Chapter 1

Introduction

The rapid development of Internet Technologies is opening up new pathways for transforming the way societies and industries work and communicate. Among other things, it has enabled various industries and enterprises to start offloading their applications (henceforth referred to as services) onto third-party infrastructures to achieve cost savings and scalability, get better performance, and deliver on-demand resource provisioning. The introduction of ubiquitous portable computing devices such as laptops, tablets, and smart phones has enabled people to interact with each other using these services at any time and from any location. To satisfy society's wide-ranging demands, services are becoming increasingly complex software systems consisting of many individual applications that are integrated to provide complex end-user functionalities such as search, social networks, news, and e-commerce. Furthermore, to provide a good quality of service, these integrated applications may require widely varying combinations of resources during their life-span. A key challenge that has emerged as a result is to identify effective means of runtime monitoring and controlling the resource usage of such services, and provisioning them with sufficient computational resources to ensure the desired performance [18, 74, 53, 123, 9, 90].

Computing technologies such as distributed systems, parallel computing, utility computing, grid computing, and virtualization have evolved through a number of phases over time as people have attempted to solve the problem of large-scale shared resource provisioning for services [60, 23, 111, 43, 138, 141, 106, 117, 105, 108, 63]. The emergence of large-scale Internet services coupled with the evolution of these technologies has fueled a trend toward a new resource provisioning paradigm called cloud computing.

The emergence of cloud computing in recent years has been a consequence of the progressive enhancement of these technologies with new features that enable resource provisioning in a flexible, cost effective, and scalable fashion while offering good ease of use. Multiple definitions and concepts of cloud computing have been proposed[12, 95, 142, 156, 54, 11, 102]. According to the NIST [95], cloud computing can be defined as a model that provides large-scale computing

farms based on the utility computing service model [1] to enable convenient on-demand network access to a shared pool of configurable computing resources such as CPU cores, networks, storage, and memory, which can be quickly provisioned and released with minimal management effort. The resources are usually geographically dispersed around the world in datacenters consisting of tens of thousands of commodity machines [24, 70, 100]. The number of machines available in this way will probably continue to grow for the foreseeable future because the demand for hosted services and remote resources is increasing steadily [156, 59, 67].

The challenge and complexity of provisioning resources for services on demand are increasing rapidly due to the growing interest in cloud computing among both the general public and industrial actors, and the rapid developments in the size and diversity of cloud computing resources and the services hosted on them [24, 70, 100, 156, 109, 77, 148]. At the large scales that are becoming increasingly common, it is necessary to account for the fact that there will always be a significant number of servers and network components that have failed at any given time. Further complexity is introduced by the heterogeneity of hosted services and the unpredictability of their workloads. Due to this heterogeneity and the sheer scale of cloud computing systems, there is an increasingly strong expectation that services should be self-configuring and self-managing. However, as the range of permissible configurations grows it is becoming increasingly difficult to achieve this goal.

The cloud infrastructure as a whole should not be simply regarded as heterogeneous collections of co-located commodity hardware that have been wired up together. Rather, it should be seen as a set of commodity machines that have been aggregated seamlessly and transparently into a single computing unit [24, 72, 16, 71, 144, 122, 130, 35]. Such an aggregation of commodity-class machines into a single computing unit provides an illusion of an operating system running on top of an infrastructure of tens of thousands of servers. This vision for large-scale systems can only be realized through advancements in the software management layer that abstracts the infrastructure [146, 72, 69, 121, 7, 34].

The management layer is responsible for many tasks such as the deployment of services to hardware resources, resource provisioning, scheduling, enforcement of quotas, monitoring, load balancing, resource usage collection, and handling component failures. These tasks must all be managed in a way that imposes a minimal overhead on the underlying system. The management layer is also responsible for ensuring that the system satisfies all of the hosted services' non-functional requirements relating to performance, reliability, scalability, availability, latency, and fault tolerance.

The task of the management layer for a large-scale distributed system has become more complex than the elements being managed. Consequently, as

[1] The utility computing service is a service provisioning model in which a service provider makes computing resources (e.g. processing power, storage, and high level services) available to the customer as needed, and charges the customer on a metered basis as is commonly done for water and electricity.

the demand for intricate distributed services has grown, the development of management tools for such systems has become a complicated problem in its own right [112, 116]. Due to the sheer complexity of the managed elements, it is imperative for the management system to be autonomous. That is to say, the management system should continually adapt the whole system to changes in environmental variables such as the workload, hardware availability, and software failure by automatically adjusting the decision variables [81, 33, 13] without any human intervention. Specifically, such solutions should have capabilities such as self-configuration, self-optimization, self-adaptation and self-healing that allow the system to control its managed elements and facilitate continuous functioning in the face of unpredictable changes in the infrastructure and hosted services.

This thesis focuses on the design and implementation of efficient algorithms, models and techniques for the autonomous monitoring, control, and provisioning of the diverse resources required to meet the demands of services and account for their resource usage. The aim is to develop effective tools for (i) enforcing quotas, (ii) provisioning optimal amounts of physical resources to meet service performance requirements, and (iii) collecting and aggregating service resource usage data for accounting and billing purposes in a timely fashion at runtime without any service interruption.

Quota management mechanisms are essential for controlling distributed shared resources to ensure that services do not exceed their allocated or paid-for budgets of credit, CPU hours, CPUs, RAM, storage capacity, and so on. Quota management tools for services must have mechanisms that both regulate resource usage and achieve an efficient and fair distribution of quotas among services. Cloud-wide monitoring and control of quotas is needed to achieve these aims and avoid the over- or under-provisioning of resources. Papers I[2] and II focus on managing quotas for services running across distributed nodes.

Quotas that apply over multiple nodes must be mapped onto physical resources on the basis of services' performance requirements. It is important to be able to continuously adjust the physical resources allocated to services at runtime in order to achieve guaranteed performance levels at all times in an environment where frequent and unpredictable changes are the norm rather than the exception. However, this is very challenging because of multi-faceted issues such as the dynamic nature of cloud environments; the need for supporting heterogeneous services with different performance requirements; the unpredictable nature of services' workloads; the complexity and differences among Key Performance Indicators (KPIs) used by each service, and the non-triviality of mapping performance measurements into resources. Papers IV–VIII present models and techniques for addressing these issues and managing temporary resource shortages due to e.g. flash crowds or hardware failure.

Furthermore, run time monitoring data should be collected for each service

[2]The term *quota* as used in this thesis has the same meaning as the phrase *resource allocation* as used in paper I; *quota* is used throughout this thesis to avoid the risk of confusion with *resource provisioning*.

and aggregated in order to maintain a single global state of the system that facilitates management activities such as accounting and billing, scheduling, load balancing, and network analysis. Such aggregation becomes extremely complex when dealing with systems spread over geographically dispersed locations, which means that the management task itself may consume significant computing and network resources unless done with care. Paper III focuses on collecting data across multiple nodes and establishing mechanisms to ensure consistency and synchronization in order to facilitate accounting and billing operations.

The thesis is organized as follows. Chapter 2 introduces the problems that the thesis aims to address. Chapter 3 reviews established methods for solving the quota and usage management problems. Chapter 4 briefly presents state-of-the-art techniques for resource provisioning. Chapter 5 summarizes the new contributions presented in this thesis. Finally, 8 papers produced in the course of the thesis work are appended.

Chapter 2

Runtime Resource Management and Control in Cloud Computing

Resource management and control is perhaps the most important task that must be performed to deliver a reliable and resource-efficient cloud computing system [52, 72, 150]. Each service's resource consumption must be continuously tracked at runtime to determine whether services are meeting their target Service Level Objectives (SLOs). Cloud infrastructures consist of many tens of thousands of units of commodity hardware with thousands of services deployed, all of which have different requirements and demands. Effectively running such an infrastructure is extremely complicated and challenging because there are so many variables that must be considered such as unpredictable service demand changes, hardware failures, and interference between collocated services. Logically, different approaches and methods that can reflect the different variables must be employed in order to allocate and control the available resources in the best possible way.

For the purpose of our discussion we assume that these resources can be divided into *virtual* and *physical* resources, a division that can be likened to the distinction between intangible and tangible assets in business and economics [26, 10]. *Virtual* resources are abstract resources such as prepaid credits, quotas (for CPU, storage, etc.), and accounting records while *physical* resources are resources that have actual physical existence such as CPU cores, RAM, and Storage. Accordingly, the control tasks are divided into *quota and resource usage control* and *resource provisioning*. *Quota and resource usage control* entails runtime tracking and controlling, collecting and aggregating virtual resources (i.e. quotas and resource usage) for each service, while the *resource provisioning* task is responsible for autonomic runtime prediction and allotment of optimal physical resources to services to meet their requirements.

Note that from this point onwards, when we talk about *resources* in the context of quotas and resource usage, we are referring to virtual resources. Conversely, in discussions about resource provisioning, *resources* means physical resources.

The following sections discuss the issues associated with controlling these two resource types in detail.

2.1 Quota and Resource Usage Control

Large-scale services often need sophisticated and scalable quota management and resource usage monitoring support. It is important to collect resource consumption statistics across nodes, clusters, or datacenters to facilitate network analysis and to support management decisions relating to accounting and billing, scheduling, and load balancing among other things. For example, resource usage data may need to be collected from several locations in order to generate a single bill for a customer's service consumption across the entire infrastructure.

Figure 1 presents a model of a cloud computing environment. We will use this

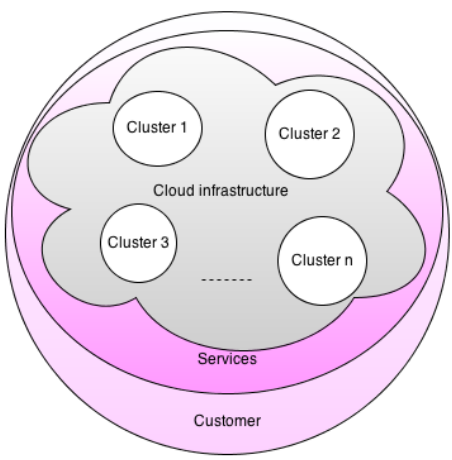

Figure 1: Elements in cloud environments.

model throughout this section when we talk about *quota and resource usage* control. The *cloud customer* deploys and runs services on a cloud infrastructure. A cloud customer may be a service provider, who leases resources offered by an infrastructure provider to host services that will be used by end-users or other service providers. The *services* are customers' applications, which are deployed across multiple clusters where they consume resources and impose resource demands on the cloud infrastructure. The *cloud infrastructure* consists of geographically distributed *clusters*, each of which is composed of thousands of commodity servers connected to a low-latency Local-Area Network (LAN) [84].

The infrastructure provider is responsible for managing the clusters' physical and virtual resources and controlling the resource usage of each individual service running across multiple clusters. The focus here is on how to manage quota and resource usage for services running across clusters.

Quotas and *resource usage* control mechanisms for services running across clusters profoundly influence the performance of the entire system, the provider's profitability, and the level of customer satisfaction. For example, to maximize customer satisfaction, the service provider should make sure to avoid over-committing the infrastructure's resources [112, 150]. Moreover, resources consumed by different services should be monitored and this monitoring data should be collected efficiently in a way that interferes minimally with the services' performance [48, 68, 61, 46, 118, 47]. Ideally, resource usage collection and quota management mechanisms should not impose significant constraints on either the customer or the cloud provider.

Quota control or enforcement systems are tools for monitoring, redistributing, and assigning shared resources (such as prepaid credits and quotas) to individual instances of a service running across multiple nodes based on each instance's demand, subject to the constraint that the sum of shares across instances must not exceed the aggregated global availability of these resources. Quota control mechanisms should guarantee that each individual instance's demands are met without over-committing resources. In other words, the management mechanisms should ensure that the aggregate resource consumption does not exceed the quota limit while simultaneously ensuring that instances running on particular nodes are not starved when the total consumption is below the quota limit. To this end, quota allocation mechanisms should know the current allocation across nodes and be able to use this information to intelligently apply algorithms to better distribute resources to instances according to their runtime requirements [115, 149, 154]. The goal is to avoid both under- and over-utilization of quotas.

On the other hand, *resource usage* management systems are tools for collecting and aggregating data about resources consumed by services across clusters in order to establish a single consistent global view for purposes such as accounting and billing [47, 48, 153, 152, 6], creating traces [132, 58, 133], and to provide input for provisioning [36, 8, 64, 83, 129, 134, 143, 76, 45, 93]. Resource consumption data should be monitored continuously and must be collected and synchronized to maintain a single consistent view of the data generated across different clusters. However, any such synchronization mechanism must make a trade-off between the consistency of the data and the performance of the infrastructure: more frequent synchronization increases data consistency but reduces performance.

The following sections discuss quota and resource usage management in more detail.

2.1.1 Quota Control

Service owners and/or infrastructure providers may want to cap (put limits on) cloud resource quotas such as the total credit, storage quota, total number of CPUs, total CPU hours, or number of VMs, IP addresses, or network connections. To do this, the provider or owner must have a mechanism for allocating and enforcing limits across services running on multiple nodes [96, 79]. This is similar to the way that prepaid telecommunication services work [2, 1]: customers pay in advance and use the service afterwards. The mechanism is expected to provide the service to customers immediately as long as the paid-for budget is not exhausted.

Managing quota limits in large distributed systems is difficult because it requires the maintenance of a consistent picture of total usage while resources are being consumed concurrently at several locations. Quota management is essential for ensuring that services do not exceed their allocated (or paid-for) budget. It can also be used to enforce global allocation limits in order to solve problems such as spurious services that flood and overburden the system with dummy tasks, denying access to other services, and malicious services that launch Distributed Denial of Service (DDoS) attacks.

The resource consumption of services running on tens of thousands of nodes in many clusters must be properly monitored. Furthermore, quotas should be distributed across nodes or clusters optimally, with the objective of minimizing their associated overhead while meeting service demands.

2.1.2 Resource Usage Management

Resource usage management involves monitoring, collecting, and aggregating information on resource consumption in order to facilitate decision making in cloud environments [32, 128, 99, 126, 38]. It requires a transparent and efficient management mechanism that can be used to produce a consistent cloud-wide view of the data. The degree to which resource usage must be managed, i.e. the necessary update frequency and the desired granularity of the monitored statistical information, will vary according to the management task at hand.

Each service may use resources from multiple clusters and can generate a variety of runtime scenarios that, if not tracked and responded to, can make usage-based reporting and billing impossible. Resource usage must be monitored, synchronized, and aggregated to perform billing operations. Thus, for cloud services to be commercialized using a pay-as-you-go model, the cloud must allow runtime usage across clusters to be accurately measured, collected and synchronized.

Therefore, an accounting and billing system capable of monitoring, collecting, and processing usage data and metrics should be incorporated into the cloud architecture to support the business model of cloud providers. Once services are deployed to the cloud, it is critical to monitor their functions and track their resource usage on an ongoing basis in order to determine how much service

owners are to be charged. Timely and consistent collection and distribution of accounting data is not only important for the infrastructure provider's billing operations: service owners may also need to bill their customers on the basis of resource consumption. Similar situations occur in the supply chains of manufacturing industries [97, 140], where a faulty material or a delay in the supply of a component will affect all of the downstream processes in the chain. Likewise, a delay or a failure in infrastructure monitoring will affect all members of the ecosystem.

2.2 Resource Provisioning

A cloud is a complex system with a very large number of shared resources whose performance and capabilities may be limited by the sometimes unpredictable behavior of the infrastructure and the services running on top. At the same time, services hosted in the cloud have become indispensable for many business and personal purposes, making the performance of these services a key issue [66, 39, 75, 73, 41, 135, 157]. Resource provisioning techniques help to determine the optimal amount of physical resources required to satisfy service demands. Today, cloud infrastructure providers either do not offer any performance guarantee or prefer coarse-grained and static resource provisioning with fixed sizes, resulting in inefficient resource utilization and Service Level Agreement (SLA) violations. Infrastructure providers will need to provide better and more stringent performance guarantees with fine-grained resource provisioning techniques to attract businesses to move their core services into the cloud.

Resource provisioning has a direct influence on the performance of services in the short-term and the survival of the infrastructure provider in the long-term. It is important to have autonomic techniques that continuously adjust services' resources at runtime in order to provide performance guarantees at all times in an environment where frequent and unpredictable changes are the norm rather than the exception. Indeed, autonomic resource provisioning technique is a necessity rather than a luxury due to the scale and complexity of cloud infrastructures, the huge number of services deployed on them, and the unpredictability of runtime changes in services' capacity demands (which may produce sudden demand spikes known as flash crowds) and operating environments, i.e. hardware failures [83, 107, 81, 76, 65, 44].

Ideally, the infrastructure provider should have a resource provisioning mechanism that maximizes resource utilization while using dynamic and fine-grained resource allocation to guarantee that services' performance requirements are met. Thus, the task of resource provisioning is to allocate services enough resources to meet their performance requirements, but no more. Unfortunately, this is very difficult to do in practice because of the dynamic nature of cloud environments, the need for supporting heterogeneous services with different performance requirements, the unpredictable nature of cloud workloads, the complexity of and differences between the KPIs used by each service, and the

non-triviality of mapping performance measurements into resources [49, 17, 25, 135, 157].

Given a particular desired service performance, the goal is to determine the optimal amount of hardware resources required based on the services' runtime behavior. Specifically, service owners should be able to specify their service resource requirements in terms of their preferred KPIs (e.g. response time, throughput, CPU and memory utilization, etc) and have the resource provisioning techniques translate those into optimal resource allocations that satisfy the SLO for each service.

Chapter 3

Resource Usage and Quota Management

Infrastructure providers find themselves in an era where customers expect their services to be delivered in a way that meets certain requirements with respect to performance, capacity, cost and geographical distribution. Recently, there has been considerable interest in using software driven infrastructure management [57, 101, 113, 78] to manage cloud infrastructures and seamlessly deliver resources to customers' services. The idea behind software driven infrastructure management is to have a software suite that provides a comprehensive abstraction of a complete large-scale distributed infrastructure similar to the way that an operating system abstracts the different components of a computer. However, traditional management mechanisms, tools and techniques cannot meet the requirements of software driven infrastructures because of their limited scaling [68, 155, 113, 78].

Software driven management of infrastructures is rapidly becoming a new target for IT enablement, but this concept will only be fully realized through the development of improved management mechanisms.

As discussed in Chapter 2, successful quota and resource usage management of services at the cloud scale requires a rich set of global information and global control for each individual service running across the infrastructure's nodes. As the number of nodes and the services' size increase, the overhead of collecting, analyzing, and acting upon the associated data grows. The basic features expected from the chosen approach are:

- *Accuracy:* Resources should be monitored, collected, aggregated or controlled accurately without being over- or under-stated.
- *Scalable:* The mechanism should remain functional and viable as the number of services and nodes increases.
- *Fault tolerance:* Partial failure of the mechanism should not bring the

whole infrastructure to a halt.

- *Less overhead:* Resources consumed by services should be monitored and collected in a way that interferes minimally with hosted services.

The management layer can be designed in centralized or decentralized fashion as described below. Each approach has its own advantages and disadvantages and is affected by a number of factors such as the size of the managed entities (measured in terms of, e.g., the number of nodes or services) and the degree of geographic dispersion.

3.1 Centralized Management

In a centralized management system, a centralized manager keeps track of all available resources, collects usage from different managed entities and makes decisions. In general the management functions and decision-making are concentrated at a central component. All managed entities send messages requesting resources and reporting resource usage to the manager. The manager is responsible for allocating resources to services and storing and processing resource usage data. With such centralized designs, few decisions need be made about the allocations and usage of resources.

The majority of cloud computing platforms rely on centralized architectures combined with a hierarchical system of control [19, 52, 67]. The main advantage of this scheme is simplicity. Moreover, it eases consistency concerns and avoids conflicting decisions by providing central control over the whole system.

However, centralized management presents the well-known problems of single point of failure and single point of congestion. Moreover, the centralized manager is burdened with multiple decisions for all managed entities and a lot of communication needs to take place from different nodes hindering the performance of the whole system. Thus, as the system size increases, the centralized manager gets heavily loaded and becomes a bottleneck. It may also limit concurrency. In general, it suffers from poor availability, lack of fault tolerance, and limited capacity for scaling.

3.2 Decentralized Management

In recent years, there has been an increased and wide interest in decentralized management techniques and cooperative distributed decision making [125, 124, 94]. The central notion of decentralized management is to pursue full local autonomy while cooperating (by communicating) to achieve a global goal. The ideal is that individual management units are able to acquire information about the state of the entire system by communicating their local information to all others. However, fine-grained information about the entire system is generally not available due to the vast amount of local information that may be

generated; if all of this information were transmitted in its entirety, it would cause significant network congestion. In addition, the information received may not be up-to-date due to delays.

Decentralized management disperses the decision-making process over multiple management components so as to alleviate the problems encountered with a centralized approach. Managers at specific locations make key decisions relating to their sphere of responsibility and cooperate with peers as necessary.

In such an architecture, all of the peers involved in a management task play similar roles, interacting cooperatively without any one having a distinct role. The aim of this scheme is that each management unit performs its responsibilities independently without being hampered by the state of others, and only interacts with others when it is not possible to achieve the global management goal locally. Decentralized management can also abstract the differences between managed entities in different locations without limiting their capabilities. It is thus a viable method for achieving a stable growth pattern in an era of rapid technological development and investment, which allows each peer to have different internal management policies and mechanisms.

The main characteristics of a decentralized approach are:

- *Increased Availability*: Availability refers to the accessibility of a system in the presence of failure. For a centralized design, if the central management or the cluster where the management is located fails, the service becomes unavailable for other clusters as well. In contrast, with decentralized management the failure of one or more clusters does not prevent the remaining clusters from providing the service among themselves.

- *Fault tolerance*: Fault tolerance is the ability to function in the presence of component failures without performing incorrect actions. With decentralized management, the failure of one or more clusters can be tolerated as it only affects those services running on the failed clusters.

- *Enhanced performance*: Since services are running at multiple clusters, requests for different services do not have to line up at one management component; instead they can line up at multiple management components located on different clusters to reduce their waiting time.

- *Better Scalability*: In general, a decentralized management scales well as the sizes of services and clusters grow.

- *Greater Autonomy*: The decentralization approach gives individual clusters autonomy in making their own decisions. A decentralized approach assumes that each entity is autonomous and self-controlled, making its management decisions based on its own policies. Local management units need only coordinate with other clusters when local information is not sufficient for making decisions.

There is a growing trend towards using decentralized management approaches to support management demands in different computing domains because the

current centralized model is increasingly failing to achieve its goals. For example, decentralized techniques have been used in quota management across nodes inside a LAN [110, 82, 22, 21, 79, 110] as well as for network management [20, 96, 114, 149, 80, 42, 5]. The goal of this thesis is to apply decentralized techniques to the quotas and resource usage of services running on multiple clusters.

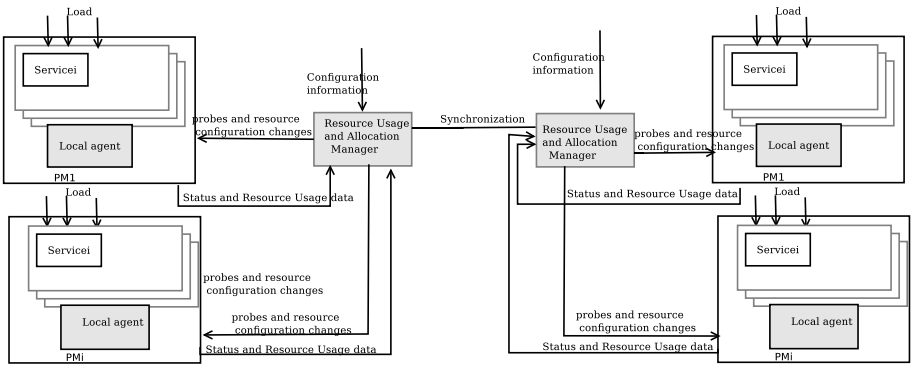

Figure 2: A decentralized management approach with distributed management entities.

Figure 2 shows the architecture of a decentralized management system. Management components interact with all or part of other management components in different situations. The arrows in the figure indicate interactions among management units. Some management units can work alone without interacting with other peers as long as their decisions can be made entirely locally.

A hybrid management approach can also be used [29, 51, 56] by incorporating some features of both the centralized and the decentralized approaches. Hybrid approaches generally rely on a central unit, the coordinator, that coordinates the actions of other units. The coordinator is responsible for performing global operations such as processing requests from other management units and making global decisions. The other management units control local resources and report status information to the coordinator. Moreover, each management unit can make local decisions and may contact the coordinator only when unable to do so. This approach may suffer from similar issues as centralized approaches due to the reliance on a single coordinator.

Chapter 4

Resource Provisioning

Virtualization technologies are among the key enablers of cloud computing. They simplify some resource management tasks such as the efficient utilization and sharing of resources by consolidating diverse enterprise services, each wrapped inside a Virtual Machine (VM) on a single Physical Machine (PM). These technologies enable resources such as CPU cycles, memory, secondary storage, and I/O and communication bandwidth, to be added or removed at runtime from one service to another on demand.

Nonetheless, current virtualization technologies are inadequate for determining resource demands and assuring performance guarantees for hosted services. Moreover, hosted services are likely to manifest emergent behaviors that lead to unhealthy resource contentions and undesirable performance interference as a result of co-location [151, 30]. For example, the performance of one service may be affected by demand changes in other co-located services. Moreover, the infrastructure may be overloaded due to sudden changes in services' capacity demand – flash crowds – and operating environments – hardware failures [83, 107, 81, 76, 65, 44].

Consequently, most existing cloud services are targeted at consumers who have low Quality of Service (QoS) expectations. However, due to its potential economic benefits, increasing numbers of enterprises wish to exploit the advantages of the cloud computing model such as pay-per-use pricing and rapid elasticity. To retain existing customers and attract new ones, existing virtualization technologies must be augmented with techniques that capture the runtime behavior of hosted services and determine the optimal resource allocation pattern for meeting each service's SLO. Specifically, mathematical and statistical models and techniques are needed to correlate resources and runtime performance behaviors in response to external changes such as changes in workloads or the infrastructure.

For the purpose of discussion, resource provisioning can be categorized into *Capacity-based* and *Performance-based* provisioning. Capacity-based approaches use resource utilization (e.g., CPU and memory utilization) when making re-

source provisioning decisions while performance-based provisioning uses metrics extracted from services (e.g. response time, throughput, number of requests). The sections below provide brief discussions about the two resource provisioning approaches as well as the management of capacity shortages.

4.1 Capacity-Based Provisioning

Cloud providers use resource utilization to guide their resource management [91, 145, 131]: resource utilization data are used to predict services' resource requirements. Efficient resource provisioning that accounts for resource utilization reduces operating costs, for example by allowing providers to consolidate VMs onto a small group of machines so that other machines can be shut down to reduce energy consumption [139, 104, 37, 50]. Consolidation is an inherently effective method for increasing resource utilization, and in turn reduces energy consumption. It can also free up resources for use by other services. However, utilization-based decisions may lead to SLA violations as they are oblivious to service performance.

4.2 Performance-Based Provisioning

A number of studies have suggested that end-users are sensitive to performance changes. For example, Amazon loses 1% of its sales for every 100ms of latency [3] while an extra half second's delay in search page generation reduces Google's traffic by 20% [62]. A broker could lose $4 million in revenues per millisecond if their service was 5 milliseconds behind the competition [4]. In general, several studies have shown that end-users will abandon an e-commerce service for a competitor if its response time is above 4 seconds [103], thus incurring long-term revenue loss. In the same manner, video streaming services can also expect users to abandon their services if the expected level of throughput drops.

All these studies demonstrate the importance of considering performance during resource provisioning. Capacity-based approaches are inadequate to ensure performance because they are oblivious to the observed performance of services. However, performance-based resource provisioning is challenging due to the non-trivial relationship between performance and capacity, and the unpredictable nature of incoming workloads.

4.3 Capacity Shortage Management

A challenging problem in cloud infrastructure management is how to deal with short- or long-term capacity shortage management, i.e. situations in which the aggregate demand for resources exceeds the resources available in the infrastructure. Several techniques such as elasticity, replication, migration, and load balancing have been suggested to mitigate this issue as long as there

is spare capacity in other hosts [15, 27, 129]. However, it is economically infeasible to reserve enough spare capacity since unexpected spikes can increase resource demand five-fold [31]. Moreover, some services may not tolerate any disruption at all. Because of this, most cloud services have targeted consumers with low performance expectations as noted above, and enterprises remain hesitant to move core business services into the cloud. Enterprises will require guaranteed performance under all conditions, expressed in terms of appropriate Key Performance Indicator (KPI) metrics (e.g. response time or throughput), in order to be confident in moving key services into the cloud.

Service differentiation schemes that decide which services to degrade can be used to address short-term capacity shortage problems [127, 119, 14, 92, 136, 98, 28, 147, 40, 158, 107]. Specifically, during service differentiation decisions, services with stringent performance needs are given higher priority than services with relatively lower performance requirements. Each service can be placed into different SLA classes (e.g., gold, silver, and bronze), signifying their relative importance. Each class is assigned a penalty weight that stipulates the relative importance of the corresponding services. This makes it possible to preferentially maintain the SLOs of higher class services by shifting resources from less performance-sensitive ones during periods of infrastructure overload.

Fig. 3 shows a logical view of a high-level architecture for an autonomic service and resource manager that would allocate the optimal resources required for each service deployed in the cloud. Note that the realization of the manager can be centralized or distributed. In general, the manager can make decisions concerning vertical and horizontal scaling of VMs, VM migration, load balancing, and service priority so as to meet service's SLO requirements and ensure efficient resource utilization. The focus in this thesis is on performance-based and fine-grained resource provisioning by resizing VMs (i.e. vertical scaling) to ensure that services' performance requirements are met. In the event that the available resources are temporarily insufficient to satisfy the demand, service differentiation is applied in order to guarantee the performance of critical services.

The following features would be expected of an autonomous resource manager:

1. *Autonomic resource provisioning.* A self-adaptive resource provisioning that dynamically adjusts resources to services based on their workload dynamics and relative importance without requiring human intervention should be employed.

2. *Timely detection of changes.* The changes in the workload behavior as well as in the infrastructure should be detected immediately in order to react to them accordingly.

3. *Fast computation.* Fast computation is needed in order to react immediately to changes in the services' performance behaviors.

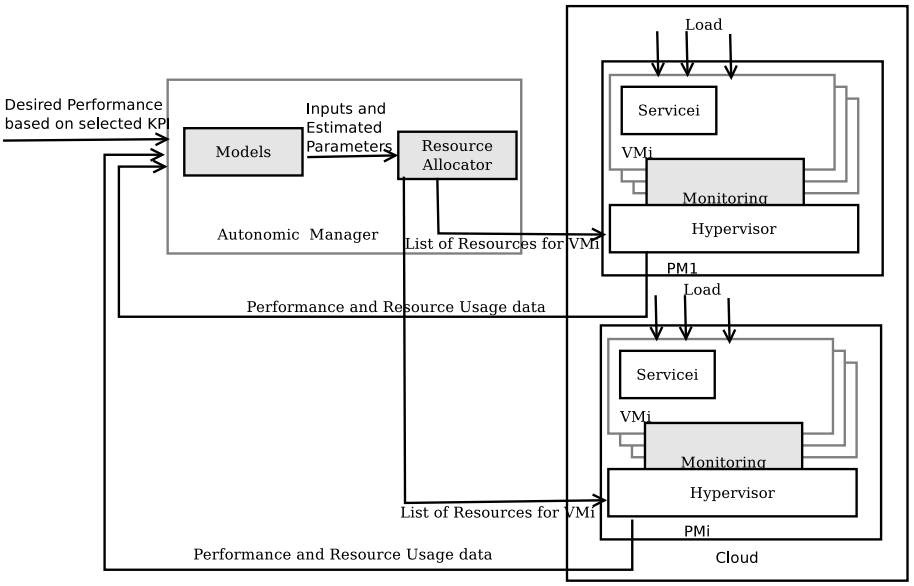

Figure 3: High-level architecture of an autonomic services and resources manager for cloud environments.

4. *Guaranteed performance.* The resources allocated should be exactly sufficient to satisfy the demand. That is to say, services should be provided with enough capacity to meet their performance targets, but no more. As a result, capacity over-provisioning or under-provisioning should be avoided under normal conditions.

5. *Service prioritization.* Capacity shortages may happen at any time in complex systems such as cloud infrastructures. Thus, when the capacity demand exceeds the supply, resources should be allocated based on service priorities using their performance as an indicator. Such priority schemes should be independent of services' KPIs and should ensure that the performances of higher priority services is less affected than their low priority counterparts.

Chapter 5

Summary of Contributions

This section presents the contributions of the thesis. Fig. 4 provides a conceptual view of these contributions. At the top are infrastructure-wide *quota enforcement* mechanisms which ensure that hosted services run without issue as long as the global quota is available. At the bottom, a non-intrusive *usage accounting* mechanism collects, aggregates and synchronizes the resource usage of services across the infrastructure to facilitate timely billing operations. The *autonomic resource manager*, deployed at each node, implements various techniques to provision resources according to services' demands.

Papers I, II and III present highly *scalable* solutions for managing *quota enforcement* and *usage accounting* efficiently and consistently. The proposed solutions scale with the size of the infrastructure as well as the services deployed on top of it. The proposed solutions are also *robust* since instance failures are localized and the system continues to operate and deliver useful service in the event of a local failure.

Papers IV–VIII present models and techniques that map service performance requirements onto physical resources. The proposed solutions continuously adjust the allocation of resources to meet the services' performance requirements irrespective of workload variation. Moreover, the proposed mechanism guarantees to meet the performance requirements of critical services during capacity shortages by shifting resources from less critical services.

5.1 Distributed Quota Enforcement

5.1.1 Paper I

Paper I [85] focuses on how to manage quota for services running in geographically distributed clusters. The paper presents a fully distributed quota management scheme for tracking and distributing quota shares across distributed clusters. The solution monitors resource consumption by services that are spread over a number of clusters with the goal of triggering global polls

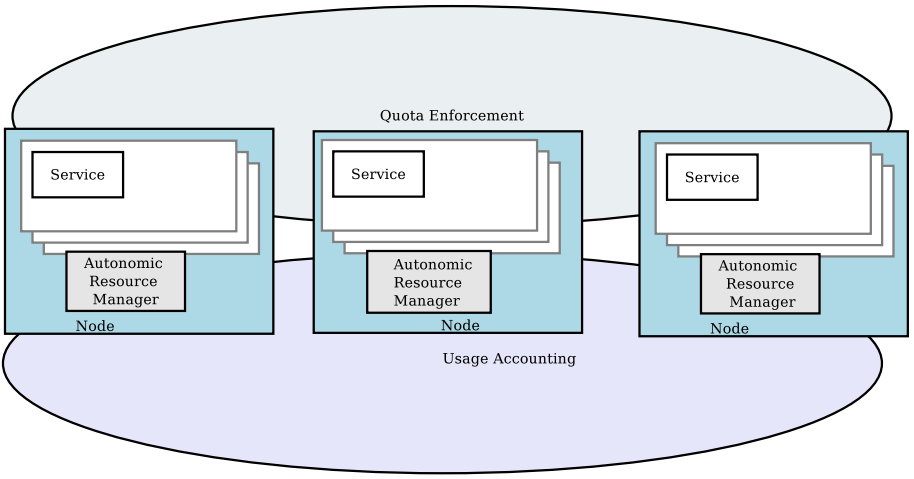

Figure 4: Pictorial representation of the contributions.

only when the allocated balance in a cluster decreases below a threshold and allocations are reassigned in a way that avoids further immediate global polls.

Unlike a single coordinator or predefined coordinator, the paper proposes a distributed scheme that selects a per-event coordinator when needed. For each allocation event, the coordinator is selected and the clusters currently running that service form a multi-cast group to perform the allocation algorithm.

It achieves scalability by minimizing global message exchanges, increases performance by distributing requests, and improves availability by avoiding a single point of failure. A range of simulations were performed and the communication cost of the fully distributed allocation approach is compared to that for a reference centralized resource manager algorithm, showing that the distributed approach is more efficient and effective.

5.1.2 Paper II

Paper II [89] extends the work presented in Paper I by adding support for multiple parallel decisions by arranging nodes using a spanning tree. Querying every node during quota redistribution decision causes latency and communication overheads as the number of nodes and the distances between them increase. Here, we address the latency and communication overhead problems by proposing a tree-based protocol for enforcing quotas in clouds. In our protocol, all nodes running a service maintain a local quota allocation so that permissions can be given immediately provided that the local allocation is not exhausted. When local quota allocations are exhausted, nodes request extra quota from nearby neighbors progressively instead of polling all nodes.

5.2 Distributed Resource Usage Management

5.2.1 Paper III

Paper III [88] presents monitoring, data-collection, and synchronization mechanisms in multi-cluster environments so as to facilitate accounting and billing operations.

With today's large-scale, geographically distributed clusters, running customers' services across multiple clusters is a possibility. As a result services generate huge amounts of accounting records that are dispersed throughout all clusters. Available accounting systems that are centralized cannot efficiently manage these huge amounts of usage records generated across multiple clusters. To cope with this problem, the paper proposes a new approach for monitoring, collecting and synchronizing accounting records in a decentralized fashion that provides a desired level of scalability.

A distributed mechanism is employed for collecting and synchronizing accounting records across a number of geographically distributed clusters as well as serving accounting requests from any of the clusters in a consistent manner. The mechanism collects and merges accounting records generated from all clusters and maintains the intrinsic requirements of accounting.

5.3 Performance-based provisioning

Capacity-based resource provisioning systems where resources are allocated based on utilization thresholds are oblivious to the observed performance of services and cannot be readily used to meet the performance needs of services expressed using KPIs such as response time or throughput. Papers IV–VI address this issue.

5.3.1 Paper IV

Paper IV [86] presents two generic performance models– the previously proposed queue length model and the novel inverse model– for mean response time that map performance to capacity in order to provide performance guarantees for interactive services deployed in the cloud. We carried out an extensive set of experiments using RUBiS, RUBBoS, and Olio–three widely used interactive cloud benchmarking applications– with varying workload mixes over time under both closed- and open-system models [120]. We also varied the target response time of each application to see how the models behave. The results demonstrate that the two models are stable for higher response time targets. However, our inverse model exhibited greater stability than the queue length model for lower targets.

5.3.2 Paper V

Paper V [86] presents two novel tail response time performance models– the queue length tail model and the inverse tail model– for predicting the capacity required to guarantee tail response times for interactive services deployed in the cloud. The two models were able to allocate just the right amount of capacity to meet tail response time targets expressed in percentiles (e.g. 95% or 99%) while avoiding substantial capacity over- or under-provisioning.

Both models were evaluated in an extensive set of experiments using RUBiS, RUBBoS, and Olio with real workloads and synthetic workload mixes that varied over time under both closed- and open-system model. We also varied the target response times of each application to see how this affected the models' behavior. Our results demonstrate that both tail response time models are stable under both more predictable and unpredictable real workloads and synthetic workloads generated using open- and closed-system models [120]. However, the inverse tail response time model is more stable than the queue length tail response time model.

5.3.3 Paper VI

In paper VI [55], we proposed an autonomic resource controller consisting of three sub-controllers– a *fuzzy controller*, a *CPU controller*, and a *memory controller*– to allocate the right amount of CPU and memory in order to meet mean response time targets for interactive services. The *fuzzy controller* acts as a coordinator, ensuring that the control actions of the cpu and memory controllers complement each other in order to fulfill the service's performance requirements, which were expressed in terms of mean response times. The *CPU controller* and *memory controller* allocate the right amount of CPU and memory, respectively, using the inputs provided by the *fuzzy controller*. In general, the proposed fuzzy control approach can be used as a coordination technique for distributed controllers. We evaluated the proposed solution with RUBiS, RUBBoS, and Olio in a virtualized environment using Xen Hypervisor. Different experiments were conducted under workload traces generated based on open and closed system models. The results show that the proposed coordination solution was able to maintain the target response time with while achieving high resource utilization.

5.4 Performance-based service differentiation

5.4.1 Paper VII

One of the many advantages of cloud computing is the ability to consolidate multiple services into a limited number of servers so as to achieve high datacenter utilization. However, maintaining high utilization while meeting SLOs is difficult, as a datacenter may become overloaded, reducing the performance of the hosted

services. Service differentiation has been proposed as a way of controlling which services get degraded. In paper VII [87], we propose performance-based service differentiation where capacity is distributed among services based on their observed performance and sensitivity to performance degradation. Specifically, when enough capacity is available, each service is automatically allocated the optimal amount of capacity to meet its target performance, expressed in terms of either response time or throughput. For cases when the available capacity is not sufficient, we propose two service differentiation schemes that dynamically decide which services to degrade and to what extent. We carried out an extensive set of experiments using different services– interactive as well as non-interactive– in which the services' workload mixes were varied over time. The results demonstrate that our solution precisely provides guaranteed performance or service differentiation depending on available capacity.

5.4.2 Paper VIII

Cloud storage is increasingly being adopted by users as simplified storage systems become available. These systems are mostly presented as Object Storage Systems (OSSs), hiding issues such as redundancy from users. As new industries are considering adopting clouds for storage, OSSs must evolve to support new needs. Among the most challenging is assuring guaranteed performance. In paper VIII [137], we present Controllable Trade-offs (CTO), an OSS-agnostic solution for providing performance guarantees. CTO presents itself as a thin layer that mediates requests between the user and the OSS. For generic support, performance is controlled by tuning the rejection probability based on priorities associated to each customer. Results show that CTO may reduce penalties by a factor of 3.23 on average and by a factor of up to 68 when the load is high.

Bibliography

[1] World telecommunication development report 1999, 1999. Available online: https://www.itu.int/ITU-D/ict/publications/wtdr_99/material/wtdr99s.pdf, Visited 2015-04-05.

[2] bcgi-anula report, 2001. Available online: http://web.archive.org/web/20080414071934/http://www.bcgi.net/assets/pdf/annual/2001.pdf, Visited 2015-04-05.

[3] Amazon found every 100ms of latency cost them 1 Available online: http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/, Visited 2015-04-06.

[4] The value of a millisecond: finding the optimal speed of a trading infrastructure., 2008. Available online: http://www.tabbgroup.com/PublicationDetail.aspx?PublicationID=346, Visited 2015-04-07.

[5] G. Aceto, A. Botta, W. De Donato, and A. Pescapè. Survey cloud monitoring: A survey. *Comput. Netw.*, 57(9):2093–2115, June 2013.

[6] V. Agarwal, N. Karnik, and A. Kumar. Metering and accounting for composite e-services. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pages 35–39, June 2003.

[7] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. The Resource-as-a-service (RaaS) Cloud. In *Proceedings of the 4th conference on Hot Topics in Cloud Ccomputing*, page 12, 2012.

[8] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 74–89, New York, NY, USA, 2003. ACM.

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

[10] V. Allee. Value network analysis and value conversion of tangible and intangible assets. 9(1):5–24, 2008.

[11] C. Almond. A practical guide to cloud computing security. *A white paper from Accenture and Microsoft*, 2009.

[12] Amazon. What is cloud computing?, Accessed: April, 2013. http://aws.amazon.com/what-is-cloud-computing/.

[13] M. Andreolini, S. Casolari, and M. Colajanni. Autonomic request management algorithms for geographically distributed internet-based systems. In *SASO*, 2008.

[14] M. Andreolini et al. A cluster-based web system providing differentiated and guaranteed services. *Cluster Computing*, 7(1):7–19, Jan. 2004.

[15] A.-F. Antonescu et al. Dynamic SLA management with forecasting using multi-objective optimization. In *Integrated Network Management (IM)*, pages 457–463. IEEE, 2013.

[16] Apache. Apache mesos, Accessed: April, 2013. http://mesos.apache.org/.

[17] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[18] D. Armstrong, D. Espling, J. Tordsson, K. Djemame, and E. Elmroth. Runtime virtual machine recontextualization for clouds. In *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*, pages 567–576. Springer Berlin Heidelberg, 2013.

[19] O. Babaoglu, M. Marzolla, and M. Tamburini. Design and implementation of a P2P cloud system. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 412–417. ACM, 2012.

[20] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 28–39, New York, NY, USA, 2003. ACM.

[21] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *Web Services, 2006. ICWS '06. International Conference on*, pages 63–71, Sept 2006.

[22] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In *ICSOC 2005, Third International Conference of Service-Oriented Computing, volume 3826 of Lecture Notes in Computer Science*, pages 269–282. Springer, 2005.

[23] P. Barham, B. Dragovic, K. Fraser, S. H, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating Systems Principles*, pages 164–177, 2003.

[24] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *Micro, IEEE*, 23(2):22–28, 2003.

[25] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, Dec. 2007.

[26] M. E. Barth and G. Clinch. Revalued financial, tangible, and intangible assets: Associations with share prices and non-market-based value estimates. *Journal of Accounting Research*, 36:199–233, 1998.

[27] A. Beloglazov and R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Trans. Parallel Distrib. Syst.*, 24(7), July 2013.

[28] N. Bhatti and R. Friedrich. Web server support for tiered services. *Network, IEEE*, 13(5):64–71, Sep 1999.

[29] L. Bin, S. Wenxiao, and L. Na. A hierarchical semi-centralized architecture for load balancing of heterogeneous wireless networks. In *Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing - Volume 02*, NSWCTC '10, pages 28–31, 2010.

[30] M. Björkqvist et al. QoS-aware service VM provisioning in clouds: Experiences, models, and cost analysis. In *ICSOC*, volume 8274 of *LNCS*, pages 69–83. Springer, 2013.

[31] P. Bodik et al. Characterizing, modeling, and generating workload spikes for stateful services. In *SoCC*, pages 241–252, 2010.

[32] R. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao. NIST cloud computing reference architecture. In *2011 IEEE World Congress on Services (SERVICES)*, pages 594–596, 2011.

[33] R. Buyya, R. N. Calheiros, and X. Li. Autonomic cloud computing: Open challenges and architectural elements. In *in IEEE International Conference on Emerging Applications of Information Technology*, pages 3–10, 2012.

[34] M. Caballer, I. Blanquer, G. Moltó, and C. de Alfonso. Dynamic management of virtual infrastructures. *Journal of Grid Computing*, 13(1):53–70, 2014.

[35] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: Easy and efficient parallel processing of massive data sets. *Proc. VLDB Endow.*, 1(2):1265–1276, Aug. 2008.

[36] F. Chang, J. Ren, and R. Viswanathan. Optimal resource allocation in clouds. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 418–425, 2010.

[37] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev.*, 35(5):103–116, Oct. 2001.

[38] M. Chen, A. Geist, D. Bernholdt, K. Chanchio, and D. Million. The design and prototype of ruda, a distributed grid accounting system. In *Computational Science and Its Applications – ICCSA 2005*, volume 3482 of *Lecture Notes in Computer Science*, pages 29–38. Springer Berlin Heidelberg, 2005.

[39] S. Chen, D. Levy, and J. Yao. Accountability for service compliance: A survey. *Int. J. Syst. Serv.-Oriented Eng.*, 3(1):16–43, 2012.

[40] X. Chen et al. Aces: An efficient admission control scheme for qos-aware web servers. *Comput. Commun.*, 26(14):1581–1593, Sept. 2003.

[41] Y. Chen et al. SLA decomposition: Translating service level objectives to system level thresholds. In *ICAC*. IEEE, 2007.

[42] S. Clayman, A. Galis, and L. Mamatas. Monitoring virtual networks with lattice. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 239–246, April 2010.

[43] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. *Distributed Systems: Concepts and Design (5th Edition)*. Addison Wesley, 5 edition, May 2011.

[44] F. A. de Oliveira, T. Ledoux, and R. Sharrock. A framework for the coordination of multiple autonomic managers in cloud environments. In *SASO*, pages 179–188, 2013.

[45] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics with Application to the Apache Web Server. In *Network Operations and Management Symposium*, pages 219–234, 2002.

[46] E. Elmroth, P. Gardfjäll, O. Mulmo, and T. Sandholm. An ogsa-based bank service for grid accounting systems. In *Applied Parallel Computing. State of the Art in Scientific Computing*, volume 3732 of *Lecture Notes in Computer Science*, pages 1051–1060. Springer Berlin Heidelberg, 2006.

[47] E. Elmroth and D. Henriksson. Distributed usage logging for federated grids. *Future Gener. Comput. Syst.*, 26(8):1215–1225, Oct. 2010.

[48] E. Elmroth, F. G. Marquez, D. Henriksson, and D. P. Ferrera. Accounting and billing for federated cloud infrastructures. In *Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing*, GCC '09, pages 268–275. IEEE Computer Society, 2009.

[49] E. Elmroth, J. Tordsson, F. Hernández, A. Ali-Eldin, P. Svärd, M. Sedaghat, and W. Li. Self-management challenges for multi-cloud architectures. In *Proceedings of the 4th European Conference on Towards a Service-based Internet*, ServiceWave'11, pages 38–49. Springer-Verlag, 2011.

[50] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Proceedings of the 2Nd International Conference on Power-aware Computer Systems*, PACS'02, pages 179–197, Berlin, Heidelberg, 2003. Springer-Verlag.

[51] V. Emeakaroha, T. Ferreto, M. Netto, I. Brandic, and C. De Rose. Casvid: Application level monitoring for sla violation detection in clouds. In *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pages 499–508, 2012.

[52] P. T. Endo, A. V. de Almeida Palhares, N. C. V. N. Pereira, G. E. Gonçalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mångs. Resource allocation for distributed cloud: concepts and research challenges. *IEEE Network*, 25(4):42–46, 2011.

[53] D. Espling, L. Larsson, W. Li, J. Tordsson, and E. Elmroth. Modeling and placement of cloud services with internal structure. *Cloud Computing, IEEE Transactions on*, PP:1–1, 2014.

[54] F. Etro. The economic impact of cloud computing on business creation, employment and output in europe. *Review of Business and Economics*, 54(2):179–208, 2009.

[55] S. Farokhi, E. Lakew, C. Klein, I. Brandic, and E. Elmroth. Coordinating cpu and memory elasticity controllers to meet service response time constraints. In *The ACM Cloud and Autonomic Computing Conference (CAC'15)*, 2015. Accepted.

[56] D. Fesehaye, R. Malik, and K. Nahrstedt. Edfs: a semi-centralized efficient distributed file system. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 28:1–28:2, 2009.

[57] R. Fichera. The software-defined data center is the future of infrastructure architecture, Accessed: April, 2013. http://www.vmware.com/files/include/microsite/sddc/the_software-defined_datacenter.pdf.

[58] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica. X-trace: A pervasive network tracing framework. In *Proceedings of the 4th USENIX Conference on Networked Systems Design and Implementation*, pages 20–20, Berkeley, CA, USA, 2007. USENIX Association.

[59] T. Forell, D. Milojicic, and V. Talwar. Cloud management: Challenges and opportunities. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pages 881–889, 2011.

[60] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[61] P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm. Scalable grid-wide capacity allocation with the swegrid accounting system (sgas). *Concurr. Comput. : Pract. Exper.*, 20:2089–2122, December 2008.

[62] J. Grossklags and A. Acquisti. When 25 cents is too much: An experiment on willingness-to-sell and willingness-to-protect personal information. 2007.

[63] R. L. Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[64] T. Gschwind, K. Eshghi, P. Garg, and K. Wurster. Webmon: A performance profiler for web transactions. In *Advanced Issues of E-Commerce and Web-Based Information Systems, 2002. (WECWIS 2002). Proceedings. Fourth IEEE International Workshop on*, pages 171–176, 2002.

[65] S. Gueye, N. De Palma, and E. Rutten. Component-based Autonomic Managers for Coordination Control. In *Coordination Models and Languages*, pages 75–89, 2013.

[66] J. Guitart, J. Torres, and E. Ayguadé. A survey on performance management for internet applications. *Concurr. Comput. : Pract. Exper.*, 22:68–106, 2010.

[67] 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*, HotCloud'11, pages 3–3. USENIX Association, 2011.

[68] 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*, HotCloud'11, pages 3–3, Berkeley, CA, USA, 2011. USENIX Association.

[69] A. Gupta, E. Ababneh, R. Han, and E. Keller. Towards elastic operating systems. In *Conference on Hot Topics in Operating Systems*, page 16, 2013.

[70] J. R. Hamilton. An architecture for modular data centers. In *Third Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 306–313, 2007.

[71] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 295–308, Berkeley, CA, USA, 2011. USENIX Association.

[72] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[73] D. Huang, B. He, and C. Miao. A survey of resource management in multi-tier web applications. *Communications Surveys Tutorials, IEEE*, 16(3):1574–1590, Third 2014.

[74] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth. Performance anomaly detection and bottleneck identification. *ACM Comput. Surv.*, 48(1):4:1–4:35, July 2015.

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

[76] P. Jamshidi, A. Ahmad, and C. Pahl. Autonomic Resource Provisioning for Cloud-based Software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 95–104, 2014.

[77] A. Kamra, V. Misra, and E. M. Nahum. Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites. In *Quality of Service, IEEE International Workshop on*, pages 47–56. IEEE, 2004.

[78] G. Kandiraju, H. Franke, M. Williams, M. Steinder, and S. Black. Software defined infrastructures. *IBM Journal of Research and Development*, 58(2/3):2:1–2:13, March 2014.

[79] K. Karmon, L. Liss, and A. Schuster. GWiQ-P: an efficient decentralized grid-wide quota enforcement protocol. *14th IEEE International Symposium on High Performance Distributed Computing (HPDC14)*, pages 222 – 232, july 2005.

[80] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan. Efficient gossip-based aggregate computation. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pages 308–317, New York, NY, USA, 2006. ACM.

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

[82] S. Kikuchi, H. Shimamura, and Y. Kanna. Monitoring method of cross-sites' processes executed by multiple ws-bpel processors. In *9th IEEE International Conference on E-Commerce Technology (CEC 2007) / 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2007), 23-26 July 2007, National Center of Sciences, Tokyo, Japan*, pages 55–64, 2007.

[83] C. Klein et al. Brownout: Building more robust cloud applications. In *ICSE*, 2014.

[84] C. Kurmann, F. Rauch, and T. M. Stricker. Cost/performance tradeoffs in network interconnects for clusters of commodity pcs. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, pages 196.2–, 2003.

[85] E. Lakew, F. Hernandez-Rodriguez, L. Xu, and E. Elmroth. Management of distributed resource allocations in multi-cluster environments. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 275–284, 2012.

[86] E. Lakew, C. Klein, F. Hernandez-Rodriguez, and E. Elmroth. Tail response time modeling and control for interactive cloud services. Submitted for journal publication.

[87] E. Lakew, C. Klein, F. Hernandez-Rodriguez, and E. Elmroth. Performance-based service differentiation in clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 505–514, May 2015.

[88] E. Lakew, L. Xu, F. Hernandez-Rodriguez, E. Elmroth, and C. Pahl. A synchronization mechanism for cloud accounting systems. In *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*, pages 111–120, Sept 2014.

[89] E. Lakew, L. Xu, F. Hernandez-Rodriguez, E. Elmroth, and C. Pahl. A tree-based protocol for enforcing quotas in clouds. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 279–286, 2014.

[90] L. Larsson, D. Henriksson, and E. Elmroth. Scheduling and monitoring of internally structured services in cloud federations. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 173–178, June 2011.

[91] Y. Lee and A. Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[92] Y.-D. Lin et al. Multiple-resource request scheduling for differentiated QoS at website gateway. *Comput. Commun.*, 31(10), June 2008.

[93] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. L. Hellerstein, and S. Parekh. Online response time optimization of apache web server. In *Quality of Service—IWQoS 2003*, pages 461–478. Springer, 2003.

[94] M. Marzolla, O. Babaoglu, and F. Panzieri. Server consolidation in clouds through gossiping. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6, June 2011.

[95] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6), 2009.

[96] S. Meng, L. Liu, and T. Wang. State Monitoring in Cloud Datacenters. *IEEE Trans. on Knowl. and Data Eng.*, 23:1328–1344, September 2011.

[97] J. T. Mentzer, W. DeWitt, J. S. Keebler, S. Min, N. W. Nix, C. D. Smith, and Z. G. Zacharia. Defining supply chain management. *Journal of Business Logistics*, 22(2):1–25, 2001.

[98] M. Mesnier et al. Differentiated storage services. In *Symposium on Operating Systems Principles (SOSP)*, pages 57–70. ACM, 2011.

[99] A. Mihoob, C. Molina-Jimenez, and S. Shrivastava. A case for consumer centric resource accounting models. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 506–512, July 2010.

[100] R. Miller. Who has the most web servers?, Accessed: April, 2013. http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/.

[101] I. Monga, E. Pouyoul, and C. Guok. Software-defined networking for big-data science - architectural models from campus to the wan. In *Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, SCC '12, pages 1629–1635, Washington, DC, USA, 2012. IEEE Computer Society.

[102] F. Muhss, R. Neumann, and A. Schmietendorf. The commoditization of it services with cloud computing. 2011.

[103] F. F.-H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour and Information Technology*, 23(3), 2004.

[104] D. Niyato, S. Chaisiri, and L. B. Sung. Optimal power management for server farm to support green computing. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 84–91, May 2009.

[105] P.-O. Östberg and E. Elmroth. Increasing flexibility and abstracting complexity in service-based grid and cloud software. In *CLOSER*, pages 240–249. SciTePress, 2011.

[106] P.-O. Östberg and E. Elmroth. Gjmf - a composable service-oriented grid job management framework. *Future Gener. Comput. Syst.*, 29(1):144–157, Jan. 2013.

[107] P. Padala et al. Automated control of multiple virtualized resources. In *European Conference on Computer Systems (EuroSys)*. ACM, 2009.

[108] D. Petcu. Multi-Cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM, 2013.

[109] D. Petcu. Consuming Resources and Services from Multiple Clouds. *Journal of Grid Computing*, pages 1–25, 2014.

[110] K. T. Pollack, D. D. E. Long, R. A. Golding, R. A. Becker-Szendy, and B. Reed. Quota enforcement for high-performance distributed storage systems. *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, pages 72–86, 2007.

[111] G. J. Popek and R. P. Goldberg. Formal Requirements for Virtualizable Third-Generation Architectures. *Communications of the ACM*, 17(7):412–421, 1974.

[112] G. Quecke and W. Ziegler. MeSch - An Approach to Resource Management in a Distributed Environment. In *Grid Computing — GRID 2000*, volume 1971 of *Lecture Notes in Computer Science*, pages 47–54. Springer Berlin Heidelberg, 2000.

[113] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker. Software-defined internet architecture: Decoupling architecture from infrastructure. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 43–48. ACM, 2012.

[114] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren. Cloud control with distributed rate limiting. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 337–348. ACM, 2007.

[115] A. Rai, R. Bhagwan, and S. Guha. Generalized resource allocation for the cloud. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 15:1–15:12. ACM, 2012.

[116] R. V. Renesse and K. Birman. Scalable management and data mining using astrolabe. In *International workshop on Peer-To-Peer Systems (IPTPS)*, 2002.

[117] G. P. Rodrigo Álvarez, P.-O. Östberg, E. Elmroth, and L. Ramakrishnan. A2l2: An application aware flexible hpc scheduling model for low-latency allocation. In *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*, VTDC '15, pages 11–19, New York, NY, USA, 2015. ACM.

[118] T. Sandholm, P. Gardfjäll, E. Elmroth, L. Johnsson, and O. Mulmo. An ogsa-based accounting system for allocation enforcement across hpc centers. In *Proceedings of the 2Nd International Conference on Service Oriented Computing*, ICSOC '04, pages 279–288, New York, NY, USA, 2004. ACM.

[119] N. Saxena et al. A new service classification strategy in hybrid scheduling to support differentiated QoS in wireless data networks. In *ICPP*, pages 389–396, 2005.

[120] B. Schroeder, A. Wierman, and M. Harchol-Balter. Open versus closed: A cautionary tale. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 18–18, Berkeley, CA, USA, 2006. USENIX Association.

[121] L. Schubert, K. G. Jeffery, and B. Neidecker-Lutz. *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010*. European Commission, 2010.

[122] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 351–364, New York, NY, USA, 2013. ACM.

[123] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 6:1–6:10, New York, NY, USA, 2013. ACM.

[124] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth. Autonomic resource allocation for cloud data centers: A peer to peer approach. In *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*, pages 131–140, Sept 2014.

[125] M. Sedaghat, F. Hernandez-Rodriguez, E. Elmroth, and S. Girdzijauskas. Divide the task, multiply the outcome: Cooperative vm consolidation. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 300–305, Dec 2014.

[126] V. Sekar and P. Maniatis. Verifiable resource accounting for cloud computing services. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 21–26, New York, NY, USA, 2011. ACM.

[127] A. W. Services. Amazon ec2 spot instances. Available online: http://aws.amazon.com/ec2/purchasing-options/spot-instances/.

[128] B. Sharma, R. Thulasiram, P. Thulasiraman, S. Garg, and R. Buyya. Pricing cloud compute commodities: A novel financial economic model. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 451–457, 2012.

[129] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. CloudScale: Elastic Resource Scaling for Multi-tenant Cloud Systems. In *2nd ACM Symposium on Cloud Computing*, page 5, 2011.

[130] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura. Cosmos: Computation offloading as a service for mobile devices. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '14, pages 287–296, New York, NY, USA, 2014. ACM.

[131] Y. Shi, X. Jiang, and K. Ye. An energy-efficient scheme for cloud resource provisioning based on cloudsim. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 595–599, 2011.

[132] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. Technical report, Google, Inc., 2010. http://research.google.com/archive/papers/dapper-2010-1.pdf.

[133] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. Technical report, Google, Inc., 2010.

[134] C. Stewart et al. Exploiting nonstationarity for performance prediction. In *EuroSys*. ACM, 2007.

[135] C. Stewart and K. Shen. Performance modeling and system management for multi-component online services. In *NSDI*, pages 71–84. USENIX, 2005.

[136] V. Sundaram et al. A practical learning-based approach for dynamic storage bandwidth allocation. In *IWQoS*, volume 2707 of *LNCS*, pages 479–497. Springer, 2003.

[137] R. Talyansky., E. Lakew, C. Klein, F. Hernández-Rodriguez, E. Levy, and E. Elmroth. On guaranteed performance over object storage systems. Submitted for publication.

[138] A. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms (2nd ed.)*. Prentice Hall, 2007.

[139] S. Tesfatsion, E. Wadbro, and J. Tordsson. A combined frequency scaling and application elasticity approach for energy-efficient cloud computing. *Sustainable Computing: Informatics and Systems*, 4(4):205 – 214, 2014. Special Issue on Energy Aware Resource Management and Scheduling (EARMS).

[140] D. J. Thomas and P. M. Griffin. Coordinated supply chain management. *European Journal of Operational Research*, 94(1):1 – 15, 1996.

[141] L. Tomas, P.-O. Östberg, B. Caminero, C. Carrion, and E. Elmroth. Addressing qos in grids through a fairshare meta-scheduling in-advance architecture. In *3PGCIC*, pages 226–233. IEEE, 2012.

[142] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst.*, 21(2):164–206, May 2003.

[143] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini. DejaVu: Accelerating Resource Allocation in Virtualized Environments. volume 40, pages 423–436. ACM, 2012.

[144] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.

[145] G. Velkoski, M. Simjanoska, S. Ristov, and M. Gusev. Cpu utilization in a multitenant cloud. In *EUROCON*, pages 242–249. IEEE, 2013.

[146] K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC*, pages 193–204. ACM, 2010.

[147] T. Voigt. Kernel mechanisms for service differentiation in overloaded web servers. In *Usenix Annual Technical Conference*, pages 189–202, 2001.

[148] H. Wu, A. N. Tantawi, and T. Yu. A Self-Optimizing Workload Management Solution for Cloud Applications. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 483–490. IEEE, 2013.

[149] F. Wuhib, M. Dam, and R. Stadler. Decentralized detection of global threshold crossings using aggregation trees. *Comput. Netw.*, 52(9):1745–1761, June 2008.

[150] H. Xingye, L. Xinming, and L. Yinpeng. Research on resource management for cloud computing based information system. In *Proceedings of the 2010 International Conference on Computational and Information Sciences*, pages 491–494, 2010.

[151] P. Xiong et al. vperfguard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments. In *The 4th SPEC International Conference on Performance Engineering*. ACM, 2013.

[152] L. Xu and E. Elmroth. A time interval-based credit reservation approach for prepaid composite services in cloud environments. In *Web Services (ECOWS), 2011 Ninth IEEE European Conference on*, pages 158–165, Sept 2011.

[153] L. Xu, E. Lakew, F. Hernandez-Rodriguez, and E. Elmroth. A scalable accounting solution for prepaid services in cloud systems. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 81–89, 2012.

[154] C.-T. Yang, K.-C. Wang, H.-Y. Cheng, C.-T. Kuo, and W.-C. Chu. Green power management with dynamic resource allocation for cloud virtual machines. In *2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, pages 726–733, 2011.

[155] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software-defined networking. *Communications Magazine, IEEE*, 51(2):136–141, 2013.

[156] S. Zhang, S. Zhang, X. Chen, and X. Huo. Cloud computing research and development trend. In *Proceedings of the 2010 Second International Conference on Future Networks*, ICFN '10, pages 93–97. IEEE Computer Society, 2010.

[157] X. Zhang, Y. Qu, and L. Xiao. Improving distributed workload performance by sharing both cpu and memory resources. In *20th International Conference on Distributed Computing Systems*, pages 233–241, 2000.

[158] H. Zhu et al. Demand-driven service differentiation in cluster-based network servers. In *INFOCOM*, volume 2, pages 679–688, 2001.