Workload Characterization, Controller Design and Performance Evaluation for Cloud Capacity Autoscaling

Ahmed Ali-Eldin Hassan

PhD thesis, 2015
Department of Computing Science
Umeå University
SE-901 87 Umeå
Sweden

This work has been generously supported by the Swedish Government's strategic effort eSSENCE, the European Community's Seventh Framework Programme under grant agreement #257115 for the project OPTIMIS, the European Union Seventh Framework Programme under grant agreement 610711 for the project CACTOS, and the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. Part of the research was conducted using the resources of High Performance Computing Center North (http://www.hpc2n.umu.se/).

UMINF:15.09
ISSN: 0348-0542
ISBN: 978-91-7601-330-4

Electronic version available at http://umu.diva-portal.org/
Printed by Print & Media, Umeå University
Umeå, Sweden 2015

Abstract

This thesis studies cloud capacity auto-scaling, or how to provision and release resources to a service running in the cloud based on its actual demand using an automatic controller. As the performance of server systems depends on the system design, the system implementation, and the workloads the system is subjected to, we focus on these aspects with respect to designing auto-scaling algorithms. Towards this goal, we design and implement two auto-scaling algorithms for cloud infrastructures. The algorithms predict the future load for an application running in the cloud. We discuss the different approaches to designing an auto-scaler combining reactive and proactive control methods, and to be able to handle long running requests, e.g., tasks running for longer than the actuation interval, in a cloud. We compare the performance of our algorithms with state-of-the-art auto-scalers and evaluate the controllers' performance with a set of workloads. As any controller is designed with an assumption on the operating conditions and system dynamics, the performance of an auto-scaler varies with different workloads.

In order to better understand the workload dynamics and evolution, we analyze a 6-years long workload trace of the sixth most popular Internet website. In addition, we analyze a workload from one of the largest Video-on-Demand streaming services in Sweden. We discuss the popularity of objects served by the two services, the spikes in the two workloads, and the invariants in the workloads. We also introduce, a measure for the disorder in a workload, i.e., the amount of burstiness. The measure is based on Sample Entropy, an empirical statistic used in biomedical signal processing to characterize biomedical signals. The introduced measure can be used to characterize the workloads based on their burstiness profiles. We compare our introduced measure with the literature on quantifying burstiness in a server workload, and show the advantages of our introduced measure.

To better understand the tradeoffs between using different auto-scalers with different workloads, we design a framework to compare auto-scalers and give probabilistic guarantees on the performance in worst-case scenarios. Using different evaluation criteria and more than 700 workload traces, we compare six state-of-the-art auto-scalers that we believe represent the development of the field in the past 8 years. Knowing that the auto-scalers' performance depends on the workloads, we design a workload analysis and classification tool that assigns a workload to its most suitable elasticity controller out of a set of implemented controllers. The tool has two main components; an analyzer, and a classifier. The analyzer analyzes a workload and feeds the analysis results to the classifier. The classifier assigns a workload to the most suitable elasticity controller based on the workload characteristics and a set of predefined business level objectives. The tool is evaluated with a set of collected real workloads, and a set of generated synthetic workloads. Our evaluation results shows that the tool can help a cloud provider to improve the QoS provided to the customers.

Sammanfattning

Denna avhandling studerar autoskalning, det vill säga hur en regulator automatiskt kan allokera och avallokera resurser för en tjänst som körs i molnet beroende på tjänstens belastningsmönster. Prestandan för ett serversystem beror på systemets design, dess implementation och den belastning systemet utsätts för. Denna avhandling fokuserar på dessa aspekter för design av algoritmer för autoskalning. Två algoritmer för autoskaling i molninfrastrukturer designas och implementeras. Algoritmerna predikterar framtida belastningsmönster för en tjänst som körs i molnet. Olika ansatser till att designa en regulator som kombinerar reaktiva och proaktiva reglermetoder utvärderas. Vi designar även metoder för att hantera serverförfrågningar som tar lång tid att betjäna, det vill säga beräkningar som körs längre än regulatorns aktiveringsintervall.

Vi jämför våra algoritmer med existerande autoskalare från forskningsfronten och utvärderar dess prestanda med hjälp av olika belastningskurvor. Då alla regulatorer är designade med vissa antaganden om belastningönster och datorsystemets dynamik varierar en regulators prestanda för olika typer av belastningskurvor.

För att bättre förstå egenskaperna hos belastningskurvor och hur dessa utvecklas med tiden analyserar vi 6 år av loggdata från Wikipedia, den sjätte mest populära websidan på Internet. Vi analyserar även en belastningskurva från en av de största videostreaming-tjänsterna i Sverige. I dessa analyser studeras populariteten hos dataobjekt på dessa två tjänster, hur belastningstoppar ser ut samt invarianter i belastningen, exempelvis periodiska mönster som å terkommer varje dag eller vecka. Vi introducerar även ett volatilitetsmått för oordningen i en belastningkurva, vilket är baserat på metoder och mått som används för signalbehandling inom biomedicin. Det nya måttet kan användas för att karaktärisera belastningskurvor. Vi utvärderar det mot existerande mått från forskningslitteraturen och på visar fördelar.

För att bättre förstå avvägningar mellan olika autoskalare för olika belastningsmön ster designar vi ett ramverk för att jämföra autoskalare. Ramverket ger probabilistiska garantier för autoskalarnas prestanda i värsta-fall-scenarion. Med olika utvärderingskriteria och över 700 olika belastningskurvor jämför vi sex olika autoskalningsalgoritmer från litteraturen, vilka representerar forskningsområdets utveckling under de senaste åtta åren.

Då prestandan hos en autoskalare beror på belastningskurvan designar vi ett analys- och klassificeringsverktyg som matchar en belastningskurva till den mest lämpliga av en mängd tillgängliga autoskalare. Vilken autoskalare som väljs beror på belastningskurvans egenskaper och på förvalda preferenser såsom prestandakriterier. Verktyget utvärderas med både riktiga belastningskurvor och syntetiskt genererade belastningsmönster. Utvärderingen visar att verktyget i de allra flesta fall väljer den bäst lämpare autoskalaren och därmed att det kan hjälpa en molnleverantör att få mer responsiva tjänster och samtidigt minska resursanvändningen.

Acknowledgements

This is the end of a journey that started 30 years ago. I will thank people in chronological order of meeting. To my parents, thank you for giving me everything you had, and everything I have. Now that I am a father, I know what it is like to have a kid who loved to experiment with boiling water. Part of this journey was to make you proud! I love you! To my siblings, Maie, Yomna, and Hosam, I love you, you are great and you are all what anyone would want as siblings. I miss being the evil brother :). My first few experiments were done on you as subjects, and boy, they were fun!

To my wife, I met you first one week before starting my PhD. You were foolish enough to get engaged to me after having long interrogations, with you interrogating me :). You have written more of this thesis than I did with your support and love. This journey has been rough for you. I hope I will be able to pay back some of my debt. Thank you for believing in me, giving us our children, and dealing with my tantrums :). I love you and I look forward to our new adventures!

To my advisors, Erik and Johan, I still remember my interviews with you like it was yesterday. You have made this group among the best in the world. You have helped me build my career and be (hopefully) a good scientist. You are above all friends I cherish! It has been fun working with you! I am looking forward to all the great things we will do together in the future! Erik, you are awesome on so many levels, hard to sum up actually! Johan, your help when I was stuck in the beginning, and your always positive suggestions and feedback made this possible!

To my colleagues in the group, are not you the most awesome research group in the world? Thank you Lars, Peter, Petter, P-O, Mina, Ewnetu, Muyi, Amardeep, Viali, Jakub, Selome, Gonzalo, Abel, Luis, Cristian, Francisco, Tomas, Lennart, Daniel, Lei, and Christina. This is in no particular order. Each one of you made this a great place! To my department colleagues, well I can not thank you enough for making this a very special place for me! I will always remember this department as my academic birthplace. Keep up the great work. Special thanks go to my friends Emad and Mahmoud!

To my Collaborators, this thesis was a result of endless late-night work and discussions with you. Thank you Alessandro, Sara, Oleg, Maria, Tania, Amardeep, Ali, Stas, and Karl-Erik! Working with you has been a lot of fun and learning! Looking forward to continue these collaborations with less late-nights!

To my daughter, Salma, in our deen, having a daughter is a privilege. You light the days and nights. You will probably not read this until 10 years, but daddy loves you. To my son, Yusuf, my little man, you came to life with a miracle, you are a miracle who touched my life. Daddy loves you too! Both of you make my life meaningful. To Mohamed, Mariam and Abul-Rahman, I love you!

To the people who died in my country between January, 2011 (roughly when I started my PhD), until this day. Frequently, I envy you for being in a better place. I get the "why-not-me?" syndrome. May god bless your souls! I promise to live for your cause, and tell my children about you.

This part of the thesis should have been the longest part. I did my best to keep it short. Thank you!

Preface

This thesis consists of an introductory chapter, the following papers, and an appendix:

I. A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 204-212, IEEE, 2012.

II. A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing (ScienceCloud)*, pages 31-40, ACM, 2012.

III. A. Ali-Eldin, A. Rezaie, A. Mehta, S. Razroevy, S. Sjöstedt-de Luna, O. Seleznjev, J. Tordsson, and E. Elmroth. How will your workload look like in 6 years? analyzing wikimedia's workload, In *Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E)*, pages 349-354, IEEE Computer Society, 2014.

IV. A. Ali-Eldin, O. Seleznjev, S. Sjöstedt-de Luna, J. Tordsson, and E. Elmroth. Measuring cloud workload burstiness, *IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*, pages 566 - 572, IEEE, 2014

V. A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Analysis and characterization of a Video-on-Demand service workload, *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 189-200, ACM, 2015.

VI. A. Papadopoulos, A. Ali-Eldin, J. Tordsson, K.E. Årzén, and E.Elmroth. PEAS: A Performance Evaluation framework for Auto-Scaling strategies in cloud applications. *Submitted for Journal Publication.*

VII. A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl. WAC: A Workload analysis and classification tool for automatic selection of cloud auto-scaling methods. *To be submitted.*

Introduction

1 Introduction

Cloud computing is a computing model that allows better management, higher utilization and reduced operating costs for datacenter operators while providing on-demand resource provisioning for multiple customers. Datacenters are often enormous in size and complexity. This complexity is further increased because of the multitude of (complex) hosted applications [47], making datacenter management a very complex task [11]. The management of a cloud datacenter is too complex for a human to manage and automated management systems are required. This thesis studies automated cloud elasticity management, one of the main datacenter management capabilities. Elasticity can be defined as the ability of cloud infrastructures to rapidly change *the amount of resources* allocated to an application in the cloud according to its demand. This work studies algorithms, techniques and tools that a cloud provider can use to automate dynamic resource provisioning allowing the provider to better manage the datacenter resources.

Performance of Internet-scale server systems like cloud systems depends on three main factors; design of the system, implementation of the system, and the load on the system [19]. The first two factors should be optimized during system design and implementation. The third factor should be optimized at run-time. Variations in system load can be attributed to either internal events in the application or external events. Internal events can occur in the servers' hardware, e.g., bad blocks on a solid state drive [43], the operating system, e.g., thread scheduling, or the applications running on the servers. Such events cause the occurrence of very short bottlenecks that reduce system performance considerably [56]. On the other hand, external events include, for example, changes in the characteristics of the workload volume [12] or the workload mix [51]. As an example, Internet traffic due to Michael Jackson's death resulted in disruptions in many major Internet services [46]. Nowadays, a drop in QoS is often reflected in a financial loss [15]. In the future, a decrease in the QoS might be reflected in a loss of a life, e.g., in case of self-driving cars relying on cloud systems [33].

The performance of networked systems has been studied since Kleinrock's early work on queuing theory and modeling in the 1970s [32], typically with an aim of optimizing server system design and implementation. Little focus has been given to optimizing run-time performance until 15 years ago when Chase et al. published their seminal work on controlling provisioned resources according to load variations [14]. A few years before Chase et al.'s paper, Foster and Kesselman introduced Grid computing [21]. These two papers were followed by Kephart's and Chess's paper on autonomic computing [31]. These advances coupled with advances in virtualization technologies [2,9], containers [40], server power management techniques [18] and increased network bandwidth has paved the way for the era of cloud computing [10] where elastic server infrastructures allow on-demand resource provisioning which is the focus of our dissertation.

1.1 What is Cloud Computing?

Cloud computing started as a technology hype-term to describe a lot of different systems leading to a lot of confusions on what the term refers to [8, 55]. The confusion was reduced when the US National Institute of Standards and Technology (NIST) defined and characterized the important aspects of cloud computing in a special publication [38]. NIST defined cloud computing as "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".

NIST identifies in their cloud definition five essential characteristics for cloud services [38],

I. On-demand self service. Consumers can provision computing capabilities, e.g., server time and network storage, and control them automatically with no human interaction required.

II. Broad network access. The computing capabilities are available and accessible over the network through heterogeneous client platforms, e.g., mobile phones, tablets and laptops.

III. Resource pooling. Resources are pooled between multiple consumers to serve each customer's requirements.

IV. Rapid elasticity. Computing capabilities can be provisioned rapidly on-demand to deal with load changes. The provisioning can be in some cases automated.

V. Measured service. Usage can be monitored, controlled and reported transparently.

A key challenge to fulfill the cloud model is the scale of these systems. Current cloud datacenters are enormous in size. For example, on the 31st of December, 2014, Rackspace, a market leader in providing cloud computing services, had more than 112628 servers hosted in 9 datacenters serving more than 300000 customers [47]. A typical server has between 4 cores and 128 cores. Management of such huge and complex systems requires some automation in the management software. On-demand provisioning and resource pooling requires the middleware to be able to handle the provisioning demands for the customer and allocate the resources required by the service autonomically and transparently [31]

2 Cloud Capacity Auto-Scaling

Auto-scaling cloud resources, sometimes referred to as elasticity control or management, is an incarnation of the dynamic provisioning of server resources problem, a problem which has been studied for over a decade [13, 14]. For the past decade and

half, there has been tens – if not hundreds – of algorithms and techniques proposed in the literature tackling the same question; how to dynamically change the amount of provisioned resources for an application running on server systems according to the varying application's load [3, 7, 13, 14, 20, 22–29, 35–37, 39, 44, 45, 49–53]?

Horizontal elasticity is the property of a cloud datacenter to rapidly vary the number of VMs allocated to a service according to demand. Vertical elasticity is the property of a cloud datacenter to rapidly change configurations of virtual resources allocated to a service according to demand, e.g., adding more CPU power, memory or disk space to already running VMs. Autoscalers utilize the elasticity properties of a cloud by dynamically allocating sufficient capacity to a running service to maintain an acceptable level of QoS at reduced costs [54]. This dissertation focuses on automated horizontal auto-scaling of cloud resources based on workload dynamics, i.e., we focus on auto-scalers that change the number of VMs allocated to a service running in the cloud.

2.1 Auto-Scaling Controller Requirements

We have identified five key requirements for an auto-scaler for cloud resources. These requirements, in our opinion, are essential to make the auto-scaler useful and safe to use. The requirements are;

I. Scalability: It has been estimated in 2014 that Amazon's EC2 operates around one and half million servers running millions of virtual machines [42]. One single service can have up to a few thousand virtual machines [16]. Some services run in the cloud for a short period of time while other services can run for years , e.g., reddit has been running on EC2 entirely since 2009 [48]. Algorithms used for automated elasticity must be scalable with respect to the amount of resources running, the monitoring data analyzed and the time for which the algorithm has been running.

II. Adaptiveness: Workloads of Internet and cloud applications are dynamic [30]. An automated elasticity controller should be able to adapt to the changing workload dynamics or changes in the system models, e.g., new resources added or removed from the system. This should be done while reducing both *overprovisioning and underprovisioning* of resources.

III. Rapid: An automated elasticity algorithm should compute the required capacity rapidly enough to preserve the QoS requirements. Sub-optimal configurations that preserve the QoS requirements are better than any optimal configuration taking longer to compute than the time that can be tolerated by the users or the application to preserve the QoS required. Limited lookahead control is a very accurate technique for the estimation of the required resources but according to one study, it requires almost half an hour to come up with an accurate solution for 15 physical machines each hosting 4 Virtual Machines (VMs) [34].

IV. Robustness: Changing load dynamics might lead to a change in the controller behavior [41]. A robust controller should prevent oscillations in resource allocation and/or in performance with respect to the QoS requirements. While adaptiveness describes the ability of the controller to adapt to changing workloads, robustness describes the stability of the controller with changing workload and/or system dynamics. A controller might be able to adapt to the workload but with oscillations in resource allocation that results in application instability, e.g., adding and removing VMs to a traditional SQL database tier incurs high penalties with respect to consistency. Another controller might not be able to adapt to the changing workload, but is stable with changing workload or system dynamics, i.e., it is robust but not adaptive.

V. QoS and cost awareness: The automated elasticity algorithm should be able to vary the capacity allocated to a service according to demand while enforcing the QoS requirements. If the algorithm provisions less resources than required, then QoS may deteriorate, leading to possible losses. When the algorithm provisions extra capacity that is not needed by the service, then there is a waste of resources. In addition, the costs of the extra unneeded capacity increases the costs of running a service in the cloud.

2.2 Thesis Position Statement

The focus of our work started as an effort to design new auto-scaling algorithms for cloud applications. It became evident from the very start that while one can design a few hundred more auto-scaling algorithms using state-of-the-art techniques from different disciplines such as control theory, neural network, fuzzy logic, statistical inference, estimation theory, queuing theory, optimization and machine learning, there is a genuine lack of understanding of the dynamic provisioning problem stemming from the lack of understanding of the workloads. By lack of understanding of the problem we mean that most of the current work on auto-scaling is based on trial and error and empirical designs. While one can test a proposed algorithm under certain assumptions and conditions, and for a specific application, no one really knows how to generalize these results. Going back to the opening statement of the thesis, the performance of the system and the proposed auto-scaling algorithm depends on the system design, implementation and workloads.

Our position is that *deeper understanding of cloud workloads is needed to tackle and solve the dynamic resource provisioning problem.* Workloads can be classified into different classes based on their quantitative, and qualitative characteristics with respect to an application. For example, some workloads are bursty, e.g., the load on the Bulgarian Wikitionary project, a project hosted by the Wikimedia foundation [1], shown in Figure 1. If a traditional database system is subject to a bursty workload pattern where changing the amount of provisioned resources can come at a very high cost due to consistency and replication, then the auto-scaler should not for example cause oscillations in the resources provisioned.

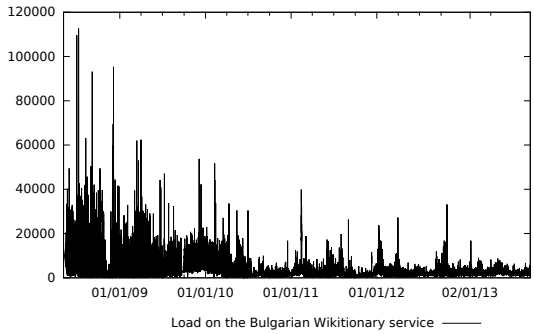

Figure 1: The Bulgarian Wikitionary project has bursty access patterns.

Our position is that *the vast space of the workloads and applications can be divided into smaller sub-spaces with each sub-space representing a category of workloads and applications with similar quantitative and qualitative characteristics.* The problem is then reduced to finding an auto-scaler for each category. The sub-spaces are multi-dimensional, with each dimension representing a feature or a characteristic of the workload or the application. Examples of workload features include periodicity, burstiness and the load-mix, i.e., the different request types [51]. Examples of application features include the amount of resources required to process one request from a certain type, the tolerance of the application to provisioning decisions, the application's ability to process delayed requests and the time required to process one request of a certain type. For example, a simple webserver application serving static content will require much less resources and processing time per request compared to an application that does optical character recognition online.

While this work does not define these categories, we show that they exist. Paper VII and Paper VIII show the varying performance of state-of-the-art auto-scaling algorithms and how one algorithm can be much better than another one for one workload or application compared to another one. Paper V introduces Sample Entropy, a measure for burstiness, one of the main features to classify workloads. Sample Entropy is used in Paper III, with other workload and application features, to classify different workloads using a k-Nearest-Neighbors classifier. The classification is used to assign each workload to the most suitable auto-scaling algorithm given some QoS constraints, e.g., on overprovisioning and underprovisioning. In papers I and II, we develop an auto-scaling algorithm capable of handling queued requests. In order to understand some of the cloud workloads, we provide deep analysis for two workloads, the workload on all Wikimedia foundation's projects including Wikipedia in Paper IV, and a Video-on-Demand workload in Paper VI.

2.3 Thesis Contributions

In support of our thesis statement, this thesis makes the following contributions towards a better understanding of workloads and elasticity controllers.

I. A comparison between all possible approaches to design a hybrid auto-scaler using a mix of reactive and proactive components for scaling. By a reactive component we mean a component that changes the future capacity using a set of predefined load thresholds, e.g., for every increase of 10 requests/second increase in the arrival rate, provision new resources. By a proactive we mean, a component capable of predicting the future load based on the patterns in the workload history. An auto-scaler was designed that predicts the future workload using the slope of the workload. The controller is further enhanced to take into account the queuing effects resulting from delayed requests.

II. Two comprehensive studies of two server system workloads. The first study is the longest server workload study we are aware of. In this study, we analyze the load on the Wikimedia foundation's servers that host among other things Wikipedia, Wikibooks and Wikitionary. The second is a study of the workload of TV4, a major Swedish Video-on-Demand (VoD) provider. These workloads are studied to provide us with a better understanding of how server systems evolve over time and thus enable us to design better auto-scalers.

III. A method to characterize and compare burstiness, a workload characteristic that affects the performance of an auto-scaler.

IV. Two comparative studies of some of the auto-scalers proposed in the literature. The first study uses scenario theory and chance constrained programming to provide some formal theoretical guarantees on the performance of different auto-scalers with different workloads. The second study compares the performance of a set of the published auto-scalers in a practical setting with different workloads.

V. A Workload Analysis and Classification tool for cloud infrastructures that acts as a meta-controller to select the most suitable auto-scaling algorithm for a workload. The tool analyzes the history of the running/new workloads and extracts some key characteristics. Workloads are then classified and assigned to the most suitable available elasticity auto-scaler based on the extracted characteristics and a set of user-defined Business-Level-Objectives (BLO), reducing the risk of bad predictions and improving the provided QoS.

3 Paper Summary and Future Work

This thesis consist of 7 papers and an appendix that support the thesis position. While most of this work was done assuming an infrastructure-as-a-Service cloud delivery model [8], the algorithms and techniques developed are suitable for all cloud delivery models. The contributions of each paper follows.

3.1 Paper I

The first paper in the thesis [7] introduces two proactive elasticity algorithms that can be used to predict future workload for an application running in the cloud. Resources are then provisioned according to the controllers' predictions. The first algorithm predicts the future load based on the workload's rate of change with respect to time. The second algorithm predicts future load based on the rate of change of the workload with respect to the average provisioned capacity. The two auto-scaling algorithms are explained and tested.

The paper also discusses the nine possible approaches to build hybrid elasticity controllers that have a reactive elasticity component and a proactive component. The reactive elasticity component is a step controller that reacts to the changes in the workload after they occur. The proactive component is a controller that with a prediction mechanism to predict future load based on the load's history. The two introduced controllers are used as the proactive component in the nine approaches discussed. Evaluation is performed using webserver traces. The performance of the resulting hybrid controllers are compared and analyzed. Best practices in designing hybrid controllers are discussed. The performance of the top performing hybrid controllers is compared to a state-of-the-art hybrid elasticity controller that uses a different proactive component. In addition, the effect of the workload size on the performance of the proposed controllers is evaluated. The proposed controller is able to reduce Service-Level-Agreement (SLA) violations by a factor of 2 to 10 compared to the state-of-the-art controller or a completely reactive controller.

3.2 Paper II

The design of the algorithms proposed in the first paper include some simplifying assumptions that ignore multiple important aspects of the cloud infrastructure and the workload's served. Aspects such as VM startup time, workload heterogeneity, and the changing request service rate of a VM are not considered in the first paper. In addition, it is assumed that delayed requests are dropped.

Paper II [4], extends on the first paper by enhancing the cloud model used for the controller design. The new model uses a G/G/N queuing model, where N is variable, to model a cloud service provider. The queuing model is used to design an enhanced hybrid elasticity controller that takes into account workload heterogeneity and the changing request service rate of a VM. The new controller is suitable for applications where buffering of delayed requests is possible. The controller also takes into account the size of the delayed requests when predicting the amount of resources required for the future load. The designed controller's performance is evaluated using webserver traces and traces from a cloud computing cluster with long running jobs. The results are compared to a controller that only has reactive components. The results show that the proposed controller reduces the cost of underprovisioning compared to the reactive controller at the cost of using more resources. The proposed controller requires a smaller buffer to keep all requests if delayed requests are not dropped.

3.3 Paper III

The third paper presents an analysis of a trace from the Wikimedia foundation spanning the period between May, 2008 till October, 2013 [17]. This is the largest web-workload analysis study we are aware of. Using descriptive statistics, time-series analysis, and polynomial splines, we study the trend and seasonality of the workload, its evolution over the years, and investigate patterns in page popularity. We show the effects that spikes on a Wikipedia page have on related Wikipedia pages. In addition, we show that the workload is highly predictable with a strong seasonality. We develop a short term prediction algorithm using splines that is able to predict the workload with a Mean Absolute Percentage Error of around 2%.

3.4 Paper IV

Sample Entropy (SampEn) is a technique that has been used in biomedical systems research for more than a decade to quantify disorder in biomedical signals. Our fourth paper introduces a modified version of the SampEn algorithm as a measure of burstiness in cloud workloads [6]. SampEn is has two main parameters. The first parameter defines how tolerant the measure is to small bursts. The second parameter defines the subset of the workload in which the algorithm searches for similarities. We show that SampEn ordering of workloads is robust against a wide selection of these two parameters. We compared the proposed algorithm to some of the state-of-the-art burstiness measures and show that the modified version of SampEn is better suited for classifying burstiness.

3.5 Paper V

Since Video-on-Demand (VoD) and video sharing services account for a large percentage of the total downstream Internet traffic and cloud applications, we analyze and model a workload trace from a VoD service provided by a major Swedish TV broadcaster [5]. The trace contains over half a million requests generated by more than 20000 unique users. We show that the user and the session arrival rates for the TV4 workload do not follow a Poisson process. The arrival rate distribution is modeled using a log-normal distribution while the inter-arrival time distribution is modeled using a stretched exponential distribution. We observe the "impatient user" behavior where users abandon streaming sessions after minutes or even seconds of starting them. Both very popular videos and non-popular videos are specially affected by impatient users. This behavior seems to be an invariant in VoD workloads and is neither affected by the average bit-rate nor by the number of videos a user watches. We discuss the spikes in the workload and their cause and show that spikes are very hard to predict in many cases.

3.6 Paper VI

In Paper VI, we introduce a method able to compare the performance of auto-scaling algorithms and provide probabilistic guarantees on their performance. The method is based on both robust control theory and stochastic control theory. The evaluation is formulated as a *chance constrained optimization problem*, which is solved using *scenario theory*. The adoption of such a technique allows one to give probabilistic guarantees on the obtainable performance of the controllers in the presence of uncertainties. Six different state-of-the-art auto-scaling algorithms have been selected from the literature for extensive test evaluation, and compared using the proposed framework. We build a discrete event simulator and parameterize it based on real experiments. Using the simulator, each auto-scaler's performance is evaluated using 796 distinct real workload traces from projects hosted on the Wikimedia foundations' servers, and their performance is compared. The evaluation is carried out using different performance metrics, highlighting the flexibility of the framework, while providing probabilistic bounds on the evaluation and the performance of the algorithms. Our results highlight the problem of generalizing the conclusions of the original published studies.

3.7 Paper VII

As it is not possible to design an autoscaler with good performance for all workloads and all scenarios as discussed in Paper VI, Paper VII proposes WAC, a Workload Analysis and Classification tool for automatic selection of a number of implemented cloud auto-scaling methods. . The tool has two main components, the analyzer and the classifier. The analyzer extracts two key features from the workload, periodicity and burstiness. Periodicity is measured using the autocorrelation of the workload, one of the standard methods for measuring periodicity. Burstiness is measured using the modified SampEn algorithm proposed in Paper IV. The classifier component uses a k-Nearest-Neighbors (kNN) algorithm to assign a workload to the most suitable elasticity controller based on the results from the analysis. The classifier requires training using training data. Four different training datasets are used for the training. The first set consists of 14 real workloads, the second set consists of 55 synthetic workloads. The third set consists of the previous two sets combined. The last set of workloads consists of a 798 workloads to different Wikimedia foundation projects. The paper then describes the training of the classifier component and the classification accuracy and results. The results show that the tool is able to assign between 87% to 98.3% of the workloads to the most suitable controller.

4 Future Work

There are several directions identified for future work starting from this thesis, some of which already started while others are planned. We are working on identifying and defining the different main workload classes in a better way and design optimal or

near-optimal auto-scaling methods for each class using stochastic and robust control theory. We are developing a statistical framework that provides probabilistic guarantees on the performance of auto-scaling algorithms based on their error models. We are working on a better statistical model for request arrival rates other than the Poisson process model. In addition, we are working on finding a bounded entropy measure for burstiness or disorder in signals that is able to provide a more solid mathematical definition than Sample Entropy.

References

[1] Page view statistics for Wikimedia projects. URL: dumps.wikimedia.org/other/pagecounts-raw/.

[2] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ACM SIGOPS Operating Systems Review*, pages 2–13. ACM, 2006.

[3] Ahmad Al-Shishtawy and Vladimir Vlassov. ElastMan: Autonomic elasticity manager for cloud-based key-value stores. In *Proc. 22nd Int. Symposium on High-performance Parallel and Distributed Computing*, HPDC 13, pages 115–116, 2013.

[4] Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, and Erik Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd Workshop on Scientific Cloud Computing Date*, ScienceCloud '12, pages 31–40, New York, NY, USA, 2012. ACM.

[5] Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, and Erik Elmroth. Analysis and characterization of a video-on-demand service workload. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 189–200, New York, NY, USA, 2015. ACM.

[6] Ahmed Ali-Eldin, Oleg Seleznjev, Sara Sjöstedt-de Luna, Johan Tordsson, and Erik Elmroth. Measuring cloud workload burstiness. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, pages 566–572, Washington, DC, USA, 2014. IEEE Computer Society.

[7] Ahmed Ali-Eldin, Johan Tordsson, and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *IEEE Network Operations and Management Symposium*, NOMS 12, pages 204–212, April 2012.

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

[9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[10] Jeff Barr, J Varia, and M Wood. Amazon ec22 beta. Amazon Web Services Blog. Accessed March, 2015, 2006. Link:`https://aws.amazon.com/blogs/aws/amazon_ec2_beta/`.

[11] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[12] Peter Bodik, Armando Fox, Michael J Franklin, Michael I Jordan, and David A Patterson. Characterizing, modeling, and generating workload spikes for stateful services. In *ACM SoCC*, pages 241–252. ACM, 2010.

[13] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic resource allocation for shared data centers using online measurements. In *Quality of ServiceâIWQoS 2003*, pages 381–398. Springer, 2003.

[14] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat, and Ronald P Doyle. Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 103–116. ACM, 2001.

[15] Paolo Costa, Thomas Zahn, Ant Rowstron, Greg O'Shea, and Simon Schubert. Why should we integrate services, servers, and networking in a data center? In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, pages 111–118, New York, NY, USA, 2009. ACM.

[16] CycleComputing. New CycleCloud HPC Cluster Is a Triple Threat, September 2011. http://blog.cyclecomputing.com/2011/09/new-cyclecloud-cluster-is-a-triple-threat-30000-cores-massive-spot-instances-grill-chef-monitoring-g.html.

[17] Ahmed Ali Eldin, Ali Rezaie, Amardeep Mehta, Stanislav Razroev, Sara SjoÌstedt-de Luna, Oleg Seleznjev, Johan Tordsson, and Erik Elmroth. How will your workload look like in 6 years? analyzing wikimedia's workload. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 349–354. IEEE, 2014.

[18] Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-efficient server clusters. In *Power-Aware Computer Systems*, pages 179–197. Springer, 2003.

[19] Dror Feitelson. *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[20] Hector Fernandez, Guillaume Pierre, and Thilo Kielmann. Autoscaling web applications in heterogeneous cloud infrastructures. In *IEEE Int. Conf. on Cloud Engineering*, IC2E 14, 2014.

[21] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, 11(2):115–128, 1997.

[22] Alessio Gambi, Mauro Pezze, and Giovanni Toffetti. Kriging-based self-adaptive cloud controllers. *Services Computing, IEEE Transactions on*, PP(99):1–1, 2015.

[23] Anshul Gandhi, Parijat Dube, Alexei Karve, Andrzej Kochut, and Li Zhang. Adaptive, model-driven autoscaling for cloud applications. In *Proc. 11th Int. Conf. on Autonomic Computing*, ICAC 14, pages 57–64, 2014.

[24] Anshul Gandhi, Mor Harchol-Balter, Ram Raghunathan, and Michael A. Kozuch. AutoScale: Dynamic, robust capacity management for multi-tier data centers. *ACM Trans. Comput. Syst.*, 30(4):14:1–14:26, November 2012.

[25] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *Proc. Int. Conf. on Network and Service Management*, CNSM 10, pages 9–16, Oct 2010.

[26] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. In *Proc. 4th ACM/SPEC Int. Conf. on Performance Engineering*, ICPE 13, pages 187–198, 2013.

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

[28] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.*, 28(1):155–162, 2012.

[29] Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proc. 6th Int. Conf. on Autonomic Computing*, ICAC 09, pages 117–126, 2009.

[30] Xiaozhu Kang, Hui Zhang, Guofei Jiang, Haifeng Chen, Xiaoqiao Meng, and Kenji Yoshihira. Understanding internet video sharing site workload: A view from data center design. *Journal of Visual Communication and Image Representation*, 21(2):129–138, 2010.

[31] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[32] Leonard Kleinrock. *Computer applications, volume 2, queueing systems*. Wiley, 1976.

[33] Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. A cloud-assisted design for autonomous driving. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 41–46. ACM, 2012.

[34] Dara Kusic, Jeffrey O Kephart, James E Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, 2009.

[35] Palden Lama and Xiaobo Zhou. Efficient server provisioning with control for end-to-end response time guarantee on multitier clusters. *IEEE Transactions on Parallel and Distributed Systems*, 23(1):78–86, Jan 2012.

[36] Harold C Lim, Shivnath Babu, and Jeffrey S Chase. Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM, 2010.

[37] A. Hasan Mahmud, Yuxiong He, and Shaolei Ren. Bats: Budget-constrained autoscaling for cloud performance optimization. *SIGMETRICS Perform. Eval. Rev.*, 42(1):563–564, June 2014.

[38] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[39] Shicong Meng, Ling Liu, and Vijayaraghavan Soundararajan. Tide: achieving self-scaling in virtualized datacenter management middleware. In *Proc. 11th Int. Middleware Conf. Industrial Track*, Middleware Industrial Track 10, pages 17–22, 2010.

[40] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.

[41] Manfred Morari. Robust stability of systems with integral control. *IEEE Transactions on Automatic Control*, 30(6):574–577, 1985.

[42] Timothy Prickett Morgan. A Rare Peek Into The Massive Scale of AWS. Accessed: August, 2015, http://www.enterprisetech.com/2014/11/14/rare-peek-massive-scale-aws/.

[43] Mark Moshayedi and Patrick Wilkison. Enterprise SSDs. *Queue*, 6(4):32–39, 2008.

[44] Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Sethuraman Subbiah, and John Wilkes. AGILE: Elastic distributed resource scaling for Infrastructure-as-a-Service. In *Proc. 10th Int. Conf. on Autonomic Computing*, ICAC 13, pages 69–82, 2013.

[45] Pradeep Padala, Kai-Yuan Hou, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM, 2009.

[46] Linnie Rawlinson and Nick Hunt. Jackson dies, almost takes internet with him. *CNN, Jun*, 2009. Acessed, March, 2015.

[47] Rackspace Investor Relations. Rackspace Reports Strong Fourth Quarter 2014 Results, Press Release. `http://phx.corporate-ir.net/phoenix.zhtml?c=221673&p=irol-newsArticle&ID=2017418`, February 2015.

[48] Amazon AWS. AWS Case Study: reddit. Accessed: May, 2013, http://aws.amazon.com/solutions/case-studies/reddit/.

[49] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Proc. 2011 IEEE 4th Int. Conf. on Cloud Computing*, CLOUD 11, pages 500–507, 2011.

[50] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: Elastic resource scaling for multi-tenant cloud systems. In *Proc. 2nd ACM Symposium on Cloud Computing*, SOCC 11, pages 5:1–5:14, 2011.

[51] Rahul Singh, Upendra Sharma, Emmanuel Cecchet, and Prashant Shenoy. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proc. 7th Int. Conf. on Autonomic Computing*, ICAC 10, pages 21–30, 2010.

[52] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, and Pawan Goyal. Dynamic provisioning of multi-tier internet applications. In *Proc. 2nd Int. Conf. on Autonomic Computing*, ICAC 05, pages 217–228, 2005.

[53] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3(1):1:1–1:39, 2008.

[54] Luis M Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[55] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[56] Qingyang Wang, Yasuhiko Kanemasa, Jack Li, Chien-An Lai, Chien-An Cho, Yuji Nomura, and Calton Pu. Lightning in the cloud: A study of transient bottlenecks on n-tier web application performance. In *Conference on Timely Results in Operating Systems (TRIOS)*, Broomfield, CO, October 2014. USENIX Association.