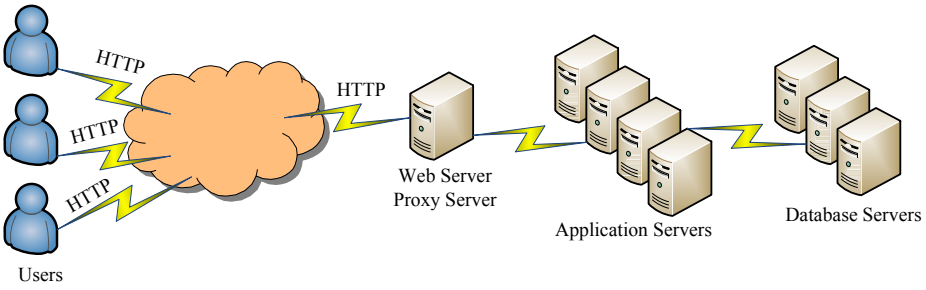

Figure 2.2: Three-tier web applications

However, VM provisioning must be done carefully because over-provisioning results in an increased operational cost, while under-provisioning leads to a subpar service.

2.3 Video Transcoding

Streaming of digital videos is increasingly common among the Internet users. Video streaming of a large number of videos may require a lot of resources on the server-side. Therefore, for efficient use of storage and transmission media, digital videos are often stored and transmitted in compressed formats, such as MPEG-4 [97] and H.264 [98]. Client-side devices that are used to play videos usually support only a subset of the existing video formats. A video on the server-side may be stored in a different format than those supported by a target device. Therefore, the video must be converted into a format that is supported by the target device [86]. The process of converting a digital video from one compressed format to another is known as video transcoding [15, 93]. It may change video format, bit rate, frame size, frame rate, or any combination of these [66].

Video transcoding involves decoding and encoding processes. It is a compute-intensive process, usually performed at the server-side [63]. Video transcoding may be done in soft realtime 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 soft realtime [89]. There are different types of video transcoding, for example, bit-rate reduction transcoding is used to meet network bandwidth availability, spatial resolution reduction transcoding is used for display size adoption, and temporal resolution reduction transcoding is used for frame rate reduction [28, 63, 93, 101].

A web-based video streaming service is implemented using a cluster-based distributed system, which may consist of different types of servers, such as, video streaming servers and video transcoding servers. Since video

transcoding is a compute-intensive operation, transcoding of a large number of simultaneous video streams requires a large-scale cluster of transcoding servers. Moreover, to handle different load conditions in a cost-efficient manner, the cluster should be dynamically scalable. IaaS clouds provide all necessary computing resources to create a dynamically scalable cluster of transcoding servers. However, the main challenge is to devise appropriate algorithms and mechanisms, which should provide cost-efficient, dynamically scalable video transcoding in the cloud.

Distributed video transcoding with video segmentation at the GOP level was proposed in [67] and [68]. Jokhio et al. [68] presented bit rate reduction video transcoding using multiple processing units. In [67], different video segmentation methods were analyzed to perform spatial resolution reduction video transcoding. However, in both papers [67, 68], video transcoding was not performed in the cloud and the VM provisioning problem was not addressed. In contrast, the main focus of our VM provisioning approach for video transcoding (Section 3.2) is on VM provisioning algorithms.

Huang et al. [59] presented a cloud-based video proxy to deliver transcoded videos for streaming. The main contribution of their work is a multilevel transcoding parallelization framework. They used Hallsh-based and Lateness-first mapping to optimize transcoding speed and to reduce transcoding jitters. Li et al. [70] proposed a cloud transcoder, which uses a compute cloud as an intermediate platform to provide transcoding service. However, none of these papers [59, 70] addressed the VM provisioning problem for video transcoding in cloud computing.

Zhu et al. [109] presented a framework for multimedia cloud computing. It addresses two perspectives of the multimedia cloud computing namely multimedia-aware cloud and cloud-aware multimedia. For video transcoding in the multimedia-aware cloud, they presented a cloud-based video adaptation and transcoding framework. However, they did not address the VM provisioning problem for soft realtime video streams. Garcia et al. [47] used a Hadoop-based[3] cloud to aid transcoding of media content. Their results show that the processing power of a Hadoop cluster can greatly reduce encoding times. Breitman et al. [21] and Pereira et al. [80] proposed a split and merge architecture for high performance video processing by generalizing the MapReduce paradigm [35]. They also considered the use of dynamic resource provisioning to reduce video encoding times. However, they did not propose a VM provisioning algorithm for video transcoding in the cloud.

[3] http://hadoop.apache.org/

2.4 VM Provisioning Approaches

Most of the existing works on VM provisioning for web-based systems can be classified into two main categories: plan-based approaches and control theoretic approaches [39, 78, 79, 83]. Plan-based approaches can be further classified into workload prediction approaches [6, 48, 81, 107] and performance dynamics model approaches [26, 31, 51, 52, 57, 62, 99, 108]. One common difference between all existing works discussed here and our proposed VM provisioning approach for multi-tier web applications (Section 3.1) is that our proposed approach uses shared hosting [13]. Another distinguishing characteristic of our approach is that in addition to VM provisioning for the application server tier, it also provides dynamic scaling of multiple web applications. In ARVUE [2, 10], we used shared hosting with reactive VM provisioning. Moreover, our hybrid reactive-proactive VM provisioning approach CRAMP [11] provides improved QoS with prediction-based VM provisioning. Similarly, a common difference between all of these VM provisioning approaches and our proposed VM provisioning approach for video transcoding (Section 3.2) 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 provisioning 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 provisioning under soft realtime constraints, and it does not depend upon performance and dynamics of the underlying system.

Ardagna et al. [6] proposed a distributed algorithm for managing SaaS cloud systems that addresses capacity allocation for multiple heterogeneous applications. Raivio et al. [81] used proactive resource allocation for short message services in hybrid clouds. The main drawback of their approach is that it assumes server processing capacity in terms of messages per second, which is not a realistic assumption for the Hypertext Transfer Protocol (HTTP) and video traffic where different types of requests may require different amounts of processing time. Zhang et al. [107] introduced a statistical-based resource allocation approach that performs load balancing on PMs by predicting VM resource demands. It uses statistical prediction and available resource evaluation mechanisms to make online resource allocation decisions. Gong et al. [48] presented a predictive resource scaling system, which leverages light-weight signal processing and statistical learning methods to predict resource demands of applications and adjusts resource allocations accordingly. Nevertheless, the main challenge in the prediction-based approaches is in making good prediction models that could ensure high prediction accuracy with low computational cost. In our proposed approach for multi-tier web applications (Section 3.1), CRAMP uses a two-step load

prediction method [4, 5] with Exponential Moving Average (EMA) and a simple linear regression model [12, 74], which provides high prediction accuracy under soft realtime constraints. Moreover, it gives more or less weight to the predicted utilizations based on the Normalized Root Mean Square Error (NRMSE). Similarly, our VM provisioning approach for video transcoding (Section 3.2) is a prediction-based approach, which uses the two-step load prediction method to predict video transcoding rate of the transcoding servers and then uses this information to provision preemptively.

TwoSpot [99] supports hosting of multiple web applications, which are automatically scaled up and down in a dedicated hosting environment. The scaling down is decentralized, which may lead to severe random drops in performance. Hu et al. [57] presented an algorithm for determining the minimum number of required servers, based on the expected arrival rate, service rate, and SLA. In contrast, our proposed VM provisioning approaches do not require knowledge about the infrastructure or performance dynamics. Chieu et al. [31] presented an approach that scales servers for a particular web application based on the number of active user sessions. However, the main challenge is in determining suitable threshold values on the number of user sessions. Carrera et al. [26] presented a utility-based web application placement approach to maximize application performance on clusters of PMs. Iqbal et al. [62] proposed an approach for multi-tier web applications, which uses response time and Central Processing Unit (CPU) utilization metrics to determine the bottleneck tier and then scales it by provisioning a new VM. Calinescu et al. [25] presented a tool-supported framework for QoS management and optimization of self-adaptive service-based systems. Zhao et al. [108] addressed the problem of minimizing resource rental cost for running elastic applications in the cloud while satisfying application-level QoS requirements. They proposed a deterministic resource rental planning model, which uses a mixed integer linear program to generate optimal rental decisions based on fixed cost parameters. They also presented a stochastic resource rental planning model that explicitly considers the price uncertainty of the Amazon EC2 spot instances in the rental decision making. However, they did not investigate cloud resource provisioning solutions for time-varying workloads. Han et al. [52] proposed a reactive resource allocation approach to integrate VM-level scaling with a more fine-grained resource-level scaling. Similarly, Han et al. [51] presented a cost-aware, workload-adaptive reactive scaling approach for multi-tier cloud applications. In contrast, our VM provisioning approach for video transcoding is a proactive approach. Moreover, CRAMP supports hybrid reactive-proactive VM provisioning with proportional and derivative factors to determine the number of VMs to provision.

Dutreilh et al. [39] and Pan et al. [78] used control theoretic models to design resource allocation solutions for cloud computing. Dutreilh et al.

presented a comparison of static threshold-based and reinforcement learning techniques. Pan et al. used Proportional-Integral (PI)-controllers to provide QoS guarantees. Patikirikorala et al. [79] proposed a multi-model framework for implementing self-managing control systems for QoS management. The work is based on a control theoretic approach called the Multi-Model Switching and Tuning adaptive control. Roy et al. [83] presented a look-ahead resource allocation algorithm based on the model predictive control. A common characteristic of the control theoretic approaches is that they depend upon performance and dynamics of the underlying system. In contrast, our proposed VM provisioning approaches for web applications and video transcoding do not require any knowledge about the performance models or infrastructure dynamics.

2.5 Admission Control Approaches

Admission control refers to the mechanism of restricting the incoming user load on a server in order to prevent it from becoming overloaded. Server overload prevention is important because an overloaded server fails to maintain its performance, which translates into a subpar service (higher response time and lower throughput) [49]. Thus, if an overloaded server keeps on accepting new user requests, then not only the new users, but also the existing users may experience a deteriorated performance.

The existing works on admission control for web-based systems can be classified according to the scheme presented in Almeida et al. [3]. For instance, Robertsson et al. [82] and Voigt and Gunningberg [95] are control theoretic approaches, while Huang et al. [58] and Muppala and Zhou [76] use machine learning techniques. Similarly, Cherkasova and Phaal [30], Almeida et al. [3], Chen et al. [29], and Shaaban and Hillston [85] are utility-based approaches.

Almeida et al. [3] proposed a joint resource allocation and admission control approach for a virtualized platform hosting a number of web applications, where each VM runs a dedicated web service application. Their admission control approach uses request-based admission, in which the optimization objective is to maximize the provider's revenue, while satisfying the customers' QoS requirements and minimizing the cost of resource utilization. It dynamically adjusts the fraction of capacity assigned to each VM and limits the incoming workload by serving only the subset of the requests that maximize profits. It combines a performance model and an optimization model. The performance model determines future SLA violations for each web service class based on a prediction of future workloads. The optimization model uses these estimates to make the resource allocation and admission control decisions.

Cherkasova and Phaal [30] proposed a Session-Based Admission Control (SBAC) approach that uses the traditional on-off control. It supports four admission control strategies: responsive, stable, hybrid, and predictive. The hybrid strategy tunes itself to be more stable or more responsive based on the observed QoS. Their proposed approach measures server utilizations during predefined time intervals. Using these measured utilizations, it computes predicted utilizations for the next interval. If the predicted utilizations exceed specified thresholds, the admission controller rejects all new sessions in the next time interval and only serves the requests from already admitted sessions. Once the predicted utilizations drop below the given thresholds, the server changes its policy for the next time interval and begins to admit new sessions again.

Chen et al. [29] proposed Admission Control based on Estimation of Service times (ACES). It differentiates and admits requests based on the amount of the processing time required by the individual requests. In ACES, admission of a request is decided by comparing the available computation capacity to the predetermined delay bound of the request. The service time estimation is based on an empirical expression, which is derived from an experimental study on a real web server. Shaaban and Hillston [85] proposed Cost-Based Admission Control (CBAC), which uses a congestion control technique. Rather than rejecting user requests at high load, CBAC uses a discount-charge model to encourage users to postpone their requests to less loaded time periods. However, if a user chooses to proceed with the request in a high load period, an extra charge is imposed. The model is effective for e-commerce web sites when more users place orders that involve monetary transactions.

Muppala and Zhou [76] proposed the Coordinated Session-based Admission Control (CoSAC) approach, which provides SBAC for multi-tier web applications with per-session admission control. CoSAC also provides coordination among the states of tiers with a machine learning technique using a Bayesian network. The admission control mechanism differentiates and admits user sessions based on their type. For example, browsing mix session, ordering mix session, and shopping mix session. However, it remains unclear how it determines the type of a particular session in the first place.

The on-off control in the SBAC approach of Cherkasova and Phaal [30] turns on or off the acceptance of the new sessions for an entire admission control interval. Therefore, the admission control decisions are made only at the interval boundaries and can not be changed within an interval. Thus, a drawback of the on-off control is that it is highly vulnerable to over-admission, especially when handling a bursty load, which may result in the overloading of the servers. To overcome this vulnerability of the on-off control, CoSAC [76] used per-session admission control. Our proposed admission control approach for multi-tier web applications, ACVAS (Section 3.3),

also implements SBAC with per-session admission control [12, 13]. Thus, it makes an admission control decision for each new session. Similarly, our proposed admission control and scheduling approach for video transcoding, SBACS (Section 3.4), implements stream-based admission control with per-stream admission [15].

Huang et al. [58] proposed admission control schemes for proportional differentiated services. It applies to services with different priority classes. The paper proposes two admission control schemes to enable proportional delay differentiated service at the application level. Each scheme is augmented with a prediction mechanism, which predicts the total maximum arrival rate and the maximum waiting time for each priority class based on the arrival rate in the current and last three measurement intervals. When a user request belonging to a specific priority class arrives, the admission control algorithm uses the time series predictor to forecast the average arrival rate of the class for the next interval, computes the average waiting time for the class for the next interval, and determines if the incoming user request is admitted to the server. If admitted, the client is placed at the end of the class queue.

Voigt and Gunningberg [95] proposed admission control based on the expected resource consumption of the requests, including a mechanism for service differentiation that guarantees low response time and high throughput for premium clients. The approach avoids over-utilization of individual server resources, which are protected by dynamically setting the acceptance rate of resource-intensive requests. The adaptation of the acceptance rates (average number of requests per second) is done by using Proportional-Derivative (PD) feedback control loops. Robertsson et al. [82] proposed an admission control mechanism for a web server system with control theoretic methods. It uses a control theoretic model of a G/G/1 system with an admission control mechanism for nonlinear analysis and design of controller parameters for a discrete-time PI-controller. The controller calculates the desired admittance rate based on the reference value of average server utilization and the estimated or measured load situation (in terms of average server utilization). It then rejects those requests that could not be admitted.

All existing admission control approaches discussed above, except CBAC [85], have a common shortcoming in that they rely only on request rejection to avoid server overloading. However, CBAC has its own disadvantages. The discount-charge model of CBAC requires additional web pages to be included in the web application and it is only effective for e-commerce web sites that involve monetary transactions. In contrast, we introduce a simple mechanism to defer user sessions that would otherwise be rejected. In ACVAS, such sessions are deferred on an entertainment server, which sends a wait message to the user and then redirects the user session to an application server as soon as a new server is provisioned or an existing server becomes

less loaded [12]. However, if the entertainment server also approaches its capacity limits, the new session is rejected. Therefore, for each new session request, the admission controller makes one of the three possible decisions: admit the session, defer the session, or reject the session. Likewise, our admission control and scheduling approach for video transcoding, SBACS, provides a stream deferment policy, which exploits the cloud elasticity to allow temporary deferment of the incoming video streams. In other words, the admission controller can decide to admit, defer, or reject an incoming stream and hence reduce the rejection rate [15].

Cherkasova and Phaal [30] defined a simple method for computing the predicted resource utilization, yielding predicted resource utilizations by assigning certain weights to the current and the past utilizations. Muppala and Zhou [76] used the EMA method to make utilization predictions. Huang et al. [58] used machine learning techniques called Support Vector Regression and Particle Swarm Optimization for time-series prediction. Shaaban and Hillston [85] assumed a repeating pattern of workload over a suitable time period. Therefore, in their approach, load in a future period is predicted from the cumulative load of the corresponding previous period. These related works clearly indicate that admission control augmented with prediction models tends to produce better results. Therefore, ACVAS and SBACS also use a prediction model. However, for efficient runtime decision making, it is essential to avoid prediction models which might require intensive computation, frequent updates to their parameters, or (off-line) training. Thus, ACVAS and SBACS use a two-step approach [4, 5], which has been designed to predict future resource loads under soft realtime constraints. The two-step approach consists of a load tracker and a load predictor. We use the EMA method for the load tracker and a simple linear regression model [74] for the load predictor [12].

2.6 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. Shin and Koh [87] presented a hybrid scheme to determine an optimal threshold between the static transcoding (batch processing) and the dynamic transcoding (soft realtime). However, they did not consider the trade-off between the computation cost and the storage cost in a cloud environment. One of the earlier attempts concerning the computation and storage trade-off in cloud computing 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. [36] studied cost and performance trade-offs for an astronomy application using Amazon EC2 and Amazon Simple Storage Service (S3)[4] cost models. They concluded that, based on the likelihood of the 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 [50] was 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. [104] 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 [103, 106] and local-optimization based storage strategy [105]. 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 [103, 106], our proposed trade-off strategy (Section 3.5) estimates an equilibrium point on the time axis where the computation cost and the storage cost of a transcoded video become equal [64, 65]. Moreover, it estimates video popularity of the individual transcoded videos to differentiate popular videos. In our opinion, the DDG-based local-optimization based storage strategy of Yuan et al. [105] is not much relevant for video transcoding because video transcoding does not involve a lot of data dependencies.

Kathpal et al. [69] analyzed compute versus storage trade-off for transcoded videos. They proposed an elimination metric to decide which transcoded videos can be removed from the video repository. However, in contrast to our proposed cost and popularity score based strategy, they did not account for the video popularity score. Moreover, although their results are also based on Amazon EC2 and Amazon S3, they used relatively short videos, which comprise up to 60 second video clips.

Most of the existing computation and storage trade-off strategies described above were originally proposed for scientific datasets. To the best

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

of our knowledge, there are currently only a few computation and storage trade-off strategies for video transcoding, such as Kathpal et al. [69] and our proposed strategy [64, 65]. The difference of the 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.

2.7 Consolidation Approaches and ACO

The existing VM management and consolidation approaches, such as [18,19, 33,40–46,54,61,71,72,77,94,96,100], are used in data centers mainly to minimize under-utilization of PMs and to optimize their power-efficiency. The main idea in these approaches is to use live VM migration [32] to periodically consolidate VMs so that some of the under-utilized PMs could be released for termination. In this thesis, we propose to use a similar technique to cost-efficiently consolidate multiple concurrent third-party web applications in a cloud-based shared hosting environment [16]. Therefore, a key difference in our proposed approach (Section 3.6) is that we consolidate applications on VMs, rather than consolidating VMs on PMs. Thus, our prime concern is to release some of the under-utilized VMs for termination so that the total number of provisioned VMs can be reduced without compromising the overall performance.

Although the web application consolidation problem has certain similarities with the VM consolidation problem, an important difference is that the application consolidation problem is intrinsically more dynamic. This is because, based on the user load, web applications keep on changing their resource demands. On the other hand, the existing VM consolidation approaches tend to assume that the VMs are static in nature [42], that is, they do not change their resource demands. Thus, one of the challenges in the web application consolidation problem is to reduce the computation time of the consolidation algorithm so that the dynamic nature of web applications and their changing resource demands can be accommodated.

Since cost-efficient web application consolidation is a NP-hard combinatorial optimization problem, we apply a highly adaptive online optimization [53] metaheuristic called Ant Colony Optimization (ACO) [37,38] to find a near-optimal solution. ACO is a multi-agent approach to difficult combinatorial optimization problems, such as, Travelling Salesman Problem (TSP) and network routing [37]. It is inspired by the foraging behavior of real ant colonies. While moving from their nest to the food source and back, ants deposit a chemical substance on their path called *pheromone*. Other ants can smell pheromone and they tend to prefer paths with a higher pheromone concentration. Thus, ants behave as agents who use a simple form of in-

direct communication called *stigmergy* to find better paths between their nest and the food source. It has been shown experimentally that this simple pheromone trail following behavior of ants can give rise to the emergence of the shortest paths [37]. It is important to note here that although each ant is capable of finding a complete solution, high quality solutions emerge only from the global cooperation among the members of the colony who concurrently build different solutions. Moreover, to find a high quality solution, it is imperative to avoid *stagnation*, which is a premature convergence to a suboptimal solution or a situation where all ants end up finding the same solution without sufficient exploration of the search space [37]. In ACO metaheuristic, stagnation is avoided mainly by using pheromone evaporation and stochastic state transitions. There are a number of ant algorithms, such as, Ant System (AS), Max-Min Ant System (MMAS), and Ant Colony System (ACS) [37, 38]. ACS [37] improves the performance of AS and is currently one of the best performing ant algorithms. Therefore, we apply ACS to the web application consolidation problem.

One of the earlier works on applying ACO to the general resource allocation problem include [102]. They applied ACO to the nonlinear resource allocation problem, which seeks to find an optimal allocation of a limited amount of resources to a number of tasks to optimize their nonlinear objective function. Chaharsooghi and Kermani [27] proposed a modified version of ACO for multi-objective resource allocation problem. A more recent work by Feller et al. [42] and Feller et al. [43] applied MMAS to the VM consolidation problem in the context of cloud computing. However, to the best of our knowledge, currently there are no existing works on using ACO metaheuristic to consolidate multiple web applications in a cloud-based shared hosting environment.

Chapter 3

Contributions of the Thesis

In this thesis, we address the problem of cost-efficient VM provisioning augmented with server consolidation and admission control on the provisioned VMs. We seek solutions for two types of applications: multi-tier web applications that follow the request-response paradigm and on-demand video transcoding that is based on video streams with soft realtime constraints. Although there are many similarities between VM provisioning for web applications and VM provisioning for video transcoding, each one of them also has its own challenges [9].

We present a cost-efficient VM provisioning approach for multi-tier web applications [2, 8, 10, 11, 13, 23] and on-demand video transcoding [63, 66]. Moreover, to prevent virtualized servers from becoming overloaded, the proposed VM provisioning approach is augmented with an admission control mechanism [12, 15]. Furthermore, to minimize the under-utilization of the virtualized application servers, we provide a web application consolidation approach [16]. We also present a computation and storage trade-off strategy for cost-efficient video transcoding in cloud computing [64, 65]. These contributions are based on the following set of general assumptions:

- We assume homogenous VMs. For example, small (*m1.small*) instances from the Amazon EC2 cloud.
- Our proposed algorithms are designed to work with on-demand instances from an IaaS cloud. For instance, Amazon EC2 on-demand instances.
- Since some contemporary IaaS clouds, such as Amazon EC2, charge on hourly basis, it is assumed that the VM costing interval is one hour.

Our thesis contributions are presented in detail in the original publications in Part II of the thesis. This chapter presents a summary of our main contributions while providing a brief overview of some of the most important challenges that they address.

3.1 VM Provisioning for Multi-tier Web Applications

Our first contribution is a cost-efficient VM provisioning approach for multiple multi-tier web applications. The proposed approach comprises two sub-approaches: a reactive VM provisioning approach called ARVUE [2,10] and a hybrid reactive-proactive VM provisioning approach called CRAMP [11,13]. The proposed approach provides automatic deployment and scaling of multiple simultaneous web applications on a given IaaS cloud in a shared hosting environment. It monitors and uses resource utilization metrics and does not require a performance model of the applications or the infrastructure dynamics. The shared hosting environment allows us to share VM resources among deployed applications, reducing the total number of required VMs. Performance under varying load conditions is guaranteed by automatic adjustment and tuning of the CRAMP parameters.

The main task of the proposed approach is to provision and remove VMs for the application server tier and to deploy and remove applications from each VM, in order to maintain a desired QoS. For cost-effectiveness, the approach supports deployment of multiple simultaneous applications on a single VM. At any given time, an application may be deployed in zero, one or more VMs. Popular applications are often deployed in many VMs, while sporadically used applications may not be deployed at all in order to save resources. Moreover, due to memory limitations, it is also not possible to assume that we can deploy all applications in one VM.

Figure 3.1 depicts the system architecture of our proposed VM provisioning, admission control, and consolidation approaches for multi-tier web applications. It consists of the following components: *global controller*, *admission controller*, *application server*, *local controller*, *load predictor*, *application repository*, *cloud provisioner*, *HTTP load balancer*, and *entertainment server*. The global controller implements VM provisioning and web application consolidation (Section 3.6) algorithms along with the session-to-server allocation and application-to-server allocation policies [10]. The admission controller and the entertainment server are part of our admission control approach for multi-tier web applications called ACVAS (Section 3.3).

An application server instance runs on a dynamically provisioned VM. Each application server runs multiple web applications in an Open Services Gateway initiative (OSGi) [90] environment. In such an environment, each application runs as an OSGi component called a bundle. OSGi also introduces the dynamic component model, which allows dynamic loading and unloading of bundles. In addition to the web applications, each application server also runs a local controller and a load predictor. The local controller monitors and logs server resource utilizations. It also controls the OSGi

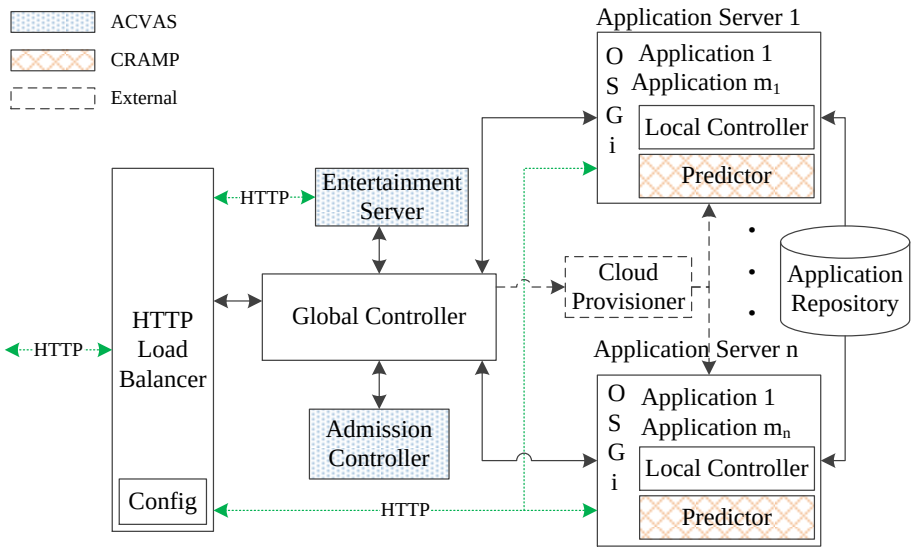

Figure 3.1: System architecture of the proposed VM provisioning, admission control, and consolidation approaches for multi-tier web applications

environment for loading and unloading of web applications. The load predictor predicts future load on an application server. It uses current and past resource utilization data of the server to predict a few steps ahead in the future. Our load prediction approach consists of a load tracker and a load predictor [5]. It uses EMA for the load tracker and a simple linear regression model [74] for the load predictor [11].

Web applications are stored in an application repository, from where they are loaded onto application servers. The cloud provisioner refers to the cloud provisioner in an external IaaS cloud, such as the provisioner in Amazon EC2. While the VM provisioning and termination decisions are made by the global controller, the actual lower level tasks of starting and terminating VMs are done by the cloud provisioner. The HTTP requests are routed through a high performance HTTP load balancer and proxy. For this, we use HAProxy[1], which balances the load of requests for new user sessions among the application servers. For its functions, HAProxy maintains a configuration file containing information about application servers and application deployments on each server. As a result of the VM provisioning and termination operations, the configuration file is frequently updated with new information.

The most important characteristics of ARVUE are that it is based on reactive feedback control [55], it does not depend upon a performance model of the application or the infrastructure dynamics, it deploys multiple appli-

[1]http://haproxy.1wt.eu/

cations per VM, and it provides server level and application level scaling. CRAMP is similar to ARVUE except that it is based on a hybrid reactive-proactive control.

3.1.1 VM Provisioning Delay

In practice, it currently takes a few minutes to provision a VM from an IaaS cloud [10, 11, 34]. Due to this inevitable VM provisioning delay, the handling of a sudden spike in the incoming user load becomes a challenge. Some of the strategies that we use to overcome this drawback of IaaS clouds include provisioning multiple VMs at a time [10, 11], using additional VM capacity [10, 11], and using load prediction to provision preemptively [11].

3.1.2 Hybrid Reactive-Proactive VM Provisioning

Many traditional VM provisioning approaches, such as [10, 31, 57, 62, 99], use reactive provisioning. However, the primary shortcoming of reactive provisioning is that it starts a VM provisioning operation only after a significant increase in the load is detected [81]. Therefore, the new VMs can only be used instantly if the VM provisioning was instantaneous [11]. However, due to the VM provisioning delay, the reactive approach may fail to handle increased load, especially under sudden load spikes. Alternatively, some approaches use a prediction of the future load to provision preemptively [6, 81]. In CRAMP [11], we provide a hybrid reactive-proactive approach for web applications, which assigns certain weights to the reactive and the proactive provisioning. Nevertheless, the main challenge in prediction-based approaches is in making predictions with high prediction accuracy under soft realtime constraints [11]. Therefore, we use a two-step load prediction method [4, 5] with EMA and a simple linear regression model [74], which predicts a few steps ahead in the future with high prediction accuracy under soft realtime constraints [12].

3.1.3 Reduced Oscillations in Number of VMs

Another important challenge is to reduce oscillations in the number of provisioned VMs. This is desirable because oscillations in the presence of the inevitable VM provisioning delay may lead to a deteriorated performance [10]. Moreover, since some IaaS clouds, such as Amazon EC2, currently charge on an hourly basis, oscillations in the number of provisioned VMs may result in a higher provisioning cost [66]. Therefore, we use a few strategies to counteract oscillations in the number of VMs, such as, delaying new provisioning operations until previous provisioning operations have been realized [10, 62] and terminating only those VMs that were constantly under-utilized for a longer period of time [10].

3.1.4 Sharing of VM Resources for Improved Utilization

For cost-efficient VM provisioning to deploy and scale multiple web applications, the proposed approach should provide a finer deployment granularity than the smallest VM provided by the contemporary IaaS clouds [11]. This is especially important when deploying a large number of web applications, most of which may have very few users, while a few of them may have many users. Therefore, we use shared hosting, which deploys one or more web applications on each VM [10, 11]. Moreover, popular applications are often deployed in many VMs, while sporadically used applications may not be deployed at all in order to save resources [10]. Thus, instead of provisioning at least one full VM per application, shared hosting effectively supports provisioning a fraction of a VM per application, resulting in a reduced number of total VMs. The deployment of multiple web application in a shared hosting environment enables two levels of scaling, namely server-level scaling and application-level scaling [10]. The server-level scaling provisions and terminates VMs from an IaaS cloud to create a dynamically scalable cluster of servers, whilst the application-level scaling deploys and removes web applications from each virtualized server.

3.2 VM Provisioning for Video Transcoding

Our second contribution in this thesis is a VM provisioning approach for video transcoding in the cloud [63, 66]. The proposed VM provisioning approach uses load prediction to proactively scale video transcoding service on a given IaaS cloud. It provides mechanisms for allocation and deallocation of VMs to a cluster of video transcoding servers in a horizontal fashion. For cost-efficiency, our work supports concurrently transcoding multiple on-demand video streams on a single VM, resulting in a reduced number of required VMs.

Figure 3.2 presents the system architecture of our proposed VM provisioning and admission control approaches for video transcoding. It consists of a *streaming server*, a *video splitter*, a *video merger*, a *video repository*, a dynamically scalable cluster of video *transcoding servers*, a *load balancer*, a *master controller*, an *admission controller*, an *entertainment server*, and a *load predictor*. The admission controller and the entertainment server are part of our admission control and scheduling approach for video transcoding called SBACS (Section 3.4). The video requests and responses are routed through the streaming server. Since our main focus 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 a job queue. The video splitting or segmentation is performed at the GOP level, where GOPs represent atomic units that can be transcoded independently of one another [66]. The load balancer employs a task assignment policy, which distributes the load on the transcoding servers. In other words, it decides when and to which transcoding server a transcoding job should be sent. The actual transcoding is performed by the transcoding servers. A transcoding server runs on a dynamically provisioned VM. It gets video segments, performs the required transcoding operations, and returns the transcoded video segments for merging.

The master controller acts as the main controller and the resource allocator. It implements prediction-based dynamic VM allocation and deallocation algorithms and one or more computation and storage trade-off strategies. For load prediction, the master controller uses the load predictor, which predicts future load on the transcoding servers. The video merger merges the transcoded jobs into video streams, which form video responses.

3.2.1 Video Segmentation at GOP Level

As described in Section 3.1.4, improved utilization and sharing of the VM resources are essential ingredients of a cost-efficient VM provisioning approach. Therefore, for video transcoding, we use video segmentation at the GOP level, which splits video streams into smaller segments that can be transcoded independently of one another [66]. It allows transcoding of multiple video streams concurrently on a single VM. The sharing of the VM resources among multiple concurrent video streams improves VM utilization, which helps in reducing the total number of required VMs.

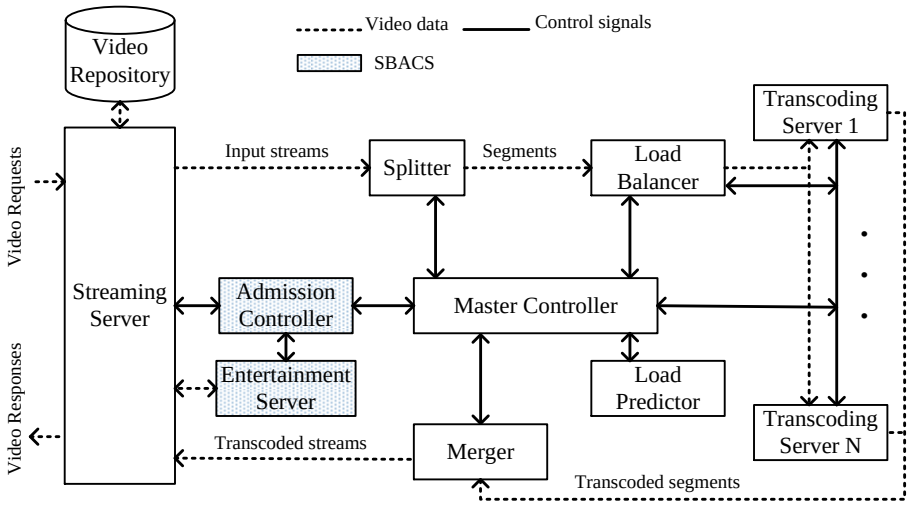

Figure 3.2: System architecture of the proposed VM provisioning and admission control approaches for video transcoding

3.2.2 Proactive VM Provisioning

Our VM provisioning approach for video transcoding is a proactive approach [63, 66]. It uses a two-step load prediction method [4, 5] with EMA and a simple linear regression model [74], which allows proactive VM provisioning with high prediction accuracy under soft realtime constraints.

3.2.3 Reduced Oscillations and Conservative Termination

As explained in Section 3.1.3, it is an important challenge to reduce oscillations in the number of provisioned VMs. Therefore, for video transcoding, we use two different strategies to counteract oscillations in the number of VMs. The first strategy delays new provisioning operations until previous provisioning operations complete and their effect becomes visible in the system [62, 66]. Our second strategy calculates the remaining time of each transcoding server with respect to the completion of the renting period. For instance, in the case of Amazon EC2, the renting period is one hour. Thus, our proposed approach terminates a VM only when the VM renting period approaches its completion and all jobs on the server complete their execution [63, 66]. Therefore, if a VM is under-utilized, but its renting period is not close to completion, it will not be terminated.

3.3 Admission Control for Multi-tier Web Applications

Admission control is often implemented at the server level. A server with admission control in place would stop accepting new user requests or sessions when the server approaches its capacity limits. Therefore, overload prevention relies on rejection of new requests or sessions. Most traditional admission control approaches implement request-based admission control. However, some recent approaches, such as [30], use SBAC, which often yields better results for stateful web applications.

Our third contribution in this thesis is an admission control approach for multi-tier web applications called ACVAS. It provides SBAC for a dynamically scalable application server tier. Instead of relying only on the rejection of new user sessions, ACVAS implements a simple mechanism that defers such sessions and then serves them as soon as possible in the near future. It implements per-session admission control and uses monitored resource utilizations to predict a few steps ahead in the future. Then, based on the measured and predicted utilizations, it computes weighted utilizations. The weighted utilizations of individual server resources are used to make an admission control decision for each new session. Performance under varying load conditions is guaranteed by automatic adjustment and tuning of the admission control mechanism.

3.3.1 SBAC with Per-Session Admission Control

The SBAC approach in [30] implements on-off control, where acceptance of new sessions is turned on or off for the entire admission control interval. Thus, the admission control decisions are made at the interval boundaries, which can not be changed inside an interval. A drawback of the on-off control is that it may lead to over-admission, especially when handling a bursty load, which can result in the overloading of the servers. To overcome this drawback of the on-off control, CoSAC [76] proposed per-session admission control. ACVAS also implements SBAC with per-session admission control. Thus, it makes an admission control decision for each new session.

3.3.2 Session Deferment Mechanism

All existing admission control approaches discussed in Section 2.5, except CBAC [85], have a common drawback that they rely only on request rejection to avoid server overloading. However, the discount-charge model in CBAC requires additional web pages to be inserted into the web application and it is only effective for e-commerce web sites that involve monetary transactions. Therefore, in ACVAS, we introduce a simple mechanism to

defer user sessions that would otherwise be rejected. Such sessions are deferred on an entertainment server, which sends a wait message to the user and then redirects the user session to an application server as soon as a new server is provisioned or an existing server becomes less loaded. However, if the entertainment server also approaches its capacity limits, the new session is rejected. Thus, for each new session request, the admission controller in ACVAS makes one of the three possible decisions: admit the session, defer the session, or reject the session.

3.3.3 Automatic Adjustment and Tuning for Better QoS

Schroeder et al. [84] considered automatic adjustment and tuning of the admission control mechanism to be the most difficult part. The SBAC approach in [30] used a hybrid policy for automatic adjustment and tuning of the admission control mechanism. It tries to achieve better QoS and higher session throughput by using a parameter called admission control *weight*, which gives more or less weight to the measured and the predicted utilizations. In their approach, the weight parameter is adjusted and tuned based on the number of aborted sessions and the number of refused connections [30].

For automatic adjustment and tuning of the admission control mechanism, ACVAS uses a similar approach as in [30]. However, it adjusts and tunes the weight parameter based on the following metrics: number of aborted sessions, number of deferred sessions, number of rejected sessions, and number of overloaded servers [12].

3.4 Admission Control and Scheduling for Video Transcoding

As our fourth contribution in this thesis, we present an admission control and scheduling approach for a dynamically scalable cluster of video transcoding servers called SBACS [15]. It provides video stream based admission control on a per-stream level. SBACS uses queue waiting time of transcoding servers to make admission control decisions. For preemptive control, it uses a two-step load prediction approach [4, 5], which predicts queue waiting times on individual servers. In addition to the traditional load rejection policy, SBACS also provides a stream deferment policy, which allows temporary deferment of an arrived stream until a new server is provisioned or an existing server becomes less loaded. Moreover, to ensure efficient utilization of the transcoding servers, server resources are shared among admitted streams by performing video segmentation at the GOP level. The video segments are then sent to the servers via a load balancer. SBACS also provides a

job scheduling algorithm, which aims to prevent transcoding jitters in the admitted streams by possibly dropping a small proportion of video frames to ensure continued delivery of the video contents to the users. Therefore, the main tasks of the proposed SBACS approach are to make admission control and job scheduling decisions for a scalable tier of transcoding servers, in which each server runs on a VM.

3.4.1 Stream-Based Admission Control with Per-Stream Admission

Some of the recent admission control approaches for web applications, such as [12, 76], use SBAC with per-session admission. We present a similar approach for video streams. SBACS provides stream-based admission control with per-stream admission, which reduces the risk of over-admission by making an admission control decision for each incoming video stream.

3.4.2 Stream Deferment Mechanism

Most of the traditional admission control approaches rely only on the load rejection policy to prevent server overloading. Therefore, as described in Section 3.3.2, we introduced a simple session deferment mechanism in ACVAS [12]. SBACS presents a similar deferment mechanism for video streams, which would otherwise be rejected. Such streams are deferred on an entertainment server until a new server is provisioned or an existing server becomes less loaded. However, if the entertainment server also approaches its capacity limits, the new streams are rejected. Thus, for each new video stream, the admission controller in SBACS makes one of the three possible decisions: admit the stream, defer the stream, or reject the stream.

3.4.3 Job Scheduling Based on Queue Waiting Time

In addition to an admission control mechanism, SBACS also features a job scheduling algorithm. The algorithm uses queue waiting time of individual transcoding servers to complement admission control and to prevent transcoding jitters in the admitted video streams. For overload prevention on a sufficiently utilized server, it starts dropping a small proportion of video frames from each subsequent transcoding segment on the server. Likewise, for preventing jitters in a video stream, it computes the estimated delivery deadline, the estimated transcoding time, and the estimated response time of each video segment. If it finds a deadline violation, the violation is prevented by dropping some video frames in proportion to the degree of violation.

3.5 Computation and Storage Trade-off Strategy

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. IaaS clouds, such as Amazon EC2, currently provide VMs for creating a dynamically scalable cluster of servers. Likewise, a cloud storage service, such as Amazon S3, may be used to store a large number of transcoded videos. However, the exact number of VMs and the exact amount of storage needed at a specific time depend upon the incoming load from service users and their performance requirements. Moreover, it may be possible to reduce the total IaaS cost by trading storage for computation, or vice versa. Therefore, finding a cost-efficient computation and storage trade-off strategy for video transcoding in cloud computing is an important problem. In this thesis, we 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 [64, 65]. The objective is to reduce the total IaaS cost of the dynamically scalable on-demand video transcoding service by trading storage for computation, or vice versa. 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.

In an on-demand video transcoding 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 strategy is based only on the computation and storage costs of the transcoded videos.

3.5.1 Computation and Storage Costs

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

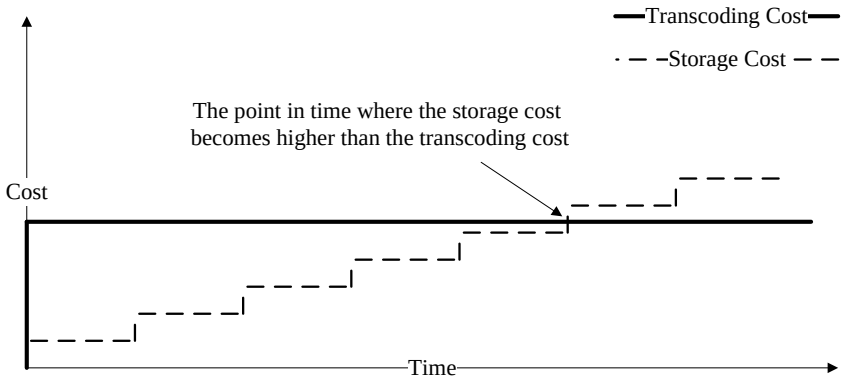

Figure 3.3: The estimated equilibrium point between the storage cost and the transcoding cost of a transcoded video

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 3.3 shows that if a 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.

3.5.2 Video Cost and Popularity Score

In an on-demand video streaming service, each transcoded video may be requested and viewed a number of times. Frequently viewed, popular videos receive 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 for each

transcoded video. 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.

3.6 Web Application Consolidation using ACO

The under-utilization of VMs becomes more pertinent when a SaaS or a PaaS provider wants to leverage an IaaS cloud to cost-efficiently deploy a large number of web applications of varying resource needs. The solution to this problem is to create a dynamically scalable application server tier that manages multiple applications simultaneously, while using shared hosting [91] to deploy multiple applications on a VM [10,11]. A similar fine-grained resource sharing approach was used in Mesos [56]. However, it provides a platform for multiple cluster computing frameworks rather than web applications. Moreover, dynamic scaling [10, 11] alone does not guarantee cost-efficient deployment. Figure 3.4 presents a simple hypothetical scenario to motivate the need to consolidate multiple web applications in a cloud-based shared hosting environment. Each application server in Figure 3.4 runs multiple web applications. Moreover, some applications run on multiple servers. It is assumed that due to some significant load variations, application server 2 and application server 3 have become under-utilized. The under-utilized servers in such a scenario may continue to remain under-utilized for several hours, days, or even weeks unless there is a significant increase in their load or some new web applications are deployed on them. Thus, it is difficult to provide cost-efficient deployment and scaling of multiple web applications in a cloud-based shared hosting environment without consolidation of web applications on under-utilized VMs.

As our sixth contribution in this thesis, we present a novel approach to consolidate multiple web applications in a cloud-based shared hosting environment [16]. We propose an application consolidation algorithm that uses a metaheuristic [20, 53] called ACO [37, 38] to build a web application migration plan, which is then used to minimize over-provisioning of VMs by consolidating web applications on under-utilized VMs.

3.6.1 Shared Hosting of Web Applications

In our cloud-based shared hosting approach, each virtualized application server runs multiple Java Servlet-based web applications in the same Java Virtual Machine (JVM) [2, 10, 11]. However, since Java lacks some important features needed to safely run multiple third-party web applications in

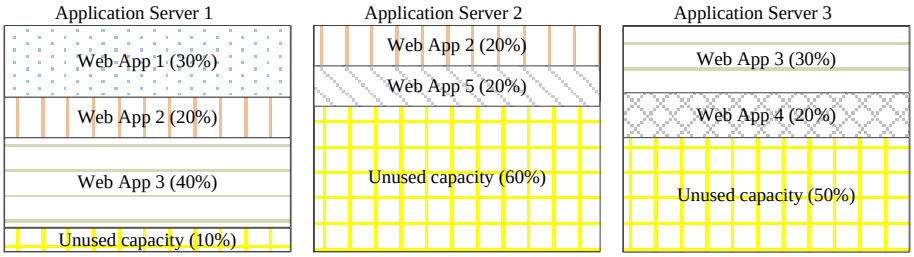

Figure 3.4: A simple example to motivate the need to consolidate multiple web applications in a cloud-based shared hosting environment

one JVM, Aho et al. [2] extended and used a widely adapted OSGi specification [90], which partly addresses this problem. Therefore, in this way, shared hosting enables safe deployment of multiple concurrent third-party web applications on each virtualized application server. Thus, in our approach, each application server runs multiple web applications in an OSGi [90] environment, where each application runs as an OSGi component called a bundle. OSGi also introduces the dynamic component model, which allows dynamic loading and unloading of bundles. In addition to the web applications, each application server also runs a local controller, as shown in Figure 3.1. The local controller controls the OSGi environment for loading and unloading of web applications. Web applications are stored in an application repository, from where they are loaded onto application servers.

3.6.2 Objective Function

The output of the application consolidation algorithm is a migration plan, which, when enforced, would result in a reduced set of VMs needed to host all web applications without compromising their performance. Thus, the objective function for the proposed algorithm is defined in terms of number of released VMs [16]. Moreover, it prefers larger migration plans because with a large set of web applications in a shared hosting environment, each VM typically hosts a number of applications, which makes it less likely to find a feasible solution with a smaller migration plan. Later on, when a migration plan is enforced, we apply a constraint which reduces the number of actual migrations by restricting migrations to only those VMs that are not included in the set of released VMs.

3.6.3 Stochastic State Transition Rule

In our proposed cloud-based shared hosting environment [16], each virtualized application server hosts one or more web applications from the set

of applications. Moreover, since frequently used applications are often concurrently run on multiple VMs, if a particular application is concurrently hosted by two VMs, then each of its deployments is considered a different *application instance*. An application instance not only contains a web application, but also the user sessions that belong to it. Furthermore, for the sake of application migration, each VM is a potential *source VM*. Both the source VM and the application instance are characterized by their resource utilizations, such as CPU load average and memory utilization. Likewise, an application instance can be migrated to any other VM. Therefore, every other VM is a potential *destination VM*, which is also characterized by its resource utilizations. Thus, the proposed ACO-based algorithm makes a set of tuples, where each tuple consists of three elements: source VM, application instance, and destination VM.

Unlike the TSP, there is no notion of a path in the application consolidation problem. Therefore, ants deposit pheromone on the tuples. Each ant uses a stochastic state transition rule to choose the next tuple to traverse. The state transition rule in ACS is called the pseudo-random-proportional-rule [38]. It prefers tuples with a higher pheromone concentration and which result in a higher number of released VMs [16].

3.6.4 Global and Local Pheromone Trail Evaporation Rules

In addition to the stochastic state transition rule, ACS also uses a global and a local pheromone trail evaporation rule. The global pheromone trail evaporation rule is applied towards the end of an iteration after all ants complete their migration plans. On the other hand, the local pheromone trail update rule is applied on a tuple when an ant traverses the tuple while making its migration plan.

The pseudo-random-proportional-rule in ACS and the global pheromone trail update rule are intended to make the search more directed. Therefore, the ants try to search other high quality solutions in a close proximity of the thus far global best solution. On the other hand, the local pheromone trail update rule complements exploration of other high quality solutions that may exist far from the thus far global best solution. This is because whenever an ant traverses a tuple and applies the local pheromone trail update rule, the tuple looses some of its pheromone and thus becomes less attractive for other ants. Therefore, it helps in avoiding stagnation where all ants end up finding the same solution or where they prematurely converge to a suboptimal solution.

Chapter 4

Description of Papers

This chapter presents a summary of the original publications presented in Part II of this thesis along with a description of the author's contribution in each publication. It also provides a mapping between the RQs posed in Section 1.1 and the individual publications in Part II. Finally, it presents a discussion on how the original publications relate to one another.

4.1 Overview of Original Publications

This thesis is a collection of 8 original publications, which are referred to in the text by their roman numerals. In this section, we present a summary of the individual publications while highlighting the author's contribution in each publication.

4.1.1 Paper I: Feedback Control Algorithms to Deploy and Scale Multiple Web Applications per Virtual Machine

Paper I presents our reactive VM provisioning approach for multi-tier web applications called ARVUE. It provides automatic deployment and scaling of multiple simultaneous web applications on a given IaaS cloud in a shared hosting [91] environment. The main task of the proposed approach is to provision and remove VMs for the application server tier and to deploy and remove applications from each VM, in order to maintain a desired QoS in a cost-efficient manner. Therefore, our main contribution in this paper is feedback control algorithms for scaling up and scaling down of the application server tier. It also provides algorithms for scaling individual web applications up and down. The results presented in this paper are based on our prototype implementation. However, since the objective of the experimental evaluation was to provide only a proof-of-concept, the paper did not provide a comparison of the results with other approaches.

Author's contribution: The main idea presented in this paper was developed by the author in a close collaboration with coauthors Benjamin Byholm and Joonas Lehtinen under the guidance of Professor Ivan Porres. Adnan Ashraf is the main author of this paper. The paper also incorporates some ideas of coauthor Joonas Lehtinen, such as, the idea of deploying multiple Java Servlet-based web applications in the same JVM. The ARVUE prototype was jointly developed by Niclas Snellman, Thomas Fors, and Benjamin Byholm at Åbo Akademi University and Marc Englund at Vaadin Ltd.

4.1.2 Paper II: A Session-Based Adaptive Admission Control Approach for Virtualized Application Servers

Paper II presents our session-based adaptive admission control approach for virtualized application servers called ACVAS. Instead of using the traditional on-off control, ACVAS implements per-session admission, which reduces the risk of over-admission. Moreover, instead of relying only on rejection of new sessions, it takes benefit of the cloud elasticity to implement a simple session deferment mechanism that reduces the number of rejected sessions while increasing session throughput. Therefore, our main contribution in this paper is an admission control algorithm to prevent virtualized application servers from becoming overloaded. Moreover, we extended the two-step load prediction method proposed by Andreolini et al. [5] to use a simple linear regression model [74] for load prediction. The results presented in this paper are based on our discrete-event simulations. We also provide a comparison of the results with an existing session-based adaptive admission control approach [30], which uses on-off control and does not provide a session deferment mechanism.

Author's contribution: The main idea presented in this paper was developed by the author under the guidance of Professor Ivan Porres. Adnan Ashraf is the main author of this paper. The discrete-event simulations were also developed by Adnan Ashraf. Coauthor Benjamin Byholm derived the traces from the IRCache[1] project access logs and analyzed them to obtain the realistic load pattern used in the second experiment.

4.1.3 Paper III: CRAMP: Cost-Efficient Resource Allocation for Multiple Web Applications with Proactive Scaling

Paper III presents our hybrid reactive-proactive VM provisioning approach called CRAMP. CRAMP is similar to ARVUE except that it is based on a hybrid reactive-proactive control. Therefore, this paper extends the scaling up and scaling down algorithms of Paper I with the extended two-step load

[1] http://www.ircache.net/

prediction method of Paper II. The results presented in this paper are based on our prototype implementation. We also provide a comparison of the results with our reactive VM provisioning approach in Paper I.

Author's contribution: The main idea presented in this paper was developed by the author under the guidance of Professor Ivan Porres. Adnan Ashraf is the main author of this paper. The paper also incorporates some ideas of coauthor Benjamin Byholm, such as, the idea of using NRMSE to compute weighted load average. The CRAMP prototype was developed by Benjamin Byholm.

4.1.4 Paper IV: Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing

Paper IV presents our VM provisioning approach for video transcoding in the cloud. It provides mechanisms for allocation and deallocation of VMs to a cluster of video transcoding servers in a horizontal fashion. Therefore, our main contribution in this paper are prediction-based dynamic VM allocation and deallocation algorithms for video transcoding. The proposed algorithms use our extended two-step load prediction method of Paper II to predict the total transcoding rate of all transcoding servers. The results presented in this paper are based on our discrete-event simulations. However, since the objective of the experimental evaluation was to provide only a proof-of-concept, the paper did not provide a comparison of the results with other approaches.

Author's contribution: The main idea presented in this paper was developed jointly by coauthors Fareed Ahmed Jokhio and Adnan Ashraf. The discrete-event simulations were developed by Adnan Ashraf. The paper was written jointly by coauthors Fareed Ahmed Jokhio and Adnan Ashraf under the guidance of Dr. Sébastien Lafond, Professor Ivan Porres, and Professor Johan Lilius.

4.1.5 Paper V: Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing

Paper V presents our admission control and scheduling approach for video transcoding called SBACS. SBACS implements stream-based admission control with per-stream admission. It uses queue waiting time of transcoding servers to make admission control decisions for incoming video streams. In addition to the traditional rejection policy, SBACS also provides a stream deferment policy, which exploits cloud elasticity to allow temporary deferment of the incoming video streams. In order to prevent transcoding jitters in the admitted streams and to ensure continued delivery of the video contents to the user, we introduce a job scheduling mechanism, which uses

temporal resolution reduction. Therefore, our main contributions in this paper are admission control and job scheduling algorithms for video transcoding in the cloud. The results presented in this paper are based on our discrete-event simulations. However, since the objective of the experimental evaluation was to provide only a proof-of-concept, the paper did not provide a comparison of the results with other approaches.

Author's contribution: The main idea presented in this paper was developed jointly by coauthors Adnan Ashraf and Fareed Ahmed Jokhio. Adnan Ashraf is the main author of this paper. The discrete-event simulations were also developed by Adnan Ashraf. The paper was written jointly by coauthors Adnan Ashraf and Fareed Ahmed Jokhio under the guidance of Dr. Sébastien Lafond, Professor Ivan Porres, and Professor Johan Lilius.

4.1.6 Paper VI: A Computation and Storage Trade-Off Strategy for Cost-Efficient Video Transcoding in the Cloud

Paper VI investigates the computation and storage cost trade-off for video transcoding in the cloud and presents a cost-efficient strategy 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 results presented in this paper are based on our discrete-event simulations. We also provide a comparison of the results with two intuitive computation and storage trade-off strategies called store all strategy and usage based strategy [104].

Author's contribution: The main idea presented in this paper was developed jointly by coauthors Fareed Ahmed Jokhio, Adnan Ashraf, and Sébastien Lafond. The paper was written jointly by coauthors Fareed Ahmed Jokhio and Adnan Ashraf under the guidance of Dr. Sébastien Lafond, Professor Ivan Porres, and Professor Johan Lilius.

4.1.7 Paper VII: Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing

Paper VII extends the works presented in Paper IV and Paper VI and provides an extended evaluation. It improves the VM provisioning algorithm of Paper IV by taking into account the average queue length of the transcoding servers. Similarly, it extends the cost and popularity score based strategy of Paper VI by providing an algorithm to calculate the cost and popularity score and an algorithm to decrement the score of an unpopular video and finally remove the video from the video repository. The results presented in

this paper are based on our discrete-event simulations. We also provide a comparison of the results with two intuitive computation and storage trade-off strategies called store all strategy and usage based strategy [104].

Author's contribution: The main idea presented in this paper was developed jointly by coauthors Fareed Ahmed Jokhio and Adnan Ashraf. The paper was also written jointly by coauthors Fareed Ahmed Jokhio and Adnan Ashraf under the guidance of Dr. Sébastien Lafond, Professor Ivan Porres, and Professor Johan Lilius.

4.1.8 Paper VIII: Using Ant Colony System to Consolidate Multiple Web Applications in a Cloud Environment

Paper VIII presents a novel approach to consolidate multiple web applications in a cloud-based shared hosting environment. It uses a metaheuristic [20,53] approach called ACO [37,38] to build a web application migration plan, which is then used to minimize over-provisioning of VMs by consolidating web applications on under-utilized VMs. Therefore, our main contribution in this paper is a web application consolidation algorithm to minimize over-provisioning of VMs in a cloud-based shared hosting environment. The results presented in this paper are based on our discrete-event simulations. We also provide a comparison of the results with a baseline, greedy application consolidation approach, which is based on an extension of our previous works in Paper I and Paper III.

Author's contribution: The main idea presented in this paper was developed by the author under the guidance of Professor Ivan Porres. Adnan Ashraf is the main author of this paper. The discrete-event simulations were also developed by Adnan Ashraf. Benjamin Byholm at Åbo Akademi University derived the traces from the IRCache project access logs and analyzed them to obtain the realistic load pattern used in the second experiment.

4.2 Discussion

Table 4.1 presents a mapping between the RQs posed in Section 1.1 and the original publications in Part II of this thesis. RQ1 concerns the problem of ensuring scalability of multi-tier web applications and on-demand video transcoding service for different types of load conditions while providing a good trade-off between performance and cost. This RQ is addressed in Paper I, Paper III, Paper IV, and Paper VII. Paper I provides a reactive dynamic scaling approach for multi-tier web applications. Paper III improves the work presented in Paper I and presents a hybrid reactive-proactive dynamic scaling approach for multi-tier web applications. Similarly, Paper IV provides a prediction-based dynamic scaling approach for video transcoding

Table 4.1: Mapping between RQs and original publications

RQs	**Publications**
RQ1	Paper I, Paper III, Paper IV, and Paper VII
RQ2	Paper II and Paper V
RQ3	Paper VI and Paper VII
RQ4	Paper VIII

service. Paper VII extends the work presented in Paper IV by taking into account the average queue length of the transcoding servers.

RQ2 pertains to the problem of cost-efficiently preventing servers from becoming overloaded. This RQ is addressed in Paper II and Paper V. Paper II presents a session-based adaptive admission control approach for multi-tier web applications. Similarly, Paper V provides a stream-based admission control and scheduling approach for video transcoding service.

RQ3 seeks a strategy to provide a good-tradeoff between the computation cost and the storage cost when using a public IaaS cloud for on-demand video transcoding. This RQ is addressed in Paper VI and Paper VII. Paper VI presents a computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. Paper VII extends the work presented in Paper VI and provides an extended evaluation. It also provides an algorithm to calculate the cost and popularity score and an algorithm to decrement the score of an unpopular video and finally remove the video from the video repository.

RQ4 concerns the problem of dynamic consolidation of multi-tier web applications on under-utilized VMs to reduce under-utilization of the virtualized application servers in a cloud-based shared hosting environment. This RQ is addressed in Paper VIII. It presents a novel approach to consolidate multiple web applications on under-utilized VMs.

Figure 4.1 illustrates the relationship among the original publications presented in Part II of this thesis. Paper I provides our reactive VM provisioning approach for multi-tier web applications called ARVUE. Paper II augments ARVUE with a session-based adaptive admission control approach for multi-tier web applications called ACVAS. It also extends the two-step load prediction method proposed by Andreolini et al. [5] to use a simple linear regression model [74] for load prediction. Paper III extends ARVUE with the extended two-step load prediction method of Paper II and presents a hybrid reactive-proactive VM provisioning approach called CRAMP. Paper IV provides our prediction-based VM provisioning approach for video transcoding in the cloud. Paper V presents a stream-based admission control and scheduling approach for video transcoding called SBACS. It provides an admission control algorithm for video transcoding servers and a job sched-

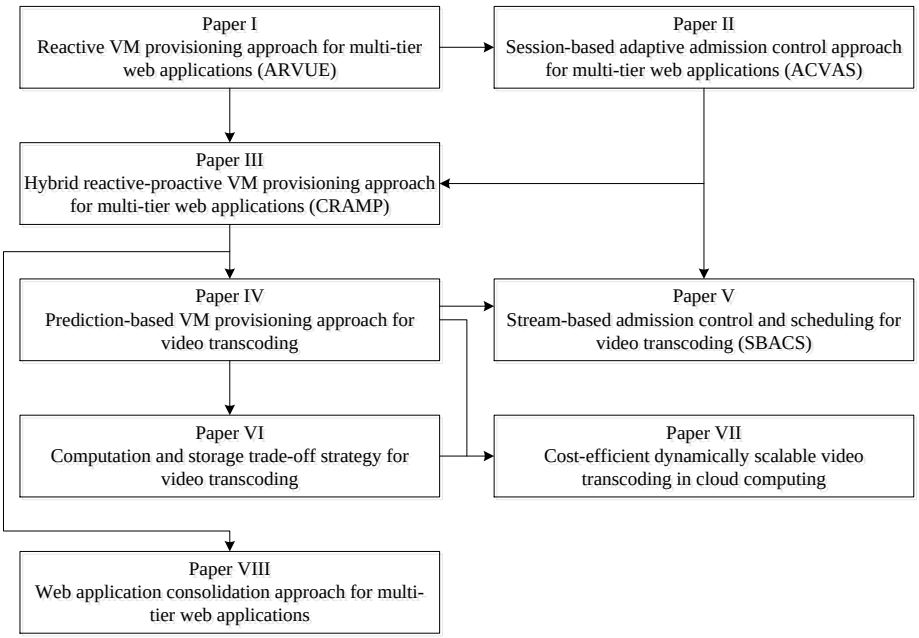

Figure 4.1: Relationship among original publications

uling algorithm for video segments. Paper VI provides a computation and storage trade-off strategy for video transcoding in the cloud. Paper VII extends the works presented in Paper IV and Paper VI, provides two new algorithms, and presents an extended evaluation. Paper VIII uses ACS to consolidate multiple web applications on under-utilized VMs.

Chapter 5

Conclusion

The main problem that we addressed in this thesis is cost-efficient Virtual Machine (VM) provisioning augmented with server consolidation and admission control on the provisioned VMs. We sought solutions for two types of applications: multi-tier web applications that follow the request-response paradigm and on-demand video transcoding that is based on video streams with soft realtime constraints.

For multi-tier web applications, we presented a reactive VM provisioning approach called ARVUE and a hybrid reactive-proactive VM provisioning approach called Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling (CRAMP). These approaches provide automatic deployment and scaling of multiple simultaneous web applications on a given Infrastructure as a Service (IaaS) cloud in a shared hosting environment. The main task of the proposed approaches is to provision and remove VMs for the application server tier and to deploy and remove applications from each VM, in order to maintain a desired Quality of Service (QoS) in a cost-efficient manner. The results from ARVUE and CRAMP are based on our prototype implementation. We also provided a comparison of the results between ARVUE and CRAMP using the Amazon Elastic Compute Cloud (EC2) cloud. The results showed that both ARVUE and CRAMP provide good performance in terms of number of VMs and memory utilization. When compared to the dedicated hosting approaches, the shared hosting of web applications in ARVUE reduced the total VM provisioning cost up to 36%. Moreover, CRAMP provided significantly better performance in terms of average response time and Central Processing Unit (CPU) load average. In contrast to the reactive VM provisioning approach of ARVUE, the hybrid reactive-proactive VM provisioning approach of CRAMP reduced the average response time up to 56% and the maximum CPU load average up to 69%.

For dynamically scalable on-demand video transcoding in the cloud, we proposed a prediction-based VM provisioning approach. It provides a mechanism to create a dynamically scalable cluster of video transcoding servers by provisioning VMs from an IaaS cloud. The proposed approach is demonstrated in a discrete-event simulation. The results indicated that it provides cost-efficient VM provisioning for transcoding a large number of on-demand video streams. The sharing of the VM resources among multiple streams resulted in a reduced number of total VMs. The proposed algorithms did not produce unnecessary oscillations in the number of VMs, which was also desirable for cost-efficiency.

To prevent virtualized application servers from becoming overloaded, we augmented ARVUE with a session-based admission control approach called adaptive Admission Control for Virtualized Application Servers (ACVAS). ACVAS uses per-session admission, which reduces over-admission. It also implements a simple session deferment mechanism, which decreases the number of rejected sessions. The proposed approach is demonstrated in a discrete-event simulation. We also provided a comparison of the results between ACVAS and an existing session-based adaptive admission control approach, which uses on-off control and does not provide a session deferment mechanism. The results showed that ACVAS provides a good trade-off between the number of VMs used and the QoS requirements. In comparison with the alternative admission control approach, ACVAS provided significant improvements in terms of server overload prevention. It also outperformed the alternative approach in reducing the number of rejected sessions.

Similarly, to prevent virtualized video transcoding servers from becoming overloaded, we presented an admission control and scheduling approach called Stream-Based Admission Control and Scheduling (SBACS). It implements stream-based admission control with per-stream admission. In addition to the traditional rejection policy, SBACS also provides a stream deferment policy. It also features a job scheduling algorithm, which complements admission control and prevents transcoding jitters in the admitted streams. The proposed approach is demonstrated in a discrete-event simulation. The results showed that SBACS provides a good trade-off between cost and QoS. It prevents servers from becoming overloaded, reduces over-admission, reduces rejected streams, and reduces transcoding jitters in the admitted streams while dropping only a small proportion of the video frames.

We also presented a computation and storage trade-off strategy for cost-efficient video transcoding in cloud computing. 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 proposed approach is demonstrated in a discrete-event simulation. We also provided a

comparison of the results with two intuitive computation and storage trade-off strategies called store all strategy and usage based strategy. The results indicated that our proposed strategy is more cost-efficient than the two intuitive strategies. It provided significant improvements in terms of the storage cost and the total cost.

Our sixth and last contribution in this thesis is a novel web application consolidation approach, which minimizes under-utilization of virtualized application servers in a cloud-based shared hosting environment. It uses Ant Colony System (ACS) to build a web application migration plan, which is then used to minimize over-provisioning of VMs by consolidating web applications on under-utilized VMs. The proposed approach is demonstrated in a discrete-event simulation. We also provided a comparison of the results with a baseline, greedy application consolidation approach, which is based on an extension of our VM provisioning algorithms in ARVUE and CRAMP. The results showed that the proposed approach provides a more cost-efficient solution for web application consolidation in a cloud-based shared hosting environment. In comparison with the baseline approach, it provided significant improvements in terms of the total number of VM hours, reducing the total VM provisioning cost up to 28%.

5.1 Future Work

Important research directions for our future work include investigating search based software engineering and machine learning approaches to improve and optimize the proposed VM provisioning and admission control approaches. Moreover, for our proposed web application consolidation approach, we used a single-objective ACS algorithm. However, the web application consolidation problem may as well be viewed as a multi-objective combinatorial optimization problem with two objectives: maximize the number of released VMs and minimize the number of application migrations. Therefore, one of our future goals is to formulate the web application consolidation problem as a multi-objective combinatorial optimization problem and then apply a multi-objective Ant Colony Optimization (ACO) algorithm to solve it. Furthermore, investigating other metaheuristic approaches for the web application consolidation problem is also part of our future work.

The results presented in this thesis are promising, but are mostly based on relatively small experiments. Therefore, one of our future goals is to conduct larger and more realistic experiments to fully realize the benefits and limitations of the proposed approaches.

Our proposed VM provisioning, admission control, and consolidation approaches use a centralized controller, which may restrict the scalability of the proposed algorithms. The centralized control in our proposed approaches

also lacks fault-tolerance. We intend to address these issues in the future. One of the most intuitive ways to improve scalability and fault-tolerance of the proposed algorithms is to use a distributed controller. Therefore, we intend to replace the centralized controller with a distributed controller.

The ARVUE and CRAMP prototype implementations presented in this thesis implement our proposed reactive and hybrid reactive-proactive VM provisioning approaches for multi-tier web applications. However, the proposed session-based admission control approach ACVAS and the ACS-based web application consolidation approach are currently implemented only in discrete-event simulations. Therefore, our future work includes implementing and testing ACVAS and our proposed web application consolidation approach on the ARVUE and CRAMP prototype implementations.

Other interesting lines of future work include implementing and testing a prototype for our proposed prediction-based VM provisioning approach for on-demand video transcoding. Similarly, prototype implementations of our proposed stream-based admission control approach SBACS and our computation and storage trade-off strategy for video transcoding are part of our future work.

Bibliography

[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] Timo Aho, Adnan Ashraf, Marc Englund, Joni Katajamäki, Johannes Koskinen, Janne Lautamäki, Antti Nieminen, Ivan Porres, and Ilkka Turunen. Designing IDE as a service. *Communications of Cloud Software*, 1(1), 2011.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Adnan Ashraf. Cost-efficient resource allocation for multi-tier web applications in a cloud environment. In *PhD Symposium held in connection with the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 1–2. Institute for Systems Engineering and Automation, Johannes Kepler University Linz, Austria, 2012.

[9] Adnan Ashraf. Cost-efficient virtual machine provisioning for multi-tier web applications and video transcoding. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 66–69, 2013.

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

[11] 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)*, December 2012.

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

[13] Adnan Ashraf, Benjamin Byholm, and Ivan Porres. Prediction-based virtual machine provisioning and admission control for multi-tier web applications. In Ivan Porres, Tommi Mikkonen, and Adnan Ashraf, editors, *Developing Cloud Software: Algorithms, Applications, and Tools*, pages 71–112. Turku Centre for Computer Science (TUCS) General Publication Number 60, October 2013.

[14] Adnan Ashraf, Mikko Hartikainen, Usman Hassan, Keijo Heljanko, Johan Lilius, Tommi Mikkonen, Ivan Porres, Mahbubul Syeed, and Sasu Tarkoma. Introduction to cloud computing technologies. In Ivan Porres, Tommi Mikkonen, and Adnan Ashraf, editors, *Developing Cloud Software: Algorithms, Applications, and Tools*, pages 1–41. Turku Centre for Computer Science (TUCS) General Publication Number 60, October 2013.

[15] Adnan Ashraf, Fareed Ahmed 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), 2013 13th IEEE/ACM International Symposium on, pages 482–489, 2013.

[16] Adnan Ashraf and Ivan Porres. Using ant colony system to consolidate multiple web applications in a cloud environment. *22nd EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 482–489, 2014.

[17] Jerry Banks. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. A Wiley-Interscience publication. John Wiley & Sons, Inc., 1998.

[18] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[19] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, September 2012.

[20] Christian Blum, Jakob Puchinger, Gnther R. Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151, 2011.

[21] Karin Breitman, Markus Endler, Rafael Pereira, and Marcello Azambuja. When TV dies, will it go to the cloud? *Computer*, 43(4):81–83, 2010.

[22] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[23] Benjamin Byholm. An autonomous Platform as a Service for stateful web applications. Master's thesis, Åbo Akademi University, 2013.

[24] 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.

[25] Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic QoS management and optimization in service-based systems. *Software Engineering, IEEE Transactions on*, 37(3):387–409, 2011.

[26] David Carrera, Malgorzata Steinder, Ian Whalley, Jordi Torres, and Eduard Ayguade. Utility-based placement of dynamic web applications with fairness goals. In *Network Operations and Management Symposium (NOMS), 2008 IEEE*, pages 9–16, 2008.

[27] S. Kamal Chaharsooghi and Amir Hosein Meimand Kermani. An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP). *Applied Mathematics and Computation*, 200(1):167–177, 2008.

[28] Shih-Fu Chang and Anthony Vetro. Video adaptation: Concepts, technologies, and open issues. *Proceedings of the IEEE*, 93(1):148–158, 2005.

[29] 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.

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

[31] Trieu C. Chieu, Ajay Mohindra, Alexei A. Karve, and Alla 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.

[32] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI)*, volume 2, pages 273–286. USENIX Association, 2005.

[33] Antonio Corradi, Mario Fanelli, and Luca Foschini. VM consolidation: A real case based on OpenStack cloud. *Future Generation Computer Systems*, 32(0):118–127, 2014.

[34] Pradipta De, Manish Gupta, Manoj Soni, and Aditya Thatte. Caching VM instances for fast VM provisioning: A comparative evaluation. In *Euro-Par 2012 Parallel Processing*, volume 7484 of *Lecture Notes in Computer Science*, pages 325–336. Springer Berlin Heidelberg, 2012.

[35] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[36] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: The Montage example. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12, 2008.

[37] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant algorithms for discrete optimization. *Artif. Life*, 5(2):137–172, April 1999.

[38] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.

[39] Xavier Dutreilh, Nicolas Rivierre, Aurélien Moreau, Jacques Malenfant, and Isis Truck. From data center resource allocation to control theory and back. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 410–417.

[40] Fahimeh Farahnakian, Pasi Liljeberg, and Juha Plosila. LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers. In *Software Engineering and Advanced Applications (SEAA), 39th EUROMICRO Conference on*, pages 357–364, 2013.

[41] Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, and Juha Plosila. Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers. In *Utility and Cloud Computing (UCC), 6th IEEE/ACM Internatonal Conference on*, 2013.

[42] Eugen Feller, Christine Morin, and Armel Esnault. A case for fully decentralized dynamic VM consolidation in clouds. *Cloud Computing Technology and Science, IEEE International Conference on*, pages 26–33, 2012.

[43] Eugen Feller, Louis Rilling, and Christine Morin. Energy-aware ant colony based workload placement in clouds. In *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, pages 26–33, September 2011.

[44] Eugen Feller, Louis Rilling, and Christine Morin. Snooze: A scalable and autonomic virtual machine management framework for private

clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 482–489, 2012.

[45] Eugen Feller, Louis Rilling, Christine Morin, Renaud Lottiaux, and Daniel Leprince. Snooze: A scalable, fault-tolerant and distributed consolidation manager for large-scale clusters. In *2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom) and International Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 125–132, December 2010.

[46] Tiago C. Ferreto, Marco A.S. Netto, Rodrigo N. Calheiros, and César A.F. De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, 2011.

[47] 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. ACM, 2010.

[48] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 9–16, 2010.

[49] Jordi Guitart, Vicenc Beltran, David Carrera, Jordi Torres, and Eduard Ayguade. Characterizing secure dynamic web applications scalability. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, April 2005.

[50] 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.

[51] Rui Han, Moustafa M. Ghanem, Li Guo, Yike Guo, and Michelle Osmond. Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, 32(0):82–98, 2014.

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

[53] Mark Harman, Kiran Lakhotia, Jeremy Singer, David R. White, and Shin Yoo. Cloud engineering is search based software engineering too. *Journal of Systems and Software*, 86(9):2225–2241, 2013.

[54] Sijin He, Li Guo, Moustafa M. Ghanem, and Yike Guo. Improving resource utilisation in the cloud environment using multivariate probabilistic models. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 574–581, 2012.

[55] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[56] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: a platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011.

[57] 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.

[58] 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.

[59] 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.

[60] Sean Hull. 20 obstacles to scalability. *Communications of the ACM*, 56(9):54–59, September 2013.

[61] Inkwon Hwang and Massoud Pedram. Hierarchical virtual machine consolidation in a cloud computing system. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 196–203, 2013.

[62] 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.

[63] Fareed Ahmed Jokhio, Adnan Ashraf, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Proactive virtual machine allocation for video transcoding in the cloud. In Ivan Porres, Tommi Mikkonen, and Adnan Ashraf, editors, *Developing Cloud Software: Algorithms, Applications, and Tools*, pages 113–143. Turku Centre for Computer Science (TUCS) General Publication Number 60, October 2013.

[64] Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius. A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. *39th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 365–372, September 2013.

[65] Fareed Ahmed 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), December 2013.

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

[67] 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, 2011.

[68] Fareed Ahmed 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 on*, pages 206–212, 2012.

[69] 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. USENIX Association, 2012.

[70] 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 *22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2012.

[71] Xiaofei Liao, Hai Jin, and Haikun Liu. Towards a green cluster through dynamic remapping of virtual machines. *Future Generation Computer Systems*, 28(2):469–477, 2012.

[72] Moreno Marzolla, Ozalp Babaoglu, and Fabio Panzieri. Server consolidation in clouds through gossiping. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, 2011.

[73] Peter Mell and Timothy Grance. The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology. Special Publication 800-145., September 2011. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

[74] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2012.

[75] Gianmario Motta, Nicola Sfondrini, and Daniele Sacco. Cloud computing: An architectural and technological overview. In *Service Sciences (IJCSS), 2012 International Joint Conference on*, pages 23–27, 2012.

[76] 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.

[77] Aziz Murtazaev and Sangyoon Oh. Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. *IETE Technical Review*, 28(3):212–231, 2011.

[78] Wenping Pan, Dejun Mu, Hangxing Wu, and Lei 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, sept. 2008.

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

[80] Rafael Pereira, Marcello Azambuja, Karin Breitman, and Markus Endler. An architecture for distributed high performance video processing in the cloud. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 482–489, 2010.

[81] Yrjo Raivio, Oleksiy Mazhelis, Koushik Annapureddy, Ramasivakarthik Mallavarapu, and Pasi Tyrväinen. Hybrid cloud architecture for short message services. In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, CLOSER '12, 2012.

[82] Anders Robertsson, Björn Wittenmark, Maria Kihl, and Mikael Andersson. Admission control for web server systems - design and experimental evaluation. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 531 –536 Vol.1, dec. 2004.

[83] Nilabja Roy, Abhishek Dubey, and Aniruddha 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.

[84] Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, Erich Nahum, and Adam Wierman. How to determine a good multi-programming level for external scheduling. In *Data Engineering (ICDE), Proceedings of the 22nd International Conference on*, April 2006.

[85] 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.

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

[87] Ilhoon Shin and Kern Koh. Hybrid transcoding for QoS adaptive video-on-demand services. *Consumer Electronics, IEEE Transactions on*, 50(2):732–736, 2004.

[88] Ilango Sriram and Ali Khajeh-Hosseini. Research agenda in cloud technologies. Technical report, Large Scale Complex IT Systems (LSCITS), 2010.

[89] Klaus Stuhlmüller, Niko Färber, Michael Link, and Bernd Girod. Analysis of video transmission over lossy channels. *Selected Areas in Communications, IEEE Journal on*, 18(6):1012–1032, 2000.

[90] The OSGi Alliance. *OSGi Service Platform Core Specification, Release 4, Version 4.3*. 2011.

[91] Bhuvan Urgaonkar, Prashant Shenoy, and Timothy Roscoe. Resource overbooking and application profiling in a shared internet hosting platform. *ACM Trans. Internet Technol.*, 9(1):1–45, February 2009.

[92] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

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

[94] Werner Vogels. Beyond server consolidation. *ACM Queue*, 6(1):20–26, January 2008.

[95] Thiemo Voigt and Per Gunningberg. Adaptive resource-based web server admission control. In *Computers and Communications (ISCC), 2002 Seventh International Symposium on*, pages 219–224.

[96] Meng Wang, Xiaoqiao Meng, and Li Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *Proceedings of IEEE INFOCOM*, pages 71–75, April 2011.

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

[98] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, July 2003.

[99] 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.

[100] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, 2009.

[101] Jun Xin, Chia-Wen Lin, and Ming-Ting Sun. Digital video transcoding. *Proceedings of the IEEE*, 93(1):84–97, 2005.

[102] Peng-Yeng Yin and Jing-Yu Wang. Ant colony optimization for the nonlinear resource allocation problem. *Applied Mathematics and Computation*, 174(2):1438–1453, 2006.

[103] 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.

[104] 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.

[105] 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.

[106] 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.

[107] Zhenzhong Zhang, Haiyan Wang, Limin Xiao, and Li Ruan. A statistical based resource allocation scheme in cloud. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 266–273, 2011.

[108] Han Zhao, Miao Pan, Xinxin Liu, Xiaolin Li, and Yuguang Fang. Optimal resource rental planning for elastic applications in cloud market. In *Parallel and Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 808–819, 2012.

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

Acronyms

ACES Admission Control based on Estimation of Service times

ACO Ant Colony Optimization

ACS Ant Colony System

ACVAS adaptive Admission Control for Virtualized Application Servers

AS Ant System

AWS Amazon Web Services

CBAC Cost-Based Admission Control

CoSAC Coordinated Session-based Admission Control

CPU Central Processing Unit

CRAMP Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling

CTT-SP Cost Transitive Tournament Shortest Path

DDG Data Dependency Graph

EC2 Elastic Compute Cloud

EMA Exponential Moving Average

GAE Google App Engine

GOP Group of Pictures

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a Service

IT Information Technology

JVM Java Virtual Machine

MMAS Max-Min Ant System

NIST National Institute of Standards and Technology

NRMSE Normalized Root Mean Square Error

OSGi Open Services Gateway initiative

PaaS Platform as a Service

PD Proportional-Derivative

PI Proportional-Integral

PM Physical Machine

QoS Quality of Service

RQ Research Question

S3 Simple Storage Service

SaaS Software as a Service

SBAC Session-Based Admission Control

SBACS Stream-Based Admission Control and Scheduling

SLA Service Level Agreement

TSP Travelling Salesman Problem

VM Virtual Machine

Complete List of Original Publications

This thesis is composed of 8 original publications, which are included in Part II of the thesis. However, the research work presented in this thesis also closely relates to some other publications of the author. The complete list of original publications published in relation to this thesis is as follows. It includes all publications, whether or not included in Part II of the thesis.

1. Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres. Feedback Control Algorithms to Deploy and Scale Multiple Web Applications per Virtual Machine. In *Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2012)*, pp. 431–438, September 2012, Cesme, Izmir, Turkey.

2. Adnan Ashraf, Benjamin Byholm, and Ivan Porres. A Session-Based Adaptive Admission Control Approach for Virtualized Application Servers. In *Proceedings of the 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012)*, pp. 65–72, November 2012, Chicago, IL, USA.

3. Adnan Ashraf, Benjamin Byholm, and Ivan Porres. CRAMP: Cost-Efficient Resource Allocation for Multiple Web Applications with Proactive Scaling. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pp. 581–586, December 2012, Taipei, Taiwan.

4. Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing. In *Proceedings of the 21st EUROMICRO International Conference on Parallel, Distributed and Network-based Processing (PDP 2013)*, pp. 254–261, February 2013, Belfast, UK.

5. Adnan Ashraf, Fareed Ahmed Jokhio, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Stream-Based Admission Control

and Scheduling for Video Transcoding in Cloud Computing. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, pp. 482–489, May 2013, Delft, the Netherlands.

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

7. Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, Ivan Porres, and Johan Lilius. Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing. *Turku Centre for Computer Science (TUCS) Technical Reports*, number 1098, pp. 1–25, December 2013.

8. Adnan Ashraf and Ivan Porres. Using Ant Colony System to Consolidate Multiple Web Applications in a Cloud Environment. In *Proceedings of the 22nd EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP 2014)*, pp. 482–489, February 2014, Turin, Italy.

9. Fahimeh Farahnakian, Adnan Ashraf, Pasi Liljeberg, Tapio Pahikkala, Juha Plosila, Ivan Porres, and Hannu Tenhunen. Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System. In *Proceedings of the 7th IEEE International Conference on Cloud Computing (IEEE CLOUD 2014)*, pp. 104–111, June 2014, Alaska, USA.

10. Ivan Porres, Tommi Mikkonen, and Adnan Ashraf (Editors). Developing Cloud Software: Algorithms, Applications, and Tools. *Turku Centre for Computer Science (TUCS) General Publication Number 60*, October 2013.

11. Adnan Ashraf, Benjamin Byholm, and Ivan Porres. Prediction-Based Virtual Machine Provisioning and Admission Control for Multi-Tier Web Applications. Book chapter in *Ivan Porres, Tommi Mikkonen, and Adnan Ashraf (Editors). Developing Cloud Software: Algorithms, Applications, and Tools. Turku Centre for Computer Science (TUCS) General Publication Number 60*, pp. 71–112, October 2013.

12. Fareed Ahmed Jokhio, Adnan Ashraf, Tewodros Deneke, Sébastien Lafond, Ivan Porres, and Johan Lilius. Proactive Virtual Machine Allocation for Video Transcoding in the Cloud. Book chapter in *Ivan*

Porres, Tommi Mikkonen, and Adnan Ashraf (Editors). Developing Cloud Software: Algorithms, Applications, and Tools. Turku Centre for Computer Science (TUCS) General Publication Number 60, pp. 113–143, October 2013.

13. Adnan Ashraf, Mikko Hartikainen, Usman Hassan, Keijo Heljanko, Johan Lilius, Tommi Mikkonen, Ivan Porres, Mahbubul Syeed, and Sasu Tarkoma. Introduction to Cloud Computing Technologies. Book chapter in *Ivan Porres, Tommi Mikkonen, and Adnan Ashraf (Editors). Developing Cloud Software: Algorithms, Applications, and Tools. Turku Centre for Computer Science (TUCS) General Publication Number 60*, pp. 1–41, October 2013.

14. Adnan Ashraf. Cost-Efficient Virtual Machine Provisioning for Multi-tier Web Applications and Video Transcoding. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, pp. 66–69, May 2013, Delft, the Netherlands.

15. Adnan Ashraf. Cost-Efficient Resource Allocation for Multi-Tier Web Applications in a Cloud Environment. In *PhD Symposium held in connection with the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2012)*, pp. 1–2, September 2012, Cesme, Izmir, Turkey. (Best Doctoral Research Proposal Award)

16. Timo Aho, Adnan Ashraf, Marc Englund, Joni Katajamäki, Johannes Koskinen, Janne Lautamäki, Antti Nieminen, Ivan Porres, and Ilkka Turunen. Designing IDE as a Service. In *Communications of Cloud Software*, 1(1), pp. 1–10, December 2011.

17. Niclas Snellman, Adnan Ashraf, Ivan Porres. Towards Automatic Performance and Scalability Testing of Rich Internet Applications in the Cloud. In *Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2011)*, pp. 161–169, September 2011, Oulu, Finland.

Part II

Original Publications

Publication I

Feedback Control Algorithms to Deploy and Scale Multiple Web Applications per Virtual Machine

Adnan Ashraf, Benjamin Byholm, Joonas Lehtinen, and Ivan Porres

Originally published in *Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2012)*, pp. 431–438, September 2012, Cesme, Izmir, Turkey.

©2012 IEEE. Reprinted with permission from the publisher.

Publication II

A Session-Based Adaptive Admission Control Approach for Virtualized Application Servers

Adnan Ashraf, Benjamin Byholm, and Ivan Porres

Originally published in *Proceedings of the 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012)*, pp. 65–72, November 2012, Chicago, IL, USA.

©2012 IEEE. Reprinted with permission from the publisher.

Publication III

CRAMP: Cost-Efficient Resource Allocation for Multiple Web Applications with Proactive Scaling

Adnan Ashraf, Benjamin Byholm, and Ivan Porres

Originally published in *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pp. 581–586, December 2012, Taipei, Taiwan..

©2012 IEEE. Reprinted with permission from the publisher.

Publication IV

Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing

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

Originally published in *Proceedings of the 21st EUROMICRO International Conference on Parallel, Distributed and Network-based Processing (PDP 2013)*, pp. 254–261, February 2013, Belfast, UK.

©2013 IEEE. Reprinted with permission from the publisher.

Publication V

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

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

Originally published in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*, pp. 482–489, May 2013, Delft, the Netherlands.

©2013 IEEE. Reprinted with permission from the publisher.

Publication VI

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

Fareed Ahmed Jokhio, Adnan Ashraf, Sébastien Lafond, and Johan Lilius

Originally published in *Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2013)*, pp. 365–372, September 2013, Santander, Spain.

©2013 IEEE. Reprinted with permission from the publisher.

Publication VII

Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing

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

Originally published in *Turku Centre for Computer Science (TUCS) Technical Reports*, number 1098, pp. 1–25, December 2013.

©2013 Turku Centre for Computer Science (TUCS). Reprinted with permission from the publisher.

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.

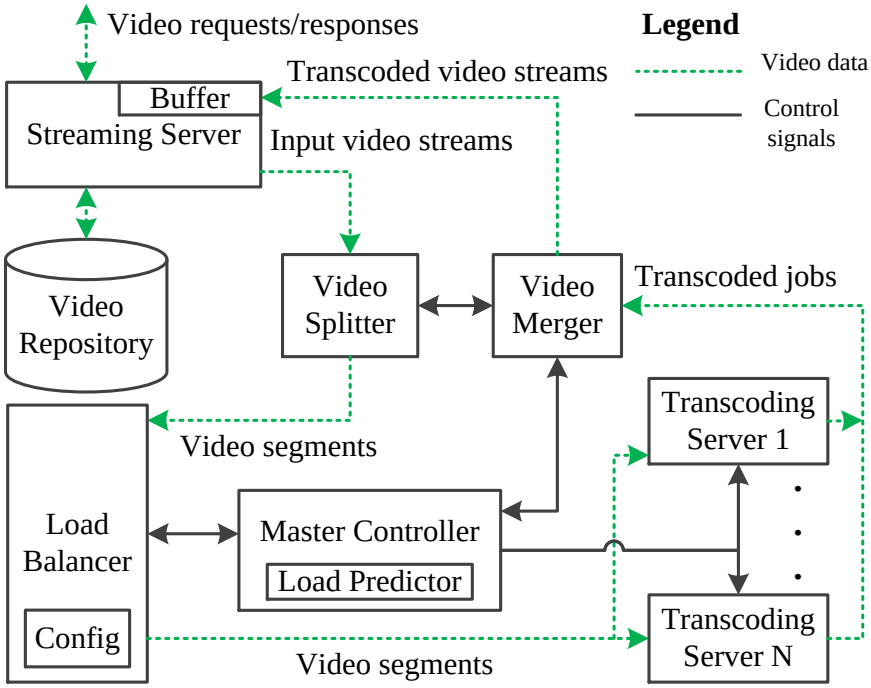

Figure 1: System architecture of the cloud-based on-demand video transcoding service

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

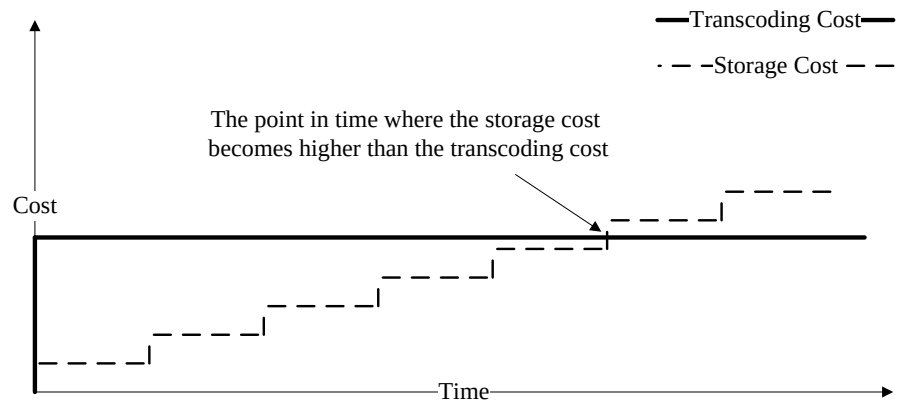

Figure 2: The estimated equilibrium point between the storage cost and the transcoding cost of a transcoded video

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.

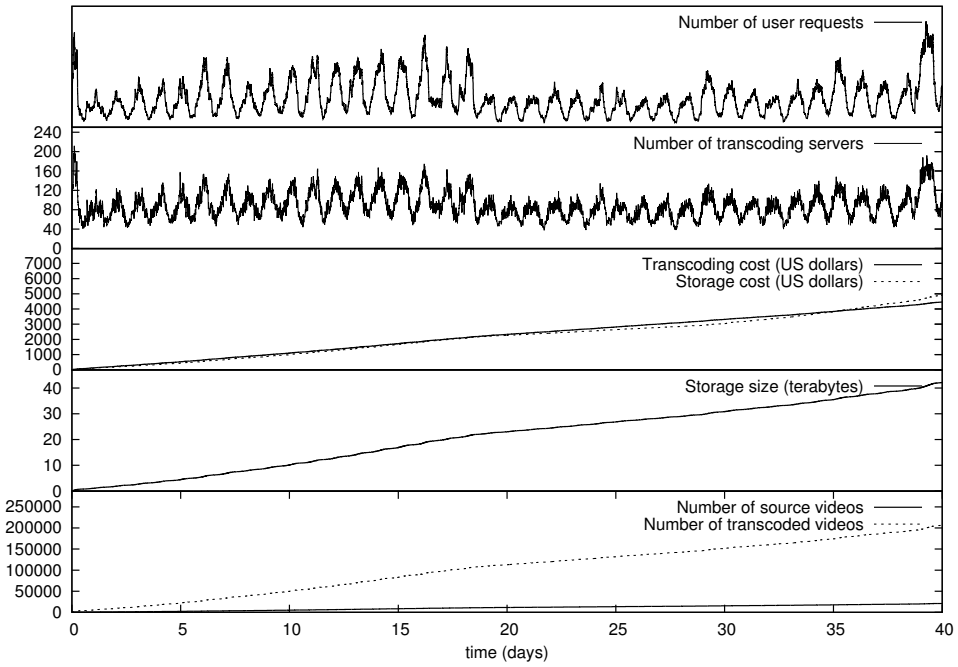

Figure 3: Store all strategy

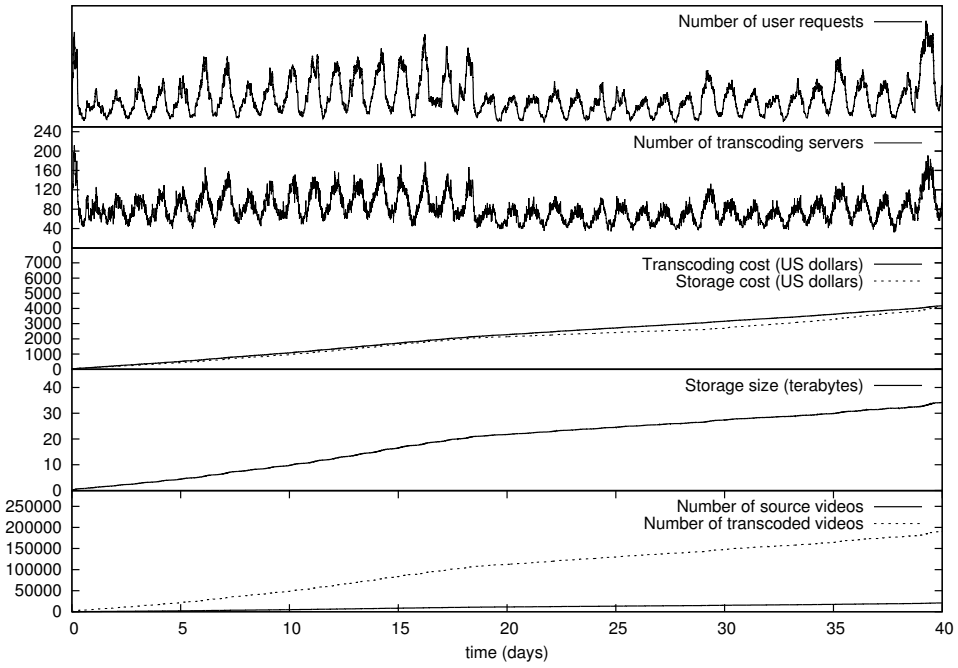

Figure 4: Usage based strategy

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]

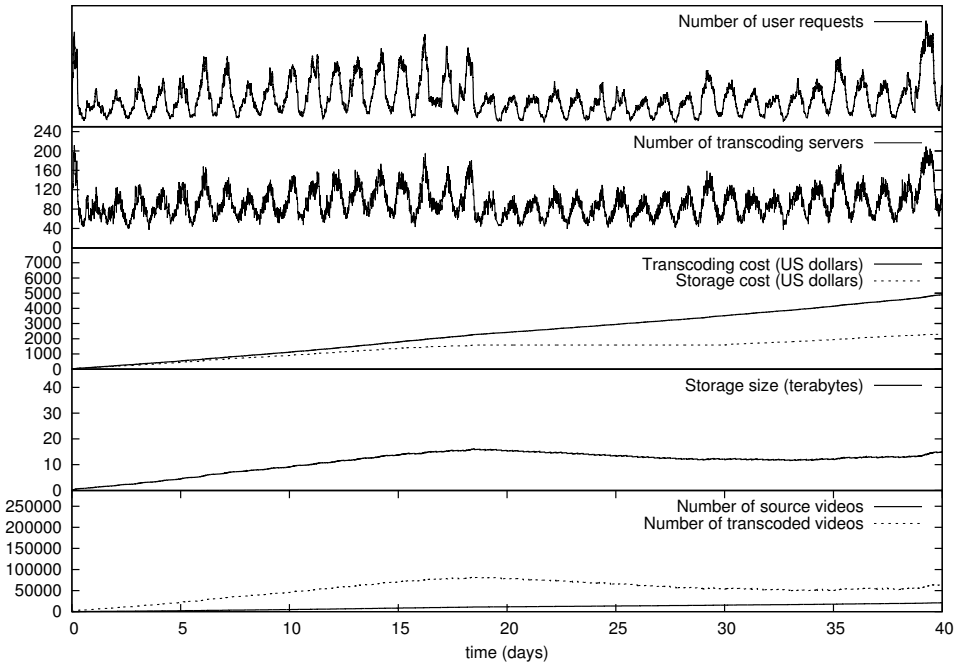

Figure 5: Proposed score based strategy

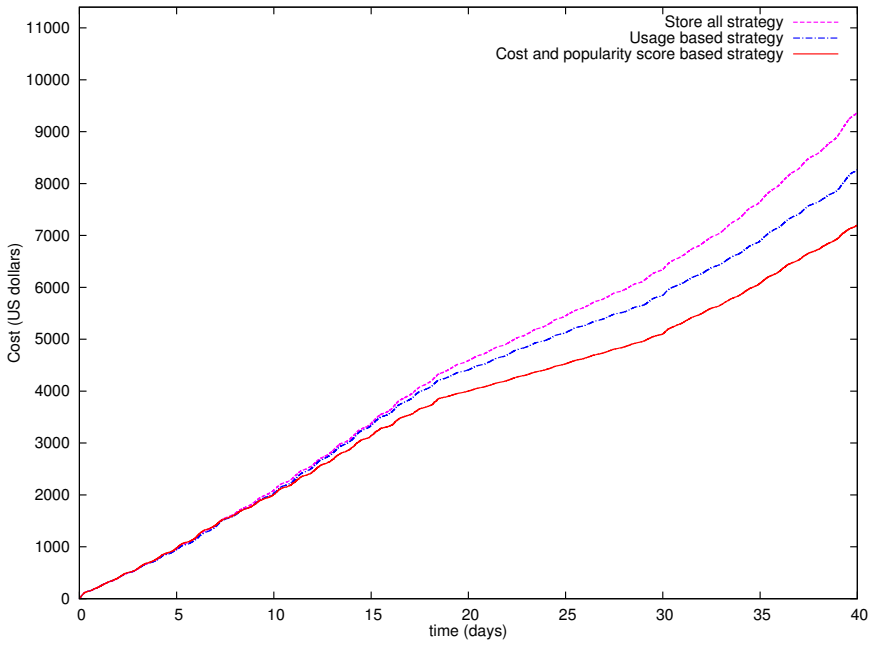

Figure 6: Cost comparison

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.

Publication VIII

Using Ant Colony System to Consolidate Multiple Web Applications in a Cloud Environment

Adnan Ashraf and Ivan Porres

Originally published in *Proceedings of the 22nd EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP 2014)*, pp. 482–489, February 2014, Turin, Italy.

©2014 IEEE. Reprinted with permission from the publisher.

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
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains
180. **Ville Salo**, Subshifts with Simple Cellular Automata
181. **Johan Ersfolk**, Scheduling Dynamic Dataflow Graphs
182. **Hongyan Liu**, On Advancing Business Intelligence in the Electricity Retail Market
183. **Adnan Ashraf**, Cost-Efficient Virtual Machine Management: Provisioning, Admission Control, and Consolidation

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 978-952-12-3111-7
ISSN 1239-1883

Adnan Ashraf

Cost-Efficient Virtual Machine Management

Adnan Ashraf

Cost-Efficient Virtual Machine Management

Adnan Ashraf

Cost-Efficient Virtual Machine Management