Performance Problem Diagnosis in Cloud Infrastructures

Olumuyiwa Ibidunmoye

LICENTIATE THESIS, MAY 2016
DEPARTMENT OF COMPUTING SCIENCE
UMEÅ UNIVERSITY
SWEDEN

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

muyi@cs.umu.se

Copyright © 2016 by the author(s)
Except Paper I, © ACM Press, 2015
Paper II, © ACM Press, 2015

ISBN 978-91-7601-500-1
ISSN 0348-0542
UMINF 16.14

Printed by Print & Media, Umeå University, 2016

Abstract

Cloud datacenters comprise hundreds or thousands of disparate application services, each having stringent performance and availability requirements, sharing a finite set of heterogeneous hardware and software resources. The implication of such complex environment is that the occurrence of performance problems, such as slow application response and unplanned downtimes, has become a norm rather than exception resulting in decreased revenue, damaged reputation, and huge human-effort in diagnosis. Though causes can be as varied as application issues (e.g. bugs), machine-level failures (e.g. faulty server), and operator errors (e.g. mis-configurations), recent studies have attributed capacity-related issues, such as resource shortage and contention, as the cause of most performance problems on the Internet today. As cloud datacenters become increasingly autonomous there is need for automated performance diagnosis systems that can adapt their operation to reflect the changing workload and topology in the infrastructure. In particular, such systems should be able to detect anomalous performance events, uncover manifestations of capacity bottlenecks, localize actual root-cause(s), and possibly suggest or actuate corrections.

This thesis investigates approaches for diagnosing performance problems in cloud infrastructures. We present the outcome of an extensive survey of existing research contributions addressing performance diagnosis in diverse systems domains. We also present models and algorithms for detecting anomalies in real-time application performance and identification of anomalous datacenter resources based on operational metrics and spatial dependency across datacenter components. Empirical evaluations of our approaches shows how they can be used to improve end-user experience, service assurance and support root-cause analysis.

Preface

This thesis contains an introduction to performance management and diagnosis in cloud computing infrastructures, and the following papers.

Paper I Ibidunmoye, O., Hernández-Rodriguez, F., and Elmroth, E. Performance Anomaly Detection and Bottleneck Identification. *ACM Computing Surveys (CSUR): Volume 48, Issue 1, July 2015*.

Paper II Metsch, T., Ibidunmoye, O., Bayon-Molino, V., Butler, J., Hernández-Rodriguez, F., and Elmroth, E. Apex Lake: A Framework for Enabling Smart Orchestration. *Proceedings of the Industrial Track of the 16th International Middleware Conference*, December, 2015.

Paper III Ibidunmoye, O., Metsch, T., Bayon-Molino, V., and Elmroth, E. Performance Anomaly Detection using Datacenter Landscape Graphs. *To be submitted*, 2016.

Paper IV Ibidunmoye, O., Metsch, T., and Elmroth, E. Real-time Detection of Performance Anomalies for Cloud Services. *To be submitted*, 2016.

This work has been funded in part by the Swedish Research Council (VR) under grant agreement C0590801 (Cloud Control), the Swedish Strategic Research Program eSSENCE, and the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS).

Acknowledgments

To **Erik Elmroth**, my supervisor, I express my deepest gratitude for believing in me enough to accept me into his group and most importantly for mentoring, advising and guiding me through the on-going doctoral journey. I also appreciate my former assistant supervisor, **Francisco Hernandez Rodriguez**, for his guidance in the early periods of my study.

I'm really honoured to be part of the Distributed Systems group, how else could I ever have met such resourceful set of people. You guys have and are continuing to impact and challenge me in ways that are simply unquantifiable. I appreciate the senior colleagues (Erik, **Johan**, **P-O**, **Cristian**, and **Luis**) for the conducive environment; the freshly minted doctors (**Mina**, **Ahmed**, and **Ewnetu**) for constantly inspiring me, as well as colleagues still in the race (**Selome**, **Jakub**, **Gonzalo**, **Amardeep**, **Abel**, and **Chanh**) for the good friendship and support base. It's a privilege to have these professional software developers next door; **Lars** and **Peter**, you two have been helpful many a time. And to past members of the group, **Daniel**, **Petter** and **Viali**!, it's encouraging to see you guys do well in your careers and even more assuring to know that you're always there when we beckon. Again, special thanks to Cristian and Ewnetu for the many discussions, deliberation, tools, and guidance.

Many thanks to all staff of the IT-Support unit, especially **Tomas Forsman**, for always helping with technicalities and knotty IT stuff. I also appreciate the support of the Cloud Services Lab, Intel Labs Europe, Ireland; to **Thijs Metsch** I say thanks so much!

To God Almighty, I say a big 'Thank You!' for bringing me this far and for giving me the grace, courage and strength to push forward against all odds. I appreciate my parents and other members of my family for their moral support and good wishes. What could I do without my loving wife, my source of peace and courage in those dark times and in the face of weakness? **Wemimo**, I can't be grateful enough to you for delaying your career and comfort to stick up with me far away from home. Thanks for having my back always.

Umeå, May 2016
Olumuyiwa Ibidunmoye

Contents

Chapter 1

Introduction

The fundamental idea behind cloud computing was first conceived by John McCarthy as far back as 1961. He envisioned that "computation may someday be organized as a public utility" so as to reduce the cost of computing, increase reliability, and efficiency by relieving users from the need to own and operate complex computing infrastructure [1, 2]. Advancement in legacy technologies such as distributed systems, commodity computing, virtualization, and Grid computing has led to the realization of the cloud computing paradigm [3].

As the flexibility and scalability provided by cloud computing continues to improve the agility, responsiveness, and effectiveness of large-scale applications and organizations, meeting reliability, availability, and performance requirements remain a major source of concern. Studies [4, 3] have demonstrated through extensive experiments that performance variability is an important concern for cloud application providers because of its impact on end-user's quality of experience. In severe cases, unstable performance may manifest high-level symptoms such as degraded quality of service (e.g. slow page response) and service downtimes (e.g. unreachable service end-points). The reason for such performance problems is multi-faceted. Unhealthy resource interference due to time-sharing of heterogeneous application services [5], capacity shortages or exhaustions due to unpredictable surge in user traffic [6], and correlated fault in the underlying infrastructure [7, 8] are factors that have been reported to induce undesirable performance behaviour and service outages.

Performance variability and degradation impact the reliability of infrastructure and services and may hinder the ability to meet both service-level and operational objectives. Improving service performance and reliability is thus important since the volume of service traffic and the revenue relates to the quality of service (QoS) experienced by end-users [8]. Here are some concrete examples. The Aberdeen Group [9] found that a one-second increase in page response time can decrease page views, end-user satisfaction, and revenue by 11%, 16%, and 7%, respectively. On June 29th 2010, Amazon.com suffered a 3-hour intermittent performance problems such as high page latency and

incomplete product search results, which led to a loss of $1.75m in revenue per hour [10]. On the 16th of October 2015, several Apple services including iTunes and App Store suffered outages spanning many hours. Other popular global services such as Twitter, Netflix, Instagram, Google services (Drive, Docs, and Sheets), and PayPal also experienced hours of unplanned and disruptive outages in the same month [11]. Even national services such as the Swedish Tax Agency (Skatteverket) has on different occasions experienced outages and poor performance during the yearly income tax declarations [12, 13]. Besides revenue loss and damaged reputation, enterprises spend many work-hours diagnosing and restoring services[14].

Due to financial and operational implications of these incidents, performance diagnosis has thus become a major challenge in the day-to-day operation of cloud services and infrastructure. Operators must discover anomalous performance behaviour quickly, identify symptoms and root-causes, and deploy corrective strategies to remediate problems. The need to improve user experience, achieve stable performance, and reduce diagnosis time has given rise to many systems and methods. The contribution of this thesis is as follows. In Paper I we present an extensive survey of existing systems and methods. The aims are a) to classify existing methods based on their objectives and methodologies, and b) to assess research trends and open challenges. Paper III elaborates on a system for identifying capacity bottlenecks in a logical graph of datacenter components. It is presented as a use-case within Intel's Apex Lake framework. The Apex Lake is framework is being developed at the Cloud Service Labs, Intel Ireland and is introduced in Paper II. Finally, Paper IV proposes two adaptive mechanisms for detecting point anomalies in real-time performance behaviour of a web application in a cloud environment.

The remainder of this thesis introduction is organized as follows. Chapter 2 introduces the general concepts of cloud computing and challenges for achieving good performance. Chapter 3 discusses concepts in performance management and diagnosis in cloud datacenters. Chapter 4 summarizes the contributions of this thesis and discusses opportunities for future work. Papers produced in the course of this work are appended.

Chapter 2

Cloud Computing

Cloud computing is a paradigm shift in the way computation is delivered and consumed. It is based on the vision that information processing can be done more efficiently on large farms of computing and storage systems accessible via the Internet [15]. It provides as set of technologies that allow computational resources (e.g. compute and storage) to be provisioned *as a service* over the Internet on a pay-per-use and on-demand manner [16]. The growing adoption rate of the cloud is due to two appeal factors a) it offers seamless scalability and elasticity to users without the huge initial capital expenditure b) resources that can be metered so users are billed for only what they used [3]. These have led to a radical change in the way IT departments and application providers deploy and manage IT services [17].

2.1 Clouds Deployment Models

Cloud infrastructures can be classified based on ownership and how they are managed.

- *Private clouds* are infrastructures managed and utilized by individual organizations. Many organizations adopt private clouds to gain the benefits of cloud computing (such as flexibility and scalability) while having full control on the infrastructure and its security [15].
- *Public clouds* are virtualized infrastructures managed by an organization but leased on a pay-per-use basis to third-parties e.g. Google Cloud Platform, Amazon Web Services, Microsoft Azure, and RackSpace.
- *Community clouds* are shared by group of organizations or individuals with a common interest (e.g. mission, security, regulatory and compliance considerations). The North Carolina Education Cloud (NCEdCloud) is a community cloud [18].

- *Hybrid clouds* allow large enterprises who typically host their core IT infrastructure (compute and storage) on private or community clouds to team up with at least a public cloud for the purpose of extending the capacity of their private clouds arbitrarily by leasing public cloud resources to meet sudden surge in user demand through a mechanism known as *cloud bursting* [3].

- *Federated clouds* are an emerging type of clouds composed of only public clouds, only private clouds or both to provide seemingly infinite service computing utility [19, 20] to end-users. A typical instance of federated clouds is one established among multiple datacenters owned by a single cloud provider. Federation enables high compatibility and interoperability between disparate cloud services through open APIs that allows cloud users to deploy services based on offerings from different vendors or move data easily across platforms.

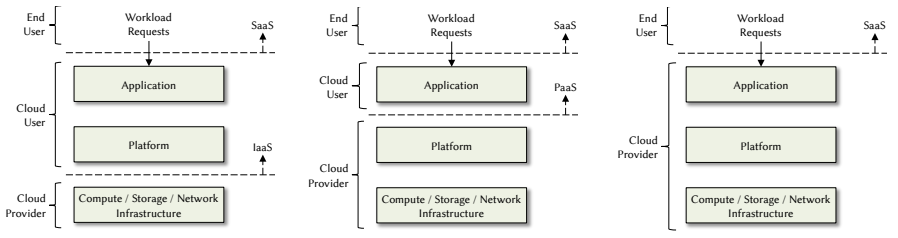
2.2 Cloud Delivery Models

Public clouds offer three major service delivery models based on the level of abstraction at which the service is offered [3, 21, 15].

- *Software as a Service* (SaaS)– delivery of finished applications, along with required software, operating system, network and hardware, to end-users and accessible through various client devices over the Internet. Though it is has minimal customization, the users generally have limited control on the application, platform and underlying infrastructure. SaaS applications include Customer Relationship Management (e.g. Salesforce, and ServiceNow), accounting/ERP systems (e.g. NetSuite) and Web 2.0 applications (e.g. LinkedIn, and WordPress).

- *Platform as a Service* (PaaS)—provides the capability for cloud users to build, deploy and manage applications using application development tools, APIs, operating systems, and hardware supported by the provider. The user only has control on the application, its architecture and hosting environment configuration but not on the underlying operating system, network, storage, or servers. Google App Engine, Windows Azure, and Amazon's Elastic Beanstalk are some of the most popular PaaS offerings today.

- *Infrastructure as a Service* (IaaS)—provisioning of fundamental computing resources such as servers, network, and storage; users are responsible for the installation of the operating system and can use application development tools of choice to develop arbitrary applications. Examples of IaaS providers are Amazon EC2, Google Cloud Platform, Microsoft Azure Infrastructure and Rackspace.



(a) IaaS (b) PaaS (c) SaaS

Figure 1: Roles of stakeholders under the three service delivery models [3].

2.3 Cloud Stakeholders, Roles and SLA

Stakeholders in cloud computing environments are generally categorized into one of three roles depending on the service delivery models [3]. According to Fig. 1, the responsibility of the *Cloud Provider* or Infrastructure Provider (IP) varies from provisioning only infrastructures in IaaS, provisioning both platforms and infrastructures in PaaS, to provisioning entire cloud stack (infrastructure, platforms, and application) in SaaS. A *Cloud User* or *Service Provider* (SP) is prominent under the IaaS and PaaS model. The service provider uses infrastructure and/or platform provided by the IP to host application services that are consumed by the End User. An End User generates application workloads that are processed by a cloud providers resources in the case of SaaS, or by a service provider in the case of PaaS and IaaS.

Interactions between the roles, especially, between the cloud and service providers may be governed by a formal contract document called the Service Level Agreement (SLA). An SLA describes expected quality of service (QoS) and legal guarantees [22]. A typical SLA contains, among others, important items such as the *service guarantees* and *service credits*. The service guarantees specifies functional metrics (e.g. availability, response time, and security) which a cloud provider strives to meet over a service guarantee time period. The service credit is an amount credited to the customer or applied towards a future payment when a cloud provider could not meet service guarantees [23]. For example, Amazon EC2, currently offer between 99% to 99.95% service availability guarantees with up to 10% of original service rate as service credits [24].

2.4 Cloud Datacenters

Cloud computing is powered by datacenters housing several servers, communications and storage systems co-located because of their common environmental, physical security, and maintenance requirements [8]. Consolidation is a method

used in cloud datacenters to ensure efficient utilization of these resources. By sharing server resources among multiple applications cloud providers prevent under-utilization and server sprawl in datacenters [10, 8].

Virtualization technology makes consolidation and sharing better by providing performance isolation among co-located applications. It is also cost-effective since consolidation reduces capital and operational expenses as well as energy consumption [25]. The two mainstream platforms for realizing virtualization today are *hypervisor-based* virtualization and *container-based* virtualization [26]. Hypervisor-based virtualization emulates a machine hardware and allows instances of the emulation (i.e., *virtual machines* (VMs)) to run on another physical machine managed by a specialized operating system (OS)–called the *hypervisor*. This approach is OS agnostic since the *guest* OS (i.e., OS installed on the VM) may differ from the *host* OS (i.e., OS running on the physical host of the VM). Xen [27], KVM [28], and Hyper-V [29] are examples of popular hypervisors [15, 30]. Rather than emulating an hardware platform, container-based virtualization provides virtualization at the OS-level in order to reduce performance and speed overhead of hypersor-based virtualization [31]. OS-level virtualization allows multiple isolated Linux environments (i.e., *containers*) to share the base kernel of the host OS [26, 32]. Docker [33, 34], LXC [35], and OpenVZ [36] are examples of container-based virtualization platforms.

Though containers are now enjoying growing adoption, VMs remain the common unit of cloud deployments due to the maturity of the technology. VMs are manageable software-defined units and facilitate the elasticity of the cloud. They can be easily moved (*migration*) from one server to another in order to balance the load across the datacenter or to consolidate workloads on fewer servers. They can also be replicated across servers (*horizontal scaling*) and their resource capacity (e.g. CPU cores) can be increased or decreased to address overload or underload situations (*vertical scaling*).

A major challenge in IaaS datacenters is resource allocation, which is concerned with how to optimally share or allocate computing, storage and network resources among a set of VMs in a manner that attempts to jointly meet service objectives of both the service providers and cloud provider [3, 37]. This has been identified as a difficult task [38, 39] due to the scale of datacenters, the plurality of applications and their complex architectures.

2.4.1 Software-defined Infrastructures

The growing adoption of cloud computing for running mission-critical and performance-sensitive applications such as analytics [40], machine-to-machine communications [41], mobile services [42] and the IoT [1] [43] is driving a higher demand for performance, agility, cost and energy efficiency. The need for increased automation and programmability of the infrastructure to meet these

[1] IoT is an acronym for Internet of Things, a term used to refer to the ever-increasing network of physical objects through the Internet, Radio Frequency IDentification (RFID) and other sensor network technologies.

requirements has led to recent trend towards Software-defined Infrastructures (SDI).

Many studies have applied *autonomic computing* [44] techniques in various aspects of managing existing clouds such as resource allocation and scheduling [45, 46, 47] as well as energy management [48, 49, 50] to facilitate automation. SDIs are envisioned to transform the compute, storage and network infrastructures to software-defined and dynamically programmable entities [51]. According to Kandiraju et al [52] an SDI is an infrastructure that is continuously transforming itself by appropriately exploiting heterogeneous capabilities, using insights gained from built-in deep monitoring, to continuously honor consumer SLAs amidst provider's constraints (e.g. cost and energy constraints). While only a few SDI products or platforms are currently available–e.g. VMware's EVO:RAIL [53] and Google's Andromeda [54]–many aspects of the SDI technology are still evolving. However, the core architectural components of SDIs are described below;

- *Software-defined Compute* (SDC): increases the programmability of existing virtualized compute resources (e.g. VMs, containers, virtual CPUs, memory) and dynamically leverages specialized processing units such as graphical processing units (GPUs), field-programmable gate arrays (FPGAs), and other accelerators [52]. SDC decouples provisioning of heterogeneous compute resources from the underlying hardware or operating system so that provisioning is based on specified or discovered workload requirements [55].

- *Software-defined Network* (SDN): separates control and management functions of the network infrastructures away from the hardware to the server for improved programmability, performance isolation, efficiency and security [51]. Through virtualization, SDN allows network resources to be transformed to virtual devices (e.g. switches, links and end-points) that connects different virtual compute and storage instances [52].

- *Software-defined Storage* (SDS): is responsible for managing huge volume of data by isolating the control and management functions from the data storage system and dynamically leverages specialized storage when available [51]. The advantage of such isolation is that it reduces management complexity and also the cost of infrastructure [56]. The basic unit of a SDS is the virtual storage—an abstraction of a combination of different storage capabilities (e.g. access bandwidth (IOPS), solid-state disk, RAID, block storages, distributed file systems) to guarantee SLAs [52].

The SDI components are managed by a SDI *controller* that provides the control intelligence required to meet workload requirements and SLA. The SDI controller uses the classical MAPE loop [57] to continuously keep track of the current state of SDI entities and acts on virtual and physical infrastructure to ensure SLA compliance [52].

Chapter 3

Cloud Performance Management

The performance of a system can be defined in terms of the system's ability to perform a given task correctly. If it does, then its performance is measured by the time taken to perform the task, the rate at which the task is performed, and the resources consumed while performing the task [58]. Hence, performance is composed of two major components [58, 59] namely;

- *Time behaviour*: The degree to which the response time, execution time and throughput rates of a system meet requirements when performing its functions.
- *Resource utilization*: The degree to which the amounts and types of resources used by a system meet requirements when performing its functions.

In general, the efficiency of a system can thus be expressed in terms of performance. That is its time behaviour relative to its resource utilization under stated conditions. The most commonly used performance metrics are [58];

- *Response time*–is time interval between the end of a request submission and the end of the corresponding response from the system for interactive users in a timeshared system (also known as *latency*). For batch jobs, it is the time between the submission of a batch job and the completion of its output (also called *turnaround time*). The latency of a web request may be expressed in milliseconds or even in minutes.
- *Throughput*–is the rate (requests per unit of time) at which application requests can be serviced by a system. For a batch job, throughput may be expressed in terms of jobs/tasks completed per unit time. For interactive applications, throughput can be described in terms of the number of requests or transactions completed over a given period of time.

- *Utilization*–is the fraction of time a resource is busy servicing requests. A simple model of CPU utilization is to compute the ratio of busy time to total elapsed time over a given period (expressed in percentage).

Performance is related to the majority of obstacles for adoption and growth of cloud computing such as availability, capacity bottlenecks, scalability, reliability and security [60]. Hence, it is important for user satisfaction and guaranteed service assurance [61, 8]. Cloud computing relies on effective performance management to meet service-level objectives (SLO) and SLA. Performance management in a cloud environment involves the monitoring and measurement of relevant performance metrics of hosted application services and infrastructure in order to assess their performance. It also encompasses the analysis and visualization of the data, as well as detection and diagnosis of performance problems.

Performance management can be viewed from two perspectives depending on the of role a cloud stakeholder [62]. Figure 2 presents these perspectives in the context of an IaaS cloud.

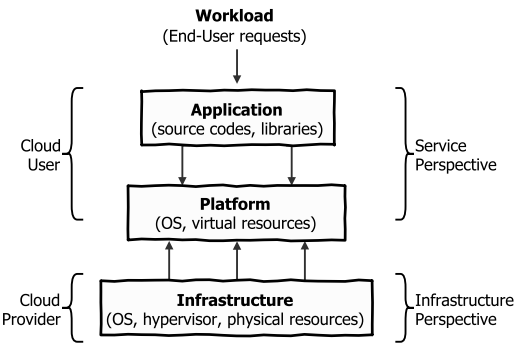


Figure 2: Performance perspectives in IaaS clouds.

The *service perspective* is a top-down approach to performance management and is in the purview of the service provider. It is concerned with performance and workload modeling and analysis for capacity planning (resource estimation), online resource control, anomaly detection, SLA monitoring, as well as end-user experience and behaviour learning. According to Gregg et al [62], the target metrics in this perspective includes requests rates–workload applied to the service, latency–responsiveness of the service, and completion–error rates. While the service provider generally has little to no knowledge about the external runtime environment, she has complete control on her service architecture, operating systems, source codes; and thus can perform drilled-down analysis.

The *infrastructure perspective* provides a bottom-up view of the entire infrastructure from physical infrastructure (compute, network, storage) through the virtual infrastructure (VMs, containers, etc) to the hosted services. Though the cloud provider has limited access to the behaviour and performance of hosted

applications and platforms, she is responsible for monitoring and analyzing the operational metrics of physical infrastructure components. The cloud provider must ensure that statistically multiplexing limited hardware resources among disparate applications black-boxes (VMs) do not affect the SLA and performance of those applications. Resource analysis is a major task in this perspective. It involves analysis of performance metrics of resources such as CPUs, memory, disks, network interfaces, buses and interconnects [62]. Results of the analysis are applicable in two associated activities a) performance diagnosis–identifying resource bottlenecks, faults or other root-causes to explain reported service-level anomalies or SLA violations, and b) capacity planning–this includes resource demand profiling, resource estimation, and resource scaling. Resource throughput, utilization, and saturation are commonly used metrics in this case.

The perspectives also affect how datacenter components are monitored. The service perspectives allows activities such as request/transaction tracing or tagging, source-code instrumentation, profiling of virtual resources by the service provider; while the cloud provider can only perform infrastructure monitoring such as physical resource profiling and kernel instrumentation in the host operating system. The monitoring aspect of performance management in cloud computing is treated extensively in [63, 64]. The focus of this thesis is on analyzing performance metrics data for diagnosing performance problems in cloud infrastructure.

3.1 Cloud Performance Diagnosis

The occurrence of performance issues is a norm, rather an exception in large scale systems [65]. Performance anomalies and unexpected variability are common even in large-scale infrastructures such as in Google clusters [5] and Amazon Web Services [4]. Apart from damaging end-user experience and service reputation, performances problems sometimes lead to unplanned outages. The far-reaching consequences of performance problems drives the need for techniques and systems to automatically detect problems and guide operators to potential root-causes and solutions.

In general, the goal of performance diagnosis in a datacenter is (a) to detect both expected and unexpected performance events, (b) analyze operational metrics to gather evidences for localizing root-cause(s) responsible for observed behaviour, and (c) to either recommend strategies that may resolve the problem or automatically actuate the remediation.

From the service performance perspective, this thesis deals only with post-deployment diagnosis (i.e. during operation). Pre-deployment diagnosis of performance anti-patterns when testing for software quality during development phase [66, 67, 68] is beyond the scope of this work.

3.1.1 Challenges

The following are cloud-specific factors that we consider as obstacles to effective performance diagnosis in datacenters.

1. *Scale*: A single datacenter may host up to tens of thousands servers. Multiple datacenters owned by a large vendor such as Microsoft or Google may contain up to one million servers in total from multiple datacenters [69]. Through consolidation, each server can host tens or even hundreds of VMs and applications.

2. *Complexity*: The topology of the cloud infrastructure is complex; they span single facility to geographically dispersed facilities. Cloud datacenters support a heterogeneous mix of application services ranging from latency-sensitive services (e.g. multi-tier web applications) to batch-oriented services (e.g. scientific simulations). Also, applications often have varying and sometimes conflicting performance objectives. Diagnosing problems in this environment is further aggravated by limited observability due to separation of control between cloud and service providers.

3. *Dynamism*: Cloud applications contain multiple inter-dependent components sharing equally inter-dependent physical resources. The ensuing complex dependency is bound to induce dynamic behaviours and cascading bottleneck or faults. While co-location of multiple applications on the same server improves datacenter resource and energy utilization, it has been reported to cause severe variability in performance of latency-sensitive services [5] due to lack of performance isolation. Due to dynamic resource management, the execution context of services is also constantly changing as load continuously flow in and out of the datacenter.

4. *Autonomics*: Datacenters have become highly autonomous since manual management is not practical from a performance perspective [70]. Such advancement require performance diagnosis systems that can learn and operate in on-line scenario. Offline analysis and detection, as well as ticket-based manual resolution may not fit this environment.

5. *Monitoring and the data deluge*: The performance of services and infrastructure is currently monitored using diverse tools [62]. The resulting data end up in varying format and semantics with noises that sometimes mask anomalies. Also, datacenters produce a staggering amount of streaming operational data [70] that can easily overwhelm operators and alert systems.

6. *Complex problem structure*: The notion of performance is generally subjective [62]. What constitute normal or anomalous behaviour varies from service to service and from one infrastructure provider to another. Also, the occurrence of anomalies is dynamic. Some anomalies are highly contextual because they occur under specific workload and usage situations.

There could be a wide variety of potential root-cause even for a minute drop in latency. Each problem has a characteristic behaviour visible only by certain performance metrics. Some problem symptoms are recurrent in nature; so eliminating a manifestation may not necessarily eliminate the actual root-cause nor solve the problem [68].

3.1.2 Categories of performance problems

Wert et al [68] proposed that performance problems can be categorized into three layers namely; *symptoms*–externally visible indicators of a performance problem, *manifestation*–internal performance indicators or evidences, and *root-causes*–physical factors whose removal thereof eliminates the manifestations and symptoms of a performance incident. We adopt their classification to categorize common performance problems in Table 3.1.

3.1.3 Activities in performance diagnosis

Performance diagnosis activities can be broadly divided into two subtasks:

- *Identification*: The goal of performance problem identification is two-fold. First task is to, in a timely manner, discover performance symptoms (e.g. degraded response time) and/or deviations from expected behaviour based on thresholds, baselines, models, or SLAs. This is known as Performance Anomaly Detection (PAD). SEAD [71], EbAT [72], PAD [73], and UBL [74] perform this task. Once a symptom is detected, next task is to figure out if it is similar to a previously seen problem so that a known root-cause and solution can be recommended. This speeds up the diagnosis process and avoids escalation [65]. If it is a new symptom, the system must find evidences or manifestations that partly explains the anomaly and help narrow search path for potential root-causes. Fingerprinting is a common technique for summarizing and persisting previously seen problems [65]. Example systems includes Spectroscope [75], Prepare [76], vPerfGuard [77], PerfCompass [78], and Orion [79].
- *Root-cause analysis*: This involves analysis of identified symptoms (anomalies) and manifestations to determine the potential root-causes and suggesting actions to correct them. Accurate localization of the problem is important to mitigate performance problems, restore the system to good state, and for future recall. Examples of such systems include X-ray [80], PerfScope [81], PeerWatch [82].

In addition to problem diagnosis, performance management also include the *remediation* of performance problems. This involves taking actions to restore the system to normal state. It relies on information and recommendations from the two phases of the diagnosis process. Systems such as CloudPD [83] and CPI2 [5] propose strategies for correcting performance problems due to capacity related causes.

Table 3.1: Classification of common systems performance problems.

Problem	**Category**	**Perspective**	**Instances**
Increasing response time	Symptoms	Service	Non-responsive application, Unexpected slow down in file transfer, Degraded video quality or resolution.
Throughput degradation under load	Symptoms	Service	Dropped requests or packets, Incomplete search results, Degraded video resolution.
Service unavailability	Symptoms	Service	Service hanging or crashing under heavy load, Unreachable service.
SLA violation	Symptoms	Service	Availability $< 99\%$, Latency $> 3ms$.
Sudden latency spikes	Symptoms	Service	Latency $>$ ($mean$ latency $+ 6sd$)
Resource under-utilization under load	Manifestation	Service/Infrastructure	Low IO utilization under heavy IO workload.
Resource over-utilization	Manifestation	Service/Infrastructure	High CPU utilization (near 100%)
High resource saturation	Manifestation	Service/Infrastructure	100% CPU utilization High average resource queue length, Excessive dropping of network packets.
Resource contention or interference	Manifestation	Service/Infrastructure	High CPU steal time, High cycle-per-instruction (CPI).
Resource dependency	Manifestation	Service/Infrastructure	High CPU waiting time due to IO
Network congestion	Root-cause	Service/Infrastructure	Packet loss, Queueing delays, Blocking of new connections.
Unexpected workload spikes	Root-cause	Service/Infrastructure	Sudden spike in update requests, Flash-crowd behaviour.
Periodic load behaviour	Root-cause	Service/Infrastructure	Expected workload fluctuation, Diurnal workload pattern.
Application-level issues	Root-cause	Service/Infrastructure	Component or dependency failure, Bugs in application codes, Exhausted thread pools, Failed application upgrade, Corrupted data error
Capacity bottlenecks	Root-cause	Service/Infrastructure	Resource hogging and contention, Memory leaks, Unanticipated interference.
Platform & OS issues	Root-cause	Service/Infrastructure	Transient events e.g. Memory hardware errors, JVM garbage collector issues.
Machine-level faults	Root-cause	Infrastructure	Faulty server components, DRAM soft-errors or disk errors, OS kernels bugs, abnormal reboots, Unplanned power outage or overheating.
Routine system maintenance	Root-cause	Service/Infrastructure	Routine backups, Forklift replacement, OS upgrade.
Operator errors	Root-cause	Service/Infrastructure	Misconfiguration error, Inadvertent database locks, Erroneous overwrite or data deletion, Wrong price setting.
Security violations	Root-cause	Service/Infrastructure	Denial of service attacks, Worms and virus infection, or Theft.

Chapter 4

Summary of Contributions

This thesis contributes to the understanding of performance problem diagnosis in systems especially in cloud environments and also propose new techniques to improve the process. Firstly, we carried out an extensive survey of the research area, covering the classification and review of different approaches. We found that application-level performance diagnosis still rely on simplistic threshold-based techniques. Considering the impact of performance problems on SLA/SLO compliance, it is surprising why this is the case. Hence, we propose techniques for automatically detecting anomalies (symptoms) in real-time service performance behaviour in order to improve end-user experience as well as prevent SLA violations or service outages. Furthermore, we devised a knowledge-driven technique for identifying performance problem manifestations in datacenter resources within the context of Apex Lake[1]. The approach is based on combining certain performance metrics with the logical dependencies across resources. The significance of the work in this thesis is to drive performance diagnosis to be more automated improve the quality of service, user experience as well as the speed of diagnosis. The thesis contributions also include scientific publications and software prototypes.

Research Methodology. The methodology used in this thesis is experimental. We begin by definition of problems, literature review, design and implementation of algorithms, design of experiments and analysis of results. There are generally two evaluation approaches in this area. The first is to evaluate proposed algorithms using public datasets. The other is through active perturbation [84] whereby algorithms are evaluated empirically on a real testbed with realistic load and fault injections to bring it close to the real environment being emulated. Though the former suffers from the lack of public application/systems performance data for broad use-cases, it is better suited for evaluating algorithms on independent data especially when the data is labeled. On the other hand, it is challenging to recreate desired behaviour in the later.

[1]Apex Lake is a framework developed at Intel for enhancing smarter orchestration in clouds especially in SDIs.

The later approach is taken to evaluate techniques proposed in Paper III and IV. We collect data from actual applications running in a virtualized testbed with emulated capacity bottlenecks via active perturbation.

4.1 Paper I

The ubiquity of performance problems has led to many interesting diagnosis techniques. However, there exist no classification of the objectives and characteristics of proposed approaches. In order to assess progress, identify trends and open challenges we conducted a survey of over forty research articles spanning over ten years (2002–2014) covering systems from a wide range of application domains (e.g. dedicated environments, distributed and grid systems, as well as multi-tier web services).

The survey, published by the ACM Computing Survey, gives a background of aspects of the problem, and categorizes existing solutions using multiple factors such as goals, domain of application, and methodology. Research trend shows that research volume and complexity of detection methodologies seem to follow the increasing complexity of systems. The task of detecting performance anomalies (i.e. symptoms of problems) dominates research contributions in this area; accounting for 53% of reviewed works. It is followed by problem identification (identifying bottlenecks or root-causes to explain anomalies) which accounts for 29% of reviewed papers. Hybrid systems combining the two tasks account for 18% of reviewed papers. Existing techniques exploit many strategies such as static thresholding, fine-grain workload models, fingerprints, flow paths or dependencies, expert knowledge (e.g. rules). Though statistical models and tests are popularly used, availability of cheap computing power has led to increasing adoption of machine learning techniques. While end-to-end, autonomous, and distributed diagnosis remain open challenges, the lack of open-source performance data and problem taxonomy hinder the pace of performance diagnosis research.

4.2 Paper II & III

Landscape colouring was introduced as a use-case within Intel's `Apex Lake` framework introduced in Paper II. While Apex Lake is being developed by Cloud Services Lab (CSL), Intel Labs Europe, Ireland, since 2014, Umeå University worked together with CSL to collaboratively develop and write Paper II as well as proposed the concept of landscape colouring as a use case for Apex Lake. Paper III refines and extends the original idea of landscape colouring as well as demonstrates it with preliminary experiments.

This work is motivated by the concerns of low datacenter utilization [8] and the need for tighter orchestration [51] in modern datacenters, such as software-defined environments, where the logical topology and composition of the datacenter is dynamic. We propose landscape colouring, a technique that

combines dynamic topology information with operational metrics of individual datacenter components (applications, servers, VMs, etc.) in order to reason about anomalies (or manifestations) such as resource over-utilization and saturation. We employed components of Apex Lake as follows. The landscaper is used to automatically and continuously organize the datacenter into a three-layer graph structure of service, virtual and physical entities. `Cimmaron` monitors operational metric, such as node (utilization) and the amount of tasks on its queue (saturation), that are used to characterize node behaviour into discrete operational states using a fuzzy inference system. Nodes are flagged anomalous if they persist in a specified set of undesirable states. When many nodes are flagged simultaneously, we rank anomalous nodes according to how important they are in the graph to enhance the resolution process. The rank score is a function of the influence of a node's neighbours, the layer it belongs, and the number of anomalous nodes depending on it.

We performed two experiments to demonstrate our approach in a virtualized environment with emulated CPU and IO saturation bottlenecks. Results show that the technique is able to identify and rank nodes with anomalies (e.g CPU contentions due to inter-node dependency). We plan to perform more experiments with multiple applications on a larger testbed (where the topology is truly dynamic) to investigate the scalability of technique and to correlate service-level symptoms with capacity bottlenecks in the infrastructure. We will also be exploring how the technique can be used to trigger corrective actions automatically or to drive other datacenter optimizations within the Apex Lake framework.

4.3 Paper IV

Application performance degradation (e.g. high latency) is common on the Internet today; and have been reported to be mostly due to capacity-related issues rather than application bugs or machine-level faults [65, 8]. However, it has received less research attention and existing studies are largely based on simplistic threshold-based techniques.

Due to the correlation between application performance and end-user satisfaction, we proposed and evaluated two schemes for detecting point anomalies in real-time service latency measurements. The first exploited the fact that applications sometimes have known baselines. We devised a behaviour-based anomaly detection technique using adaptive Kernel Density Estimation (KDE) and its KD-tree variants to detect deviations from given baseline. The second scheme is for the case where a known baseline behaviour does not exist, we proposed a prediction-based method to discover point anomalies. Significant deviations in prevailing service behaviour are detected by predicting service latency using Holt-Winter's forecasting and applying adaptive EWMA control chart limits on the prediction errors.

We evaluated the schemes using a web application benchmark on a virtualized

testbed with realistic load and bottleneck injections. We observed that the tree variant of the adaptive KDE algorithm offers lower error rates than the standard KDE. Under appropriate parameter settings, the prediction-based scheme is able to follow the trends in service behaviour and detect out-of-control anomalies. Results suggest that many service performance anomalies could be explained by system-level capacity bottlenecks but not all low-level bottlenecks lead to service-level anomalies. In the future, we plan to further evaluate accuracy of the techniques using labeled datasets from public system traces.

4.4 Future Work

This work can be extended on multiple fronts. In general, we aim to improve the evaluation of the proposed techniques. Though it is hard to perform substantial statistical evaluations of the anomaly detection using an experimental approach, we hope to accomplish this when we address root-cause analysis and problem remediation. We plan to evaluate techniques introduced in Paper IV using labeled datasets from real systems logs so that we can quantify their accuracy and false alarm rates. The experiments in Paper III are preliminary, we plan to assess the scalability of proposed method on a larger testbed with multiple applications; and to quantify accuracy with suitable statistical analysis.

Furthermore, the prediction-based anomaly detection technique will be extended with low-level resource metrics. By modeling relationships between application-level metrics and resource metrics, we may be able to pinpoint bottleneck resources to explain an observed anomaly. The static thresholds of the fuzzy membership functions in Paper III could be learned on-line to reflect prevailing workloads in the system. Also, in addition to ranking anomalous node, it could adapt the method to differentiate victim nodes from antagonist nodes when there is a resource contention.

Most performance problems are recurrent in nature [68] with well known causes. We shall attempt to structure domain knowledge about capacity-related anomalies and bottleneck in cloud infrastructure. By linking symptoms to manifestations and to root-causes, we posit this will contribute towards the realization of automated problem remediation since it will be easy to associate root-causes with potential remediation strategies. For example, upon detecting a latency spike (symptom), the fuzzy inference system may find CPU utilization to be consistently high (manifestation). Further analysis may expose unexpected workload spike (root-cause) as the actual culprit. A possible solution would be to scale CPU resource to meet the workload demand.

The overall goal of our research is to close the performance problem management loop; from problem identification through root-cause analysis to problem remediation. Specifically, we plan to investigate existing strategies and develop techniques for automatically rectifying observed problems. Such a technique could, for instance, adjust number of replicas or CPU cores of a VM to correct sustained latency growth caused by a workload spike or CPU bottleneck. On the

other hand, it could migrate a noisy VM to an idle host or enforce policy-based resource throttling, to correct throughput degradation in a batch job.

Bibliography

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree Compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, pp. 1–10, IEEE, 2008.

[2] Q. Guan, C.-C. Chiu, and S. Fu, "CDA: A Cloud Dependability Analysis Framework for Characterizing System Dependability in Cloud Computing Infrastructures," in *18th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 11–20, IEEE, 2012.

[3] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[4] A. Iosup, N. Yigitbasi, and D. Epema, "On the Performance Variability of Production Cloud Services," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 104–113, IEEE, 2011.

[5] X. Zhang, E. Tune, R. Hagmann, R. Jnagal, V. Gokhale, and J. Wilkes, "CPI 2: CPU Performance Isolation for Shared Compute Clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 379–391, ACM, 2013.

[6] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: Building more Robust Cloud Applications," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 700–711, ACM, 2014.

[7] C. Klein, A. V. Papadopoulos, M. Dellkrantz, J. Durango, M. Maggio, K.-E. Arzen, F. Hernández-Rodriguez, and E. Elmroth, "Improving Cloud Service Resilience using Brownout-aware Load-balancing," in *33rd International Symposium on Reliable Distributed Systems (SRDS)*, pp. 31–40, IEEE, 2014.

[8] L. A. Barroso, J. Clidaras, and U. Hölzle, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[9] Aberdeen Group, "The Performance of Web Applications: Customers are Won or Lost in One Second (Research Report)." http://www.aberdeen.com/research/5136/ra-performance-web-application/content.aspx, May, 2015. Accessed: 2016-03-31.

[10] C. Wang, S. P. Kavulya, J. Tan, L. Hu, M. Kutare, M. Kasick, K. Schwan, P. Narasimhan, and R. Gandhi, "Performance Troubleshooting in Data centers: An Annotated Bibliography?," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 3, pp. 50–62, 2013.

[11] CloudEndure, "The Top 9 Outages that made Headlines in Q4 2015 : CloudEndure." https://www.cloudendure.com/blog/top-9-outages-made-headlines-q4-2015/, 2015. Accessed: 2016-03-31.

[12] Göteborgs-Posten, "E-deklarationer överbelastade Skatteverket - Sverige - Göteborgs-Posten." http://www.gp.se/nyheter/sverige/1.2703888-e-deklarationer-overbelastade-skatteverket, 2015. Accessed: 2016-04-03.

[13] Aftonbladet, "Nyfikna skattebetalare sänkte sajten." http://www.aftonbladet.se/nyheter/article22488308.ab, 2016. Accessed: 2016-04-03.

[14] Evolven Software, "Downtime, Outages and Failures–Understanding Their True Costs." http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html, September, 2011. Accessed: 2016-04-18.

[15] D. C. Marinescu, "Cloud Computing: Theory and Practice." http://www.scopus.com/inward/record.url?eid=2-s2.0-84902054996&partnerID=tZOtx3y1, 2013.

[16] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of-the-art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[17] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[18] EDUCAUSE, "Spotlight on Cloud Computing: Community Clouds." https://net.educause.edu/ir/library/pdf/LIVE1017b.pdf, 2010. Accessed: 2016-04-01.

[19] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, *et al.*, "The Reservoir Model and Architecture for Open Federated Cloud Computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[20] B. Rochwerger, A. Galis, E. Levy, J. A. Caceres, D. Breitgand, Y. Wolfsthal, I. M. Llorente, M. Wusthoff, R. S. Montero, and E. Elmroth, "Reservoir: Management Technologies and Requirements for Next Generation Service Oriented Infrastructures," in *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, pp. 307–310, IEEE Press, 2009.

[21] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[22] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA Framework for Cloud Computing," in *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, pp. 606–610, IEEE, 2010.

[23] S. A. Baset, "Cloud SLAs: Present and Future," *ACM SIGOPS Operating Systems Review*, vol. 46, no. 2, pp. 57–66, 2012.

[24] Amazon, Inc., "Amazon EC2 Service Level Agreement." http://aws.amazon.com/ec2/sla/. Accessed: 2016-04-04.

[25] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," in *30th International Conference on Distributed Computing Systems (ICDCS)*, pp. 62–73, IEEE, 2010.

[26] W. Li and A. Kanso, "Comparing Containers versus Virtual Machines for Achieving High Availability," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pp. 353–358, IEEE, 2015.

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and The Art of Virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[28] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: The Linux Virtual Machine Monitor," in *Proceedings of the Linux symposium*, vol. 1, pp. 225–230, 2007.

[29] A. Velte and T. Velte, *Microsoft Virtualization with Hyper-V*. McGraw-Hill, Inc., 2009.

[30] N. Antonopoulos and L. Gillam, *Cloud computing: Principles, Systems and Applications*. Springer Science & Business Media, 2010.

[31] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors," in *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 275–287, ACM, 2007.

[32] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," in *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*, pp. 171–172, IEEE, 2015.

[33] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux J.*, vol. 2014, Mar. 2014.

[34] Docker, Inc., "Docker: The Linux Container Engine." https://www.docker.com/. Accessed: 2016-04-18.

[35] "LXC: LinuX Container." https://linuxcontainers.org/. Accessed: 2016-04-18.

[36] "OpenVZ Linux Containers." https://openvz.org/. Accessed: 2016-04-18.

[37] J. G. et al., "Survey on Cloud Computing Resource Allocation Models and Methods," *International Journal of Computer Science and Management Research*, vol. 1, December 2012.

[38] A. Gulati, G. Shanmuganathan, A. Holler, and I. Ahmad, "Cloud-scale Resource Management: Challenges and Techniques," in *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, pp. 3–3, USENIX Association, 2011.

[39] P. Xiong, "Dynamic Management of Resources and Workloads for RDBMS in Cloud: A Control-theoretic Approach," in *Proceedings of the on SIGMOD/PODS 2012 PhD Symposium*, pp. 63–68, ACM, 2012.

[40] D. Talia, "Toward cloud-based big-data analytics," *IEEE Computer Science*, pp. 98–101, 2013.

[41] I. Stojmenovic, "Fog computing: a cloud to the ground support for smart things and machine-to-machine networks," in *Telecommunication Networks and Applications Conference (ATNAC), 2014 Australasian*, pp. 117–122, IEEE, 2014.

[42] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[43] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[44] M. C. Huebscher and J. A. McCann, "A Survey of Autonomic Computing—degrees, Models, and Applications," *ACM Computing Surveys (CSUR)*, vol. 40, no. 3, p. 7, 2008.

[45] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service," in *Proceeding of the USENIX International Conference on Automated Computing (ICAC'13). San Jose, CA*, 2013.

[46] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms," in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, (Washington, DC, USA), pp. 1–8, IEEE Computer Society, 2009.

[47] A. Roytman, A. Kansal, S. Govindan, J. Liu, and S. Nath, "PACMan: Performance Aware Virtual Machine Consolidation," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, (San Jose, CA), pp. 83–94, USENIX, 2013.

[48] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proceedings of the 9th international conference on Autonomic computing*, pp. 145–154, ACM, 2012.

[49] M. Guazzone, C. Anglano, and M. Canonico, "Energy-efficient resource management for cloud computing infrastructures," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 424–431, IEEE, 2011.

[50] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 13–23, ACM, 2007.

[51] C. Li, B. Brech, S. Crowder, D. M. Dias, H. Franke, M. Hogstrom, D. Lindquist, G. Pacifici, S. Pappe, B. Rajaraman, *et al.*, "Software defined Environments: An Introduction," *IBM Journal of Research and Development*, vol. 58, no. 2/3, pp. 1–1, 2014.

[52] G. Kandiraju, H. Franke, M. Williams, M. Steinder, and S. Black, "Software defined Infrastructures," *IBM Journal of Research and Development*, vol. 58, no. 2/3, pp. 2–1, 2014.

[53] VMware, Inc., "EVO:RAIL Hyper-Converged Infrastructure Appliance." http://www.vmware.com/products/evorail/. Accessed: 2016-04-19.

[54] Google, Inc., "Enter the Andromeda zone - Google Cloud Platform's Latest Networking Stack." https://cloudplatform.googleblog.com/2014/04/enter-andromeda-zone-google-cloud-platforms-latest-networking-stack.html. Accessed: 2016-04-19.

[55] Guru Rao, "Software Defined Compute Provides Workload-aware Infrastructure and Optimization through Automation and Open Technologies." https://www.ibm.com/developerworks/community/blogs/ibmsyssw/entry/software_defined_compute_provides_workload_aware_infrastructure_and_optimization_through_automation_and_open_technologies?lang=en, January, 2014. Accessed: 2016-04-05.

[56] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos, *et al.*, "Software defined Cloud: Survey, System and Evaluation," *Future Generation Computer Systems*, 2015.

[57] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[58] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, 1991.

[59] L. Bautista, A. Abran, and A. April, "Design of a Performance Measurement Framework for Cloud Computing," 2012.

[60] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the Clouds: A Berkeley View of Cloud Computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[61] Shailesh Paliwal, "Performance Challenges in Cloud Computing." https://www.cmg.org/wp-content/uploads/2014/03/1-Paliwal-Performance-Challenges-in-Cloud-Computing.pdf, 2014. Accessed: 2016-04-05.

[62] B. Gregg, *Systems Performance: Enterprise and the Cloud*. Pearson Education, 2014.

[63] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A Survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[64] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of Cloud Monitoring Tools: Taxonomy, Capabilities and Objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, 2014.

[65] H. Malik and E. M. Shakshuki, "Classification of Post-deployment Performance Diagnostic Techniques for Large-scale Software Systems," *Procedia Computer Science*, vol. 37, pp. 244–251, 2014.

[66] C. Trubiani and A. Koziolek, "Detection and Solution of Software Performance Antipatterns in Palladio Architectural Models," in *ACM SIGSOFT Software Engineering Notes*, vol. 36, pp. 19–30, ACM, 2011.

[67] K. C. Foo, Z. M. Jiang, B. Adams, A. E. Hassan, Y. Zou, and P. Flora, "Mining Performance Regression Testing Repositories for Automated Performance Analysis," in *10th International Conference on Quality Software (QSIC)*, pp. 32–41, IEEE, 2010.

[68] A. Wert, "Performance Problem Diagnostics by Systematic Experimentation," in *Proceedings of the 18th international doctoral symposium on Components and architecture*, pp. 1–6, ACM, 2013.

[69] StorageServers, "Facts and Stats of Worlds Largest Data centers." https://storageservers.wordpress.com/2013/07/17/facts-and-stats-of-worlds-largest-data-centers/, July 2013. Accessed: 2016-04-07.

[70] J. Murphy, "Performance Engineering for Cloud Computing," in *Computer Performance Engineering*, pp. 1–9, Springer, 2011.

[71] H. S. Pannu, J. Liu, and S. Fu, "A Self-evolving Anomaly Detection Framework for Developing Highly Dependable Utility Clouds," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 1605–1610, IEEE, 2012.

[72] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, "Online Detection of Utility Cloud Anomalies using Metric Distributions," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pp. 96–103, IEEE, 2010.

[73] M. Peiris, J. H. Hill, J. Thelin, S. Bykov, G. Kliot, and C. Konig, "PAD: Performance Anomaly Detection in Multi-server Distributed Systems," in *7th International Conference on Cloud Computing (CLOUD)*, pp. 769–776, IEEE, 2014.

[74] D. J. Dean, H. Nguyen, and X. Gu, "UBL: Unsupervised Behavior Learning for Predicting Performance Anomalies in Virtualized Cloud Systems," in *Proceedings of the 9th international conference on Autonomic computing*, pp. 191–200, ACM, 2012.

[75] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger, "Diagnosing Performance Changes by Comparing Request Flows," in *NSDI*, 2011.

[76] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pp. 285–294, IEEE, 2012.

[77] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vPerfGuard: An Automated Model-driven Framework for Application Performance Diagnosis in Consolidated Cloud Environments," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pp. 271–282, ACM, 2013.

[78] D. J. Dean, H. Nguyen, P. Wang, and X. Gu, "PerfCompass: Toward Runtime Performance Anomaly Fault Localization for Infrastructure-as-a-service Clouds," in *Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing*, pp. 16–16, USENIX Association, 2014.

[79] I. Laguna, S. Mitra, F. A. Arshad, N. Theera-Ampornpunt, Z. Zhu, S. Bagchi, S. P. Midkiff, M. Kistler, and A. Gheith, "Automatic Problem Localization via Multi-dimensional Metric Profiling," in *32nd International Symposium on Reliable Distributed Systems (SRDS)*, pp. 121–132, IEEE, 2013.

[80] M. Attariyan, M. Chow, and J. Flinn, "X-ray: Automating Root-cause Diagnosis of Performance Anomalies in Production Software," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pp. 307–320, 2012.

[81] D. J. Dean, H. Nguyen, X. Gu, H. Zhang, J. Rhee, N. Arora, and G. Jiang, "Perfscope: Practical Online Server Performance Bug Inference in Production Cloud Computing Infrastructures," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 1–13, ACM, 2014.

[82] H. Kang, H. Chen, and G. Jiang, "PeerWatch: A Fault Detection and Diagnosis Tool for Virtualized Consolidation Systems," in *Proceedings of the 7th international conference on Autonomic computing*, pp. 119–128, ACM, 2010.

[83] B. Sharma, P. Jayachandran, A. Verma, and C. R. Das, "CloudPD: Problem Determination and Diagnosis in Shared Dynamic Clouds," in *43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–12, IEEE, 2013.

[84] R. Apte, L. Hu, K. Schwan, and A. Ghosh, "Look Who's Talking: Discovering Dependencies between Virtual Machines Using CPU Utilization," in *HotCloud*, 2010.