Capacity Scaling for Elastic Compute Clouds

Ahmed Aleyeldin (Ali-Eldin) Hassan

LICENTIATE THESIS, JUNE 2013
DEPARTMENT OF COMPUTING SCIENCE
UMEÅ UNIVERSITY
SWEDEN

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

ahmeda@cs.umu.se

ISBN 978-91-7459-688-5
ISSN 0348-0542
UMINF 13.14

Printed by Print & Media, Umeå University, 2013

Abstract

Cloud computing is a computing model that allows better management, higher utilization and reduced operating costs for datacenters while providing on demand resource provisioning for different customers. Data centers are often enormous in size and complexity. In order to fully realize the cloud computing model, efficient cloud management software systems that can deal with the datacenter size and complexity need to be designed and built.

This thesis studies automated cloud elasticity management, one of the main and crucial 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 introduces algorithms, techniques and tools that a cloud provider can use to automate dynamic resource provisioning allowing the provider to better manage the datacenter resources. We design two automated elasticity algorithms for cloud infrastructures that predict the future load for an application running on the cloud. It is assumed that a request is either serviced or dropped after one time unit, that all requests are homogeneous and that it takes one time unit to add or remove resources. We discuss the different design approaches for elasticity controllers and evaluate our algorithms using real workload traces. We compare the performance of our algorithms with a state-of-the-art controller. We extend on the design of the best performing controller out of our two controllers and drop the assumptions made during the first design. The controller is evaluated with a set of different real workloads.

All controllers are designed using certain assumptions on the underlying system model and operating conditions. This limits a controller's performance if the model or operating conditions change. With this as a starting point, 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.

Preface

This thesis consists of a brief introduction to the field, a short discussion of the main problems studied, and the following papers.

Paper I	Ahmed Ali-Eldin, Johan Tordsson and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, pages 204-212. IEEE, 2012.
Paper II	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 (ScienceCloud 2012)*, pages 31–40. ACM, 2012.
Paper III	Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth and Maria Kihl. Workload Classification for Efficient Cloud Infrastructure Elasticity Control. *Technical Report, UMINF 13.13*, Department of Computing Science, Umeå University, Sweden, 2013.

Financial support has been provided in part by the European Community's Seventh Framework Programme OPTIMIS project under grant agreement #257115, the Swedish Government's strategic effort eSSENCE and the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control.

Acknowledgements

This work would have been impossible if it was not for a number of people to whom I am greatly indebted. First of all, I would like to thank my advisor Erik Elmroth for his endurance, patience, inspiration and great discussions. Erik has created a very unique positive research environment that is rare to find any where else. I would also like to thank my coadvisor Johan Tordsson for the hard work, the great ideas, the long discussions, and the great feedback. The past a few years have been very unique, I got married, a revolution happened back home and I got my first kid. Erik and Johan have been considerate, helpful and supportive. They are not just advisors, they are mentors, teachers, and above all friends. It has been a privilege working with you. Your positive impact will stay with me for the rest of my life.

I would also like to thank Daniel Espling and Wubin Li for their great help with OPTIMIS, Per-Olov Östberg, Peter Gardfjäll and Lars Larsson for the inspiring discussions and the great feedback, Ewnetu Bayuh my office mate for the great time we spend together, Christina Igasto for helping me settle, Petter Svärd, Francisco Hernández, Mina Sedaghat, Selome Kosten, Gonzalo Rodrigo, Cristian Klein, Luis Tomas, Amardeep Mehta, Lei Xu, Tomas Forsman, Emad Hassan for the great time we spend together.

I would like to thank Maria Kihl at Lund university for the interesting collaboration, positive feedback and inspiring discussions. Four years ago, when I started my postgraduate studies, I met Sameh El-Ansary who hired me as a research assistant. He taught me a lot about research. He was a great mentor and now he is a dear friend !

On a more personal level, I would like to thank my parents for their love and their support. This work would have not been possible if it was not for them explaining to me maths and physics 24 years ago! I was 4 when they started teaching me the multiplication table. By the time I was five I knew a little bit more than my peers! I love you both and I pray that I will always be a source of happiness to you!

I fell in love with a girl one week before I started my PhD studies. We got married 3 months after I started my PhD. Hebatullah, thank you for being there for me always with love, support and care. I would also like to thank the rest of my family for their love and support. Last but not least, I would like to thank Salma my little daughter. She is the most precious thing I have ever had and the main source of joy in life!

Thank you all !

Contents

Chapter 1

Introduction

The idea of having computing power organized as a utility dates back to the 1960s. In a speech in 1961, John McCarthy predicted that "computation may someday be organized as a public utility" just like electricity and water [20]. His idea did not gain popularity until the late 1990s when research on Grid computing started. The term Grid computing was used to describe technologies that enable on demand usage of computing power [20]. Grid computing has been used mainly for scientific applications within the scientific community and did not gain widespread support outside that community.

Driven originally by economic needs, cloud computing can be considered as an evolution of Grid computing that gained popularity during the last a few years. The cloud computing model aims for the efficient use of datacenter resources by increasing resource utilization and reducing the operating costs while providing on demand computing resources. In contrast to Grid computing, cloud computing has mainly been used for commercial systems with some interest from the scientific community [58]. Most grid systems are used mainly for batch jobs which are very common for scientific applications. Clouds on the other hand support the deployment of more application types including webservers, data-processing applications and batch systems.

There is no agreement on how to define cloud computing [9, 13, 20]. The definition used in this thesis is aligned with the NIST definition [31] which describes cloud computing as a resource utilization model that enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources. The computing resources can be rapidly provisioned and released with minimal management effort or service provider interaction.

There are many technologies that contributed directly to the possibility of building cloud systems. Advances in virtualization technologies [1, 10], advances in server power management techniques [17] and increased network bandwidth are the main enabling technologies for cloud computing.

Customers lease resources from a cloud service provider to run their applications, services, computations or store their data on the leased resources.

Often, the resources are used to deploy a web-based service accessible by other entities, e.g, Reddit runs a social news and entertainment service on Amazon [37]. For the rest of this work, we interchangeably use the terms 'service' and 'application' to describe the customer's use of cloud resources.

1.1 Cloud computing characteristics

There are five essential characteristics of cloud computing identified by NIST [31]. These five characteristics can be summarized as follow:

1. On demand provisioning of resources requiring no human interaction with the service provider.
2. Broad network access able to handle users of different network clients such as, mobile phones and workstations.
3. Resource pooling between multiple customers having different resource requirements. The cloud is abstract to the costumers generally. Customers have no control or knowledge over the exact location of their resources except at a higher level of abstraction.
4. Rapid elasticity which is the ability to vary allocated capacity depending on the load, sometimes automatically. The resources available for provisioning often appear unlimited for the service user.
5. Transparent resource usage monitoring where running applications are actively monitored. Usage and performance reports should be reported transparently.

We add to the previous list two more characteristics that we believe are essential for the cloud service model.

1. Fault tolerance is important for cloud platforms since cloud platforms are built using commodity hardware [11]. The cloud provider should provide transparent fault tolerance mechanisms that mask failures from the customers.
2. The cloud provider has to provide Quality-of-Service (QoS) guarantees for the customers. The QoS guarantees should be at least similar to the guarantees

Although these two characteristics are not unique to clouds, they are essential for the realization of the cloud computing model. Full realization of the cloud model is still far from reality [36, 18, 54]. Limitations in the current cloud offerings require more research in order to fill the gaps between the cloud model and the reality. This thesis contributes to better management of cloud systems, thus it contributes to filling this gap.

1.2 Cloud Computing models

Cloud systems can be categorized based on their service models or deployment models. Service models describe the type of service offered by the cloud provider. Deployment models describe the way a service running on the cloud is deployed on the actual infrastructure.

1.2.1 Cloud Computing service models

Cloud computing systems are often classified into three main service models, namely:

1. The Infrastructure-as-a-Service (IaaS) model, where the service provider leases raw computing resources. These resources can be either physical (sometimes referred to as Hardware-as-a-Service) or virtual [39]. This model enables the cloud service user to specify the needs of an application in terms of raw computing resources. The cloud user is responsible for deploying and configuring the operating systems, the applications and any required software packages to run on the infrastructure. The user does not control the cloud infrastructure, e.g., the user has no control over where the assigned resources are in a datacenter. Amazon EC2 [39] and RackSpace [45] are two examples of IaaS platforms.

2. The Platform-as-a-Service (PaaS) model, where the service provider offers a platform that supports certain programming languages, libraries, services, and tools. The cloud service user can use the platform to develop and/or deploy applications using the provided tools and/or environments. The user does not manage the operating system, the underlying hardware or the software packages supported by the PaaS provider. Google's App Engine [42] and Windows Azure [43] are two examples of PaaS platforms.

3. The Software-as-a-Service (SaaS) model, where the service provider offers an application running on a cloud infrastructure to the customers. The application is used by the cloud service user as is. Oracle Cloud [44] and Salesforce [47] are two examples of SaaS providers.

These models are not mutually exclusive. They can coexist in the same datacenter. Some IaaS providers host Paas and SaaS Clouds in an IaaS cloud [38].

1.2.2 Cloud Computing deployment models

Cloud deployment models describe how and where services running in a cloud are deployed. It also describes who can access the resources of a cloud. The deployment models identified by NIST are:

1. Private clouds: Owned, managed and used by a single organization. Access to the cloud is restricted to entities within the organization. This

model provides higher security for the organization since all sensitive data is kept internal. The National Security Agency in the USA recently revealed that they are operating a private cloud [12].

2. Community clouds: shared between multiple organizations having common interests. Access to the cloud is restricted to entities within the organizations. The G-Cloud project in the UK started as a community cloud [41].

3. Public clouds are infrastructures that lease computing resources to the public. They are typically operated and managed by business or academic organizations. The cloud resources are shared between the customers. Amazon's EC2, RackSpace and Salesforce are all examples of public clouds.

4. Hybrid clouds describes distinct cloud systems (public, private or community) that have mutual agreements and technologies enabling data and application portability. The model allows one cloud entity to extend its capacity by using resources from another cloud entity. In addition, the model allows load balancing between the partner clouds. The recently announced hybrid cloud between Netapp, Equinix and Amazon [40] is an example of this deployment model.

1.3 Cloud Middlewares

Cloud middlewares are software systems used to manage cloud computing systems. These middlewares should be designed to provide the essential characteristics of the cloud computing model. Cloud middlewares should provide APIs and abstraction layers through which the customer can submit requests for resources and monitor resource usage. The middlewares need to manage admission control, resource pooling, fault tolerance, on demand provisioning, resource placement, and possibly automatic elasticity. In addition, there should be enough hardware resources to enable efficient resource pooling and rapid elasticity. The middleware is also responsible for enforcing all QoS guarantees. The service and deployment models supported by the cloud also affects the middleware design.

Current cloud datacenters are huge in size. For example, Rackspace has more than 94000 servers hosted in 9 datacenters serving more than 200000 customers. Typically, a 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 [24]. Fault tolerance should be transparent to the user and the applications with minimal effect on the QoS of the service. Resource usage monitoring should be done frequently and transparently. The main focus of this thesis is the design of algorithms that enable the automation of cloud middlewares with an emphasis on algorithms for automated rapid elasticity.

Chapter 2

Rapid Elasticity: Cloud Capacity Auto-Scaling

NIST describes *rapid elasticity* as a cloud essential characteristic that enables capabilities to be elastically provisioned and released, to scale rapidly according to applications' demand. To the cloud user, the resources available often appear unlimited. The NIST definition does not require elasticity to be automated although it can be automated in some cases [31]. Elasticity control can be divided in to two classes, namely, horizontal elasticity and vertical elasticity. Horizontal elasticity is the ability of the cloud to rapidly vary the number of VMs allocated to a service according to demand [4]. Vertical elasticity is the ability of the cloud to rapidly change configurations of virtual resources allocated to a service to vary with demand, e.g., adding more CPU power, memory or disk space to already running VMs [57]. This thesis focuses on automated horizontal elasticity or resource auto-scaling and the challenges associated with the automation.

2.1 Elasticity Controller Requirements

Considered by some as the game-changing characteristic of cloud computing [34], elasticity has gained considerable research interest [7, 28, 32, 50, 55]. Most of the research on cloud elasticity has focused on the design of automated elasticity controllers. We believe there are essential characteristics for automated elasticity [3] controllers to be useful, namely,

1. Scalability: It has been estimated recently that Amazon's EC2 operates around half a million servers [30]. One single service can have up to a few thousand machines [15]. Some services run on the cloud for a short period of time while others services can run for years entirely on the cloud, e.g., reddit has been running on EC2 entirely since 2009 [37]. 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.

2. Adaptiveness: Workloads of Internet and cloud applications are dynamic [23, 5]. 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.

3. Rapid: An automated elasticity algorithm should compute the required capacity rapidly enough to preserve the QoS requirements. Sub-optimal decisions that preserves the QoS requirements are better than optimal decisions that might take longer to process than can be tolerated. 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) [26].

4. Robustness: The changing load dynamics might lead to a change in the controller behavior [33]. A robust controller should prevent oscillations in resource allocation. While adaptiveness describes the ability of the controller to adapt to changing workloads, robustness describes the behavior of the controller with respect to its stability. A controller might be able to adapt to the workload but with oscillations in resource allocation that results in system instability. Another controller might not be able to adapt to the changing workload, but is stable with changing workload or system dynamics.

5. 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 resources less 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.

We note that adaptivity and scalability were also identified by Padala et. al. [35] as design goals for their controller design.

2.2 Cloud Middlewares and Elasticity Aspects

Cloud middlewares typically have different components managing different functionalities such as elasticity, resource and data placement, security, monitoring, admission control, accounting and billing. We discuss the effect of having an automated elasticity component on different middleware components.

2.2.1 Monitoring

Monitoring is an integral part of datacenter management. Almost all components of a cloud middleware are dependent on the presence of reliable monitoring data. Monitoring of cloud services has been identified as a significant challenge to cloud systems adoption [20].

Elasticity algorithms are dependent on the available monitoring data since the data is used to calculate and predict the current and future load based on the monitored demand. There are significant challenges when the automated elasticity component is managed by the cloud provider. Different services have different measures of performance, e.g., response time, CPU utilization, memory utilization, network bandwidth utilization, request arrival rates or any specific metric for that particular service. The monitoring component should have the ability to monitor different metrics, including application specific metrics, for different services running on the cloud.

2.2.2 Placement

Resource placement and elasticity are two complementary problems that highly affect each other. The placement problem is concerned with the assignment of actual hardware in the datacenter to a service, i.e., where the VMs of a service run [27]. When the required capacity is predicted by an elasticity algorithm and new VMs are to be deployed, the placement component chooses where should these VMs run in the datacenter. Similarly, when the decision is to remove some VMs allocated to a service, the placement component is responsible for choosing which VMs to remove. Therefore, intelligent placement algorithms are required to make sure that the QoS does not deteriorate due to bad placement decisions, e.g., placing a VM on an already overloaded physical machine.

2.2.3 Security

Almost all the security threats to traditional online systems are present for services running on a cloud. Automatic elasticity adds a new dimensionality to Denial-of-Service attacks (DoS). DoS attacks are usually performed by saturating the servers of a service by bogus requests. In traditional online systems, when the servers are saturated, the service responsiveness deteriorates and the service may crash. In cloud environments having automated elasticity, if such attacks are not discovered early on, additional resources are added to the service to serve the bogus requests. These resources are paid for while not doing any actual work resulting in an economical Denial of Service attack [2, 25, 51].

2.2.4 Admission Control

Admission control is the process concerned with accepting new customer services. Admission control mechanisms aim to keep the cloud infrastructure highly utilized while avoiding overloading that may results in QoS deterioration. Admission control can be easily done when all the deployed services are static and no changes occur in the amount of resources allocated to any of the services. On the other hand, for elastic applications, admission control becomes more complex since the admission control mechanism has to take into account the current and predicted future load for all services running on the cloud [56]. Careful resource overbooking can increase the profit of a cloud provider but it requires accurate elasticity predictions [52].

2.2.5 Elasticity and Accounting

Since the amount of resources allocated to a service change dynamically, the accounting component must be designed to handle these volatile resource usages [16]. Current Cloud providers typically charge for resources in billing cycles of length one hour each [48, 46]. For a public cloud, An automated elasticity algorithm should be aware of the billing cycle length. Removing a VM before the end of its billing cycle is a waste of resources since the price for that VM is already paid.

2.3 Thesis Contributions

This research on automated elasticity algorithms extends on the research done on dynamic resource provisioning that started more than a decade ago [6, 14]. Designing an automated elasticity controller that meets the desired requirements for a wide spectrum of applications and workloads is not an easy task. Most of the proposed elasticity controllers lack at least one of the identified properties. Some elasticity controller designs assume a certain model for the infrastructure and certain operating conditions [8, 28]. These controllers lack robustness against changes in the infrastructure and changes in workload dynamics. Other controller designs are not scalable with respect to time [21]. Yet other designs are not scalable with respect to the amount of resources allocated to a service [26, 49]. Some of the proposed solutions do not take into account costs associated with dropped requests [29] or neglects overprovisioning costs by not scaling down the extra resources [53]. Almost all the controllers proposed in the literature we are aware off were evaluated with less than three real workloads [29], typically one or less [21, 22, 49, 53] sometimes for a period equivalent to less than a day [21].

The first contribution of this thesis is the design of two automated adaptive hybrid elasticity controller that uses the slope of a workload to predict its future values [4, 19]. The controller's design is further extended and evaluated with additional workloads of different natures [3]. Since no controller is able to have good performance on all different workloads, our second contribution is a workload analysis and classification tool that assigns a workload to the most suitable controller out of a set of implemented elasticity controllers [5].

The assignment is calculated based on the workload characteristics and service level objectives defined by the cloud customer. Thesis contributions include scientific publications addressing the design of algorithms for cloud capacity auto-scaling and the design and implementation of a tool for workload analysis and classification. In addition, software artifacts using the proposed algorithms for auto-scaling were developed.

Chapter 3

Paper Summary and Future Work

As previously stated, the main focus of this thesis is the design and implementation of algorithms that enable the automation of cloud middlewares with an emphasis on algorithms for auto-scaling. Although all our publications assume that the middleware is for an IaaS public or private cloud, the algorithms and techniques developed are suitable for all cloud models.

3.1 Paper I

The first paper in the thesis [4] introduces two proactive elasticity algorithms that can be used to predict future workload for an application running on 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 designs of the two algorithms are explained.

The paper also discusses the nine 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 has 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 done using webserver traces. The performances 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 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, [3] 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 the VM startup time, workload heterogeneity and the changing request service rate of a VM. The new controller allows the buffering of delayed requests and 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 extends on the first two papers. The performance of the designed controllers in the first two papers varies with different workloads due to different workload characteristics. Paper III discusses the effect of different workload characteristics on the performance of different elasticity controllers. The design and implementation of an automatic workload analysis and classification tool is proposed as a solution to the performance variations. The tool can be used by cloud providers to assign workloads to elasticity controllers based on the workloads' characteristics. The tool has two main components, the analyzer and the classifier.

The analyzer analyzes a workload and extracts it periodicity and burstiness. Periodicity is measured using the autocorrelation of the workload since autocorrelation is a standard method to measure the periodicity of a workload. The use of Sample Entropy (SampEn) as a measure of burstiness is proposed. SampEn has been used in biomedical systems research for more than a decade and has proven robust. To the best of our knowledge, this is the first paper proposing SampEn usage for characterizing bursts in cloud computing workloads. 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. Three different training datasets are used for the training. The first set consists of 14 real workloads, the second set consists of 55 synthetic workloads and the third set consists of the previous two sets combined. The analysis results of 14 real workloads are described.

Paper III also proposes a methodology to compare the performance of an application's workload on different elasticity controllers based on a set of predefined business level objectives by the application's owner. The performance of the training set is the workloads in the training set is discussed using the proposed method. 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 92% and 98.3% of the workloads to the best suitable controller.

3.4 Future Work

There are several directions identified for future work starting from this thesis, some of which already started while others are planned. The design of more efficient cloud management systems depends on better understanding of the workloads running on the cloud systems. The workload analysis and classification component proposed in Paper III has used only two characteristics to analyze the different workloads. We have currently started investigating what other additional characteristics can be used for workload analysis. We plan to use the identified important characteristics to analyze longer workloads to better understand the evolution of a workload with time. Since the available real cloud workloads are scarce, the analysis results will be used to improve the workload generator used in Paper III to generate synthetic traces. requires The algorithms presented in Paper I and Paper II are useful for short term predictions. The proposed algorithms do not predict long term capacity requirements. Predictions of long term capacity requirements are important for different reasons such as admission control of new services and resource placement. Accurate admission controllers require some knowledge about the predicted aggregate load on the infrastructure in order to preserve QoS guarantees for the running services. Since resources are typically consolidated in a cloud, the middleware should consolidate orthogonal loads on the same physical machine in order to preserve the QoS requirements, e.g., computationally intensive workloads with predicted high peaks in CPU usage should not be consolidated on the same physical server but rather with memory intensive workloads. We are currently working on a design of an elasticity controller that can predict short term and long term capacity requirements with high accuracy. The workload analysis results will also be taken in to consideration for the new controller's design. Resource provisioning should be based on a combination of both the short term and long term predictions.

The workload analysis and classification tool described in Paper III is used to assign workloads to elasticity algorithms. In principle, the tool can be used to assign workloads to a group of predefined classes in general, e.g., elasticity algorithms, placement algorithms or even different computing environments. We plan to extend and adapt the current tool to cover different use-cases.

Bibliography

[1] 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.

[2] Fahd Al-Haidari, Mohammed H Sqalli, and Khaled Salah. Enhanced EDoS-shield for mitigating EDoS attacks originating from spoofed IP addresses. In *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2012*, pages 1167–1174. IEEE, 2012.

[3] 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*, pages 31–40. ACM, 2012.

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

[5] Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth, and Maria Kihl. Workload classification for efficient cloud infrastructure elasticity control. Technical report, UMINF 13.13, Umeå University, 2013.

[6] Guillermo A Alvarez, Elizabeth Borowsky, Susie Go, Theodore H Romer, Ralph Becker-Szendy, Richard Golding, Arif Merchant, Mirjana Spasojevic, Alistair Veitch, and John Wilkes. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems (TOCS)*, 19(4):483–518, 2001.

[7] Ganesh Ananthanarayanan, Christopher Douglas, Raghu Ramakrishnan, Sriram Rao, and Ion Stoica. True elasticity in multi-tenant data-intensive compute clusters. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 24. ACM, 2012.

[8] Ala Arman, Ahmad Al-Shishtawy, and Vladimir Vlassov. Elasticity controller for cloud-based key-value stores. In *Parallel and Distributed Systems*

(ICPADS), 2012 IEEE 18th International Conference on, pages 268–275. IEEE, 2012.

[9] 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. *Communications of the ACM*, 53(4):50–58, 2010.

[10] 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.

[11] Carsten Binnig, Donald Kossmann, Tim Kraska, and Simon Loesing. How is the weather tomorrow?: towards a benchmark for the cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*, page 9. ACM, 2009.

[12] Nathanael Burton. "Keynote: OpenStack at the National Security Agency (NSA)". Accessed: May, 2013, http://www.openstack.org/summit/portland-2013/session-videos/presentation/keynote-openstack-at-the-national-security-agency-nsa.

[13] 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.

[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 *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 103–116. ACM, 2001.

[15] 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.

[16] Erik Elmroth, Fermin Galan Marquez, Daniel Henriksson, and David Perales Ferrera. Accounting and billing for federated cloud infrastructures. In *Eighth International Conference on Grid and Cooperative Computing, 2009. GCC'09.*, pages 268–275. IEEE, 2009.

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

[18] Benjamin Farley, Ari Juels, Venkatanathan Varadarajan, Thomas Ristenpart, Kevin D Bowers, and Michael M Swift. More for your money: Exploiting performance heterogeneity in public clouds. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 20. ACM, 2012.

[19] Ana Juan Ferrer, Francisco Hernandez, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, Raul Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, Nikolaus Forgo, Tabassum Sharif, and Craig Sheridan. Optimis: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1):66 – 77, 2012.

[20] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. IEEE, 2008.

[21] 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.

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

[23] 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.

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

[25] Soon Hin Khor and Akihiro Nakao. spow: On-demand cloud-based eddos mitigation mechanism. In *Fifth Workshop on Hot Topics in System Dependability*, 2009.

[26] 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.

[27] Wubin Li, Johan Tordsson, and Erik Elmroth. Virtual machine placement for predictable and time-constrained peak loads. In *Economics of Grids, Clouds, Systems, and Services*, pages 120–134. Springer Berlin Heidelberg, 2012.

[28] 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.

[29] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. Online dynamic capacity provisioning in data centers. In *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2011*, pages 1159–1163. IEEE, 2011.

[30] Huan Liu. "Amazon data center size". Accessed: May, 2013, http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/.

[31] Peter Mell and Timothy Grance. The NIST definition of cloud computing. *NIST special publication*, 800:145, 2011.

[32] Shicong Meng, Ling Liu, and Vijayaraghavan Soundararajan. Tide: achieving self-scaling in virtualized datacenter management middleware. In *Proceedings of the 11th International Middleware Conference Industrial track*, pages 17–22. ACM, 2010.

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

[34] Dustin Owens. Securing elasticity in the cloud. *Commun. ACM*, 53(6):46–51, June 2010.

[35] 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.

[36] David Price. Is Cloud Computing Still a Dream? Accessed: May, 2013, http://www.pcworld.com/article/221268/Is_Cloud_Computing_Still_a_Dream.html.

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

[38] Amazon Web Service. "Case Studies". Accessed: May, 2013, http://aws.amazon.com/solutions/case-studies/.

[39] Amazon Web Services. "Amazon EC2 instances". Accessed: May, 2013, http://aws.amazon.com/ec2/instance-types/.

[40] Equinix. "Rethink Your Storage Strategy for the Digital Economy". Accessed: May, 2013, http://www.equinix.com/solutions/partner-solutions/netapp/.

[41] G-Cloud. "The G-Cloud Programme". Accessed: May, 2013, http://gcloud.civilservice.gov.uk/.

[42] Google. " google app engine". Accessed: May, 2013, https://developers.google.com/appengine/.

[43] Microsofot. Windows Azure. Accessed: May, 2013, http://www.windowsazure.com/en-us/.

[44] Oracle. "Oracle Cloud". Accessed: May, 2013, https://cloud.oracle.com/mycloud/f?p=service:home:0.

[45] Rackspace. " The Rackspace Cloud". Accessed: May, 2013, http://www.rackspace.com/cloud.

[46] Rackspace. "cloud servers pricing". Accessed: May, 2013, http://www.rackspace.com/cloud/servers/pricing/.

[47] Salesforce. "CRM and Cloud Computing To Grow Your Business". Accessed: May, 2013, http://www.salesforce.com/.

[48] Windows Azure. "pricing at-a-glance". Accessed: May, 2013, http://www.windowsazure.com/en-us/pricing/overview/.

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

[50] Dan Schatzberg, Jonathan Appavoo, Orran Krieger, and Eric Van Hensbergen. Why elasticity matters. Technical report, Boston University, 2012.

[51] Mohammed H Sqalli, Fahd Al-Haidari, and Khaled Salah. EDoS-shield-a two-steps mitigation technique against EDoS attacks in cloud computing. In *Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 2011*, pages 49–56. IEEE, 2011.

[52] Luis Tomas and Johan Tordsson. Improving cloud infrastructure utilization through overbooking. In *The ACM Cloud and Autonomic Computing Conference (CAC 2013), to appear*, 2013.

[53] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1, 2008.

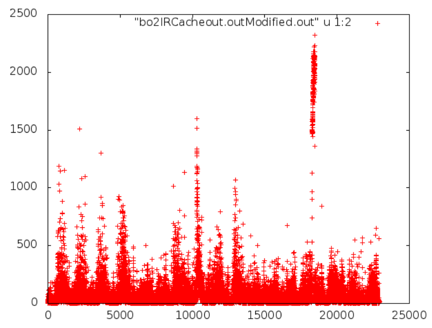
[54] Venkatanathan Varadarajan, Thawan Kooburat, Benjamin Farley, Thomas Ristenpart, and Michael M Swift. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 281–292. ACM, 2012.

[55] David Villegas, Athanasios Antoniou, Seyed Masoud Sadjadi, and Alexandru Iosup. An analysis of provisioning and allocation policies for infrastructure-as-a-service clouds. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2012*, pages 612–619. IEEE, 2012.

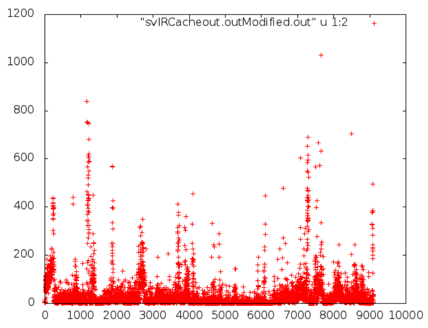
[56] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. Sla-based admission control for a software-as-a-service provider in cloud computing environments. *Journal of Computer and System Sciences*, 78(5):1280–1299, 2012.

[57] Lenar Yazdanov and Christof Fetzer. Vertical scaling for prioritized vms provisioning. In *Second International Conference on Cloud and Green Computing (CGC), 2012*, pages 118–125. IEEE, 2012.

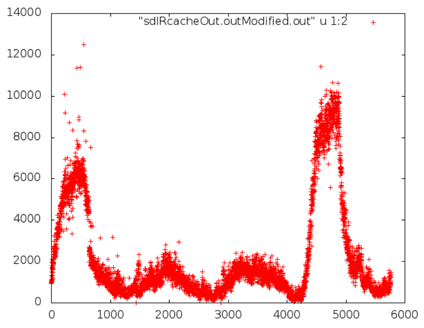
[58] Katherine Yelick, Susan Coghlan, Brent Draney, and Richard Shane Canon. The magellan report on cloud computing for science. Technical report, Technical report, US Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), 2011.



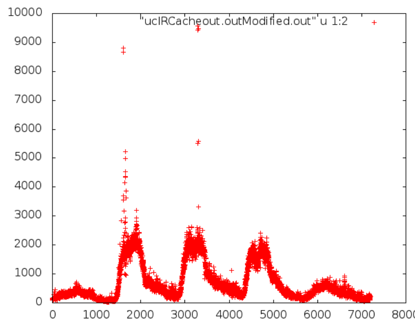
(a) Workload trace for IRCache service runnings ar at Boulder, Colorado (Bo).



(b) Workload trace for IRCache service running at Silicon Valley, California (SV).

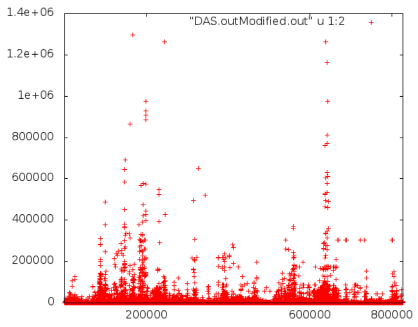


(c) Workload trace for IRCache service running at San Diego, California (SD).

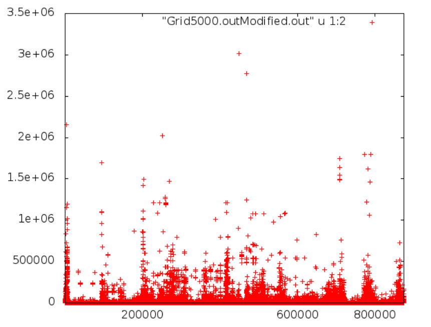


(d) Workload trace for IRCache service running at Urbana-Champaign, Illinois (UC).

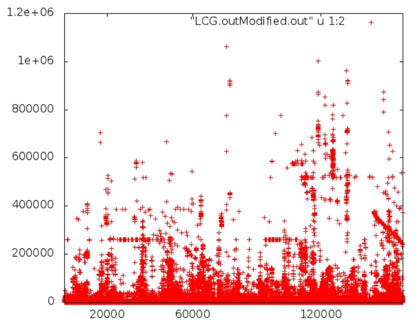
Figure 13: Workload traces of the different Caching Services.



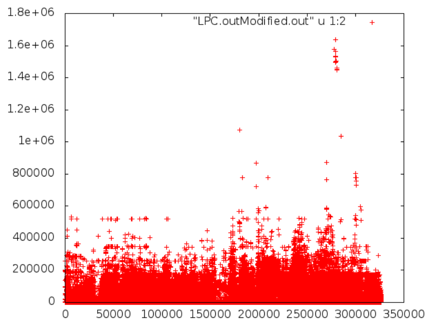
(a) Workload trace for the DAS workload.



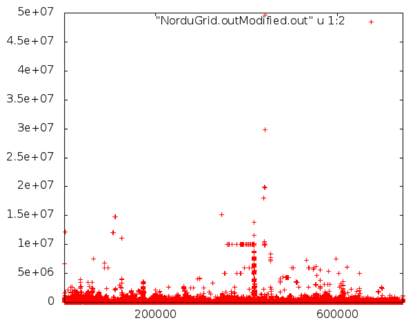
(b) Workload trace for Grid5000 workload.



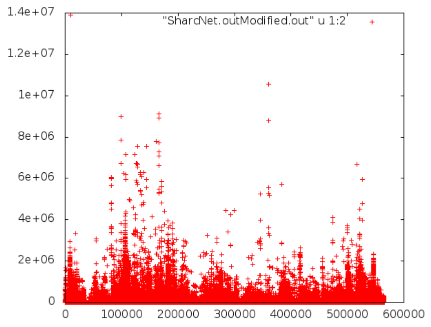
(c) Workload trace for the LCG workload.



(d) Workload trace for LPC workload.

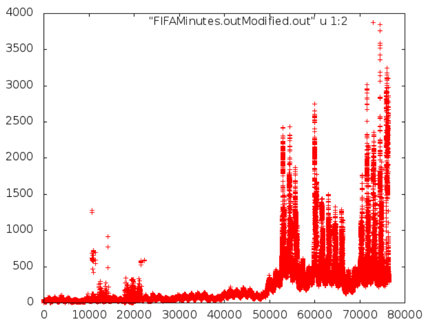


(e) Workload trace for the NorduGrid workload.

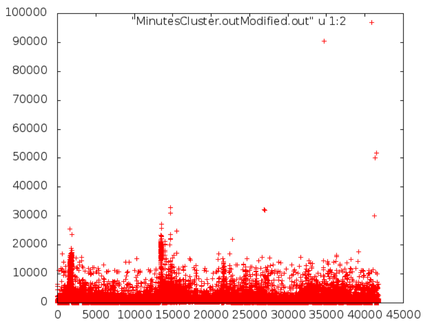


(f) Workload trace for SharcNet workload.

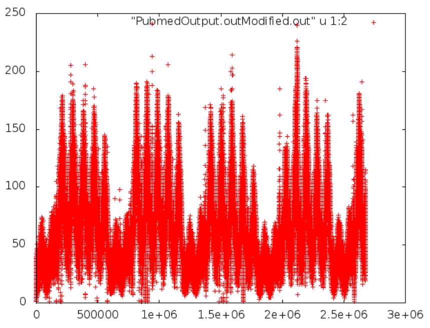
Figure 14: Workload traces of the Grid workloads analyzed.



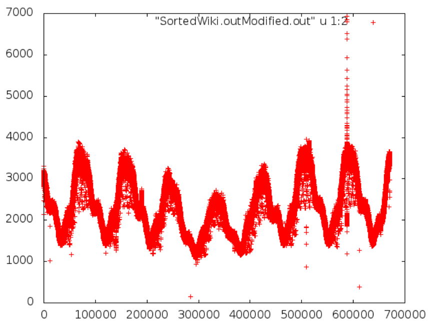
(a) Workload trace for the world cup workload.



(b) Workload trace for the Google Cluster workload.



(c) Workload trace for PubMed access traces.



(d) Workload trace for the Wikipedia workload.

Figure 15: Workload traces of web-hosting workloads and the Google cluster workload analyzed.