Managing Resource Usage and Allocations in Multi-Cluster Clouds

Ewnetu Bayuh Lakew

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

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

ewnetu@cs.umu.se

ISBN 978-91-7459-687-8
ISSN 0348-0542
UMINF 13.16

Printed by Print & Media, Umeå University, 2013

Abstract

The emergence of large-scale Internet services has fueled a trend toward large-scale systems composed of geographically distributed clusters. Managing resource allocations and resource usage is an important task for such services.

Resource allocations and resource usage management mechanisms for services running across clusters play vital roles in the performance of the entire system, economical sustainability of the provider, and level of customers satisfaction provided by the system. However, when providing the utmost customer satisfaction the service provider ought to make sure not to over-commit resources beyond the agreed limit between the customer and the provider. Moreover, statistics of resources consumed by different services should be monitored and collected using an efficient mechanism with minimal overhead and interference on the system and the services. Thus, resource usage collection and allocations mechanisms should impose economical constraints to both sides, the customer and the cloud provider.

This thesis focuses on decentralized resource allocation and resource usage management for services running in multi cluster environments. Theoretical as well as experimental results indicate that our proposed approaches provide efficient management of resources for services running in a large-scale geographically distributed systems.

Preface

This thesis contains an introduction to resource management in large-scale distributed systems, with focus on resource usage and allocations management, and the papers listed below.

Paper I E.B. Lakew, F. Hernandez-Rodriguez, L. Xu, and E. Elmroth. Management of Distributed Resource Allocations in Multi-cluster Environments. *The 31st IEEE International Performance Computing and Communications Conference (IPCCC 2012)*, 275–284, 2012.

Paper II L. Xu, E.B. Lakew, F. Hernandez-Rodriguez, and E. Elmroth. A Scalable Accounting Solution for Prepaid Services in Cloud Systems. *2012 IEEE Ninth International Conference on Service Computing (IEEE SCC 2012)*, 81–89, 2012.

Paper III E.B. Lakew, F. Hernandez-Rodriguez, L. Xu and E. Elmroth. A synchronization mechanism for cloud accounting systems. *To be submitted*, 2013.

In addition to the papers included in this thesis the following paper have been produced.

- E. K. Kolodner, S. Tal, D. Kyriazis, D. Naor, M. Allalouf, L. Bonelli, P. Brand, A. Eckert, E. Elmroth, S. V. Gogouvitis, D. Harnik, F. Hernandez, M. C. Jaeger, E. B. Lakew, J. M. Lopez, M. Lorenz, A. Messina, A. ShulmanPeleg, R. Talyansky, A. Voulodimos, and Y. Wolfsthal. A cloud environment for data-intensive storage services. *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, 357–366, Dec 2011.

This work has been funded in part by the European Community's Seventh Framework Programme (FP7/2001-2013) under grant agreement no. 257019 (VISION Cloud) and the Swedish Government's strategic effort eSSENCE.

Acknowledgments

I would like to express my deepest gratitude to my supervisor Erik Elmroth for accepting me into his group and for his moral support and constructive guidance throughout the work. It has been an honor and a privilege to work with him. A special appreciation goes to my co-supervisor Francisco Hernandez. I am thankful for his continuous support, unwavering patience and quality advice. The good advice and friendship have been invaluable on both academic and personal levels, for which I am extremely grateful. I would also like to express my deep appreciation to Stephen Hegner for his invaluable comments on the introduction part of the thesis.

I would like to thank all members of the distributed systems group for their constructive critique and collegial support towards my research work, which helped to improve the quality of this work.

I would not have contemplated this road if not for my parents, Etatey and Ebabey, who paved the road for me. To my parents, thank you. My siblings, have also been the best of friends along this journey especially Tesfaye Bayuh, my brother, who has spent all his whole life for the good of the family.

Last but not least, my deepest gratitude goes to my wife, Selome Kostentinos, and my daughter, Aenon, who gave me the strength and courage, and who have been with me through thick and thin.

አመሰግናለሁ !!

Umeå, June 2013
Ewnetu Bayuh Lakew

Contents

Chapter 1

Introduction

Internet technologies are transforming industries and our culture today at a rapid pace with no sign of slowing down. Today's Internet applications (hereafter services) are complex software systems consisting of a large number of individual applications which are integrated together to provide complex end-user services such as Web-based email, search, social networks, news, and e-commerce.

These services require so much computing and storage power that they are a natural fit for a massive computing infrastructures. The emergence of these large-scale Internet services has fueled a trend toward cloud computing.

The computing platform required for such large-scale services can consist of hundreds or thousands of commodity machines [5, 19, 27]. This number will continue to grow in the foreseeable future as the demand of services for more resources is growing [41].

The resources in these large-scale distributed systems are divided into multiple management units, termed clusters, that are geographically dispersed around the world. Each cluster, in turn, can consist of hundreds or even thousands of nodes.

The management of the infrastructures to efficiently provision resources to services is a challenging task. At this scale, there are always a significant number of servers and network components that fail at any given time. Besides, the need to integrate several heterogeneous environments introduces new levels of complexity. To a growing degree, services are expected to be self-configuring and self-managing, and as the range of permissible configurations grows, this undertaking becomes very complex.

Clusters should not be simply considered as heterogeneous collections of commodity hardware co-located and wired-up together in a single building. Rather, each cluster should be considered an aggregation of commodity machines in a seamless and transparent fashion that forms a single computing unit [5, 20]. This aggregation of thousands 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 fully realized through advancement in the software management layer [36].

The management layer is responsible for different tasks such as deployment of services to hardware resources, resource allocation management (enforcement of quotas), scheduling, monitoring, load balancing, resource usage collection and handling component failures, all without causing lots of overhead in the underlying system. This layer is also responsible for assuring performance, reliability, scalability, availability, latency, and fault tolerance requirements for each service.

The management layer itself must also be scalable, decentralized, fault tolerant and address the heterogeneity of underlying resources and services. Thus, the task of the management layer for a large-scale distributed system can be more complex than the services themselves. Consequently as the necessity of building complex, distributed services continues, building management tools for such systems becomes complex in itself [30, 33].

From the management tasks, this thesis, which consists of three papers, focuses on resource allocation and resource usage management for services running in multi-cluster environments. The goal is to monitor and manage resource allocations as well as collect and synchronize resource usage for services running across geographically distributed clusters in cloud computing environments.

Resource allocation management mechanisms are essential for controlling distributed shared resources so that services are not able to exceed their allocated or paid-for budget of credit, CPU hours, CPUs, RAM sizes, storage sizes, etc. Resource allocation management for a service not only needs mechanisms to control that the consumption of resources do not exceed certain allocation limits but also requires mechanisms for efficient and fair utilization and distribution of allocations among services running at different clusters. Thus, proper cloud wide monitoring and controlling of resource allocation must be in place in order not to over or under-provision resources. Papers I and II focus on managing allocations for services running across clusters.

Furthermore, run time monitoring data of each service should be collected and aggregated in order to maintain a single global state of the system which aids for further management activities such as accounting and billing, scheduling, load balancing, and network analysis. It is far more complicated to collect and aggregate information about all resources consumed across clusters as the management task itself may consume too much computing and network resources unless done properly. Paper III focuses on how to collect data across multiple clusters and provide consistency and synchronization mechanisms with the main goal of facilitating accounting and billing operations.

The thesis is organized as follows. Chapter 2 introduces the resource management problems that we address. Chapter 3 discusses the approach followed to address the problems. Chapter 4 summaries the contributions of this work. Chapter 5 outlines potential future directions. Finally, three papers produced in the course of the work are appended.

Chapter 2

Resource Management in Cloud Data centers

Recently, Internet computing technologies have emerged to provide large-scale computing farms based on utility computing service [1] provisioning driven by efficiency and scalability needs which are known as *cloud computing*. There are different definitions and concepts of cloud computing [1, 25, 35, 41]. According to NIST [25], cloud computing can be defined as a model for enabling convenient on-demand network access to a shared pool of configurable computing resources such as CPU, networks, storage, and memory, which can be quickly provisioned and released with minimal management effort.

To facilitate management and locality cloud systems are usually built on an infrastructure that consists of multiple geographically distributed data centers. Each data center may have one or more clusters containing physical compute, storage and networking resources. Each cluster may also be composed of hundreds or thousands of nodes constructed from commodity machines and connected by commodity interconnect. On top of the infrastructure there is a management software layer deployed in each cluster.

Figure 1 shows a model that illustrates a cloud computing environment. We will use the following model throughout this thesis. The *cloud customer* deploys and runs services over a cloud infrastructure. A cloud customer can be seen as 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 deployed across multiple clusters consuming resources and causing resource allocation demands for the cloud infrastructure. The *cloud infrastructure* is composed of geographically distributed clusters where each *cluster* is composed of many thousands of commodity servers connected

[1] Utility computing service is a service provisioning model in which a service provider makes computing resources, for example for computation, storage and high level services, available to the customer as needed, and charges them for metered usage similar to water and electricity.

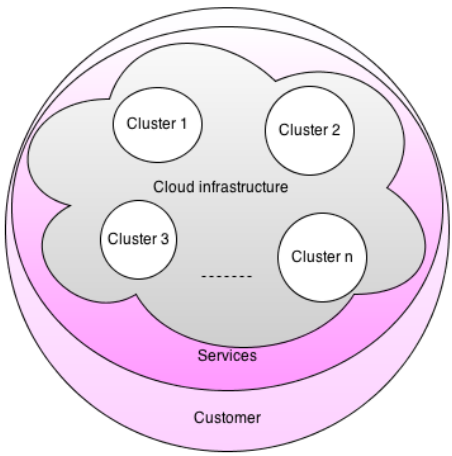

Figure 1: Elements in cloud environments.

using low-latency Local-Area Network (LAN) [22]. The infrastructure provider is responsible for managing physical and virtual resources and controlling resource usage of each individual service running across multiple clusters. The focus of this thesis is on how to manage resource allocations and resource usage for customers' services running across clusters.

Resource management is perhaps the most indispensable task of cloud computing [13, 20, 38]. Resource consumption of each service must be tracked to determine whether services are meeting their target service levels. Large-scale services often need more sophisticated and scalable resource usage monitoring and allocation support. It is important to collect resource consumption statistics across clusters to make further management decisions such as accounting and billing, scheduling, load balancing, and network analysis. For example, resource usage data need to be collected from several clusters in order to generate a single bill for a customer's service consumption across the entire infrastructure.

Resource allocations and *resource usage* management mechanisms for services running across clusters play vital roles in the performance of the entire system, economical sustainability of the provider, and level of customers satisfaction. However, when providing the utmost customer satisfaction the service provider ought to make sure not to over-commit resources beyond the agreed limit between the customer and the provider [30, 38]. Moreover, resources consumed by different services should be monitored and collected using an efficient mechanism with minimal interference to the performance of the services [11, 17]. Thus, resource usage collection and allocations mechanisms should impose economical constraints to both sides, the customer and the cloud provider.

Resource allocation management is about assigning shared resources (such as CPU, network, storage, and credit) to services in a manner that satisfies service demands across multiple clusters while maintaining the aggregated global

limit. Resource allocation mechanisms in cloud computing guarantees that the requirements of services are met. In other words, the management mechanisms should ensure that resource consumption does not exceed the upper bound while simultaneously ensuring that services running in some clusters are not starved while the total consumption is below the upper bound. To this end, allocation mechanisms should know the status of service allocation across clusters and, based on that, intelligently apply algorithms to better distribute shares to services according to their runtime requirements [32, 37, 40]. The goal is to avoid both under- and over-provisioning of resources.

On the other hand, *resource usage* management is concerned with how data about resources consumed by services across clusters is collected and aggregated to have a single consistent global view for different purposes such as accounting and billing [10, 11], creating traces [34] and input for resource allocation [9]. Resource consumption data should be monitored continuously and need to be collected and synchronized to maintain a single consistent view of the data generated across different clusters. However, the synchronization mechanism involves a trade-off between consistency of the data and frequency of data synchronization since it has impact on the performance of the system.

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

2.1 Resource Allocations Management

Service owners may want to cap (put limits on the) maximum amount of resource allocations (such as total credit, storage quota, total number of CPUs, total CPU hours, number of VMs, IP addresses, and network connections) to be consumed in the cloud and the cloud provider should have a mechanism to enforce and properly allocate the limits across the services running on multiple clusters [26, 21].

Managing resource allocation limits in these large distributed systems is difficult as it requires maintaining a consistent view of total usage when resource consumption occurs concurrently at several clusters. Resource allocation management is essential so that services cannot exceed their allocated or paid-for budget. It may also enforce global allocation limits in order to solve problems such as spurious services that flood and overburden the system with dummy tasks while denying access to other services, and malicious services that launch distributed denial of service (DDoS) attacks.

Resource consumption of services running on tens of thousands of nodes in different clusters must be properly monitored. Besides, resource allocations should be distributed across clusters optimally with the objective of minimizing the overhead associated with them while meeting service demands.

2.2 Resource Usage Management

Resource usage management involves monitoring, collecting, and aggregating information on consumption of resources, in order to facilitate decision making in cloud environments [7]. It requires a transparent and efficient management mechanisms to have consistent cloud wide view of the data. The degree to which resource usage must be managed, i.e., update frequency and granularity of monitored statistics information depends and varies 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. The resource usage has to be monitored, synchronized and aggregated to perform billing operations. Thus, for cloud services to be commercialized in support of the pay-as-you-go pattern, the cloud needs to support the ability for runtime usage across clusters to be accurately measured, collected and synchronized.

Therefore, an accounting and billing system that is capable of monitoring, collecting and processing usage data and metrics should be incorporated into the cloud architecture to enable the business model of cloud providers. Once services of customers are deployed to the cloud, it is critical to monitor its functions and track usage on an ongoing basis in oder to determine how much customers are being charged for use of the cloud resources.

Chapter 3

Resource Usage and Allocations Management Approaches

Despite the significant challenges due to the scale and complexity of large-scale distributed systems, the IT industry has made great progress as can be seen in cloud computing and its evolution toward commoditization [1] of the infrastructure. However, the abstraction of consolidated compute resources provided through the cloud model is only a partially sufficient response to dealing with this scale and complexity [16, 18]. The cloud model only solves the problem of having access to a large infrastructure. In order to fully realize the benefits an effective management model is also required.

Infrastructure providers find themselves in an era where customers expect that their services are delivered with certain requirements in performance, capacity, cost and geographical distribution. Recently, there has been much interest in software driven infrastructure management [15, 28]. 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 abstraction provided by an operating system to the different components of a computer. However, traditional management mechanisms, tools and techniques fall short of the emerging software driven infrastructure requirements due to their limitation to scale [17].

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

As discussed in Chapter 2, successful resource allocation and resource usage management of services at the cloud scale requires a rich set of global information and global control about each individual service running across

[1] Commoditization is the transformation of computing resources to utility services.

clusters. As the number of clusters and the size of services increase, the overhead in collecting, analyzing, and acting upon the associated data grows, and the need to develop an appropriate level of global knowledge about the state of services for decision making is required. The input data required to compute a timely response to changes in resource demands and settings increase unless appropriate management is in place.

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 size of the managed entities (e.g., of the clusters or services) and the amount of geographic dispersion.

3.1 Centralized Management

In this approach, a centralized manager keeps track of all available resources, collects usage from clusters and make decisions. In general the management functions and decision-making are concentrated at a central component. All clusters send messages requesting resources and reporting resource usage to the manager. The manager is responsible for allocation of resources to services and storing and processing resource usage data. With such centralized design, few decisions are required about the allocations and usage of resources.

The majority of cloud computing platforms rely on centralized architectures combined with a hierarchical system of governance [4, 13, 18]. 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 clusters and a lot of communication needs to take place from different clusters 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 ability to be scaled.

3.2 Decentralized Management

In recent years, there has been an increased and wide interest in decentralized management techniques and cooperative distributed decision making. 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

that, if entirely transmitted, can lead to network congestion. Besides, the information received may not be up-to-date due to delays.

As a result of decentralized management the decision-making process is dispersed among different management components so as to alleviate the problems encountered in a centralized approach. The decision-making is not confined to a central component, rather it is performed by multiple management components, with managers at different clusters making key decisions relating to their sphere of responsibility and cooperating with other peers when necessary.

In such an architecture all the management peers involved in a management task play similar roles, interacting cooperatively without any distinctive role. The aim of this scheme is that each cluster performs its responsibilities independently without being hampered by the state of other clusters and only interacts with others when it is not possible to meet the global management goal locally. Besides, decentralized management can abstract the differences between managed entities in different clusters without limiting their capabilities. It is thus, a viable method to achieve a stable growth pattern in an era of ever-growing technology with the possibility of each peer having different internal management policies and mechanisms.

The main characteristics of a decentralized approach are the following.

- *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 in case of failure of one or more clusters the remaining clusters can still provide 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 different management components located on different clusters reducing their waiting time.

- *Better Scalability*: In general, a decentralized management scales well as the size 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 has its own control that derives its management decision based on its policies. Local management units need only coordinate with other clusters when local information is not sufficient for making decisions.

There is a growing trend in using decentralized management approaches to support the management demands in different computing domains as the current centralized model fails to achieve the goal. For example, they are used in allocation management across nodes inside a LAN [29] as well as for network management [26, 31, 37]. The goal in this thesis is to apply this technique for inter-cluster management of resource allocations and resource usages of services that run on multiple clusters.

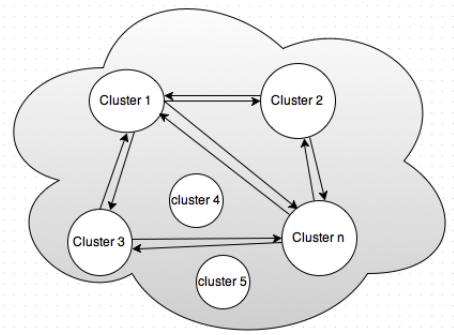

Figure 2: Decentralized management approach with a management entity in each cluster.

Figure 2 shows the architecture of decentralized management. Management components interact with all or part of other management components depending on situations. The arrows in the figure indicate interaction among management units. Some management units can work alone without interaction with other peers as long as their decisions can be satisfied locally.

3.2.1 Challenges

Despite its advantages, there are a lot of challenges in the decentralized management model which are discussed below.

- *Balancing the level of autonomy.* A decentralized approach assumes that each management entity is autonomous and has its own control that derives its management decision based on its policies. However, if several management units take independent decisions, the overall decisions made jointly by all management units might diverge from the global system goal. To avoid the divergence of individual management decisions, management units should communicate regularly with each other to align their decisions with the global system goal.For example, individual management units should try to make local decisions as much as possible using local information and cooperate with others when that is not possible. However, finding the balance between making local decisions and cooperative decisions is a challenge.

- *Complexity of decentralized management.* Complexity of decentralized management is higher due to the lack of a single point of control and synchronization. Consistent decentralized management is difficult to maintain when the system state constantly evolves as services are expected to consume resources continuously across clusters. The system should incur management overheads that scale slowly with the number of clusters.

- *How often to share information.* Determining how often to synchronize information among management units is challenging and depends on the management task. The goal is to address the issue of communication and synchronization overheads by minimizing the amount of information shared among peers and reducing the update frequency.

- *Decisions based on partial inforamtion.* Decision making requires knowledge of the global state of the system. In a dynamic system such knowledge, when available, may be out of date. Moreover, exchanging the whole system state among peers could put too much overhead on the system Thus, an efficient algorithm that can work on partial information from recent historical data is required.

- *Scalability.* Due to the growing size and complexity of large-scale systems, scalability of the management unit is an important issue. The system should scale gracefully with the number of clusters and services in the system.

- *Robustness.* The system should be robust to management unit and network failures of various types. As systems scale, failures become both inevitable and commonplace. The system should localize such failures so that it continues to operate and deliver useful service in the presence of failures.

- *Long delays.* There could be long delays as some clusters communicate via WAN and the decision process should take this into account.

- *Fast optimization techniques.* Determining optimal solutions in short periods of time requires fast optimization techniques which may be difficult to realize.

A hybrid management approach can also be used [6, 12, 14] by incorporating some features of both the centralized and the decentralized approaches.Typically hybrid approaches 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 decision and contact the coordinator when it is not possible. This approach suffers from similar issues as centralized approach due to the single coordinator.

Chapter 4

Summary of Contributions

The publications in this thesis focus on resource allocations and resource usage management in multi-cluster large-scale environments.

4.1 Challenges Addressed

This section presents some of the challenges discussed under Section 3.2.1 which are addressed in this thesis. Papers I, II and III present highly *scalable* soultions for resource allocations and usage managements since the solutions scale with the number of clusters and the number of as services running on them. The solutions are also *robust* since instance failures are localized and the system continues to operate and deliver useful service.

Papers I and II reduce *communication* overhead by triggering global operations only when the allocated balance in a cluster decreases below a threshold and allocations are distributed based on demand of each cluster in a manner that avoids further immediate global operation. Moreover, Paper II combines multiple services as a bundle, which further reduces communication. Allocations are distributed using exponential moving average (EMA), which is a simple but fast and efficient technique to predict the right allocation fairly by smoothing out short peaks in demand. Allocations are calculated based on each cluster's history of resource consumption to estimate how much allocation it can receive. Furthermore, each management unit is autonomous with respect to its sphere of control and can have its own local policies on how to spend allocations. To avoid over or under provisioning of allocations, management units communicate occasionally with each other to redistribute of allocations. I.e., individual management units try to use local allocations as long as they have enough allocations, and only communicate with each other when local allocation is below a threshold.

Paper III reduces *communication* by processing usage records in each cluster and applying asynchronous update propagation across clusters. Since updates

are propagated asynchronously, each instance can wait to collect a large amount of records and compress them in order to save network bandwidth and reduce update frequency. Moreover, when a request for accounting data arrives, the request can be served either locally without contacting others (provided that the requested data is in the local data store) or by reaching consensus from a majority of accounting instances. Besides, each accounting instance autonomously collect and process usage data locally while propagating updates to others occasionally. However, during accounting data request accounting instances should agree on data that should be include in the response.

4.2 Distributed Resource Allocation

4.2.1 Paper I

Paper I [24] focuses on how to manage resource allocation for services running in geographically distributed clusters. The paper presents a fully distributed resource allocation scheme for tracking resource allocations across distributed clusters. The solution monitors resource consumptions by services that are spread over a number of clusters with the goal that global polls are triggered 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. Range of simulations were performed and the communication costs of the fully distributed allocation approach is compared against a base centralized resource manager algorithm, showing that the distributed approach is more efficient and effective.

4.2.2 Paper II

Paper II [39] extends the work on Paper I by adding support for prepaid credit allocations for composite services running in multiple clusters. The goal is to combine customers' services together forming a bundle of services and to allocate credit for the bundle in order to reduce the overhead on the accounting system that could be incurred if performed individually for each service.

A distributed accounting component is deployed to manage credit allocations at each cluster. Each of the accounting components supervises the bundles in that particular cluster, and is also responsible for credit redistribution when a customer's credit in one cluster is not sufficient to compensate for further resource usage. Credit redistribution is performed based on bundle consumption rates in a cluster.

The approach increases the overall throughput of the accounting system. Besides, it minimizes synchronization overheads so that the response time to the client is reduced.

The theoretical analyses performed indicate that the work can provide an inexpensive accounting solution for long-lived services.

4.3 Distributed Resource Usage Management

4.3.1 Paper III

Paper III [23] 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, collection and synchronization of 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 request 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.

Chapter 5

Future Work

5.1 Extensions to Current Work

The following are the possible extensions on resource allocations.

- *Proof for conflict resolution.* Conflicts occur when more than one cluster contends to run the algorithm simultaneously for the same service. Each management units deployed in every cluster has a unique ID as identifier and we have handled the conflict using cluster with a higher-ID-Wins protocol. The protocol has not been proved for its correctness. Formal proof of correctness of conflict resolution during coordinator selection is a natural extension to work on.

- *Resource allocations management inside clusters.* The focus of our current work has been to handle resource allocations across clusters, but we have not considered how resource allocations are handled inside clusters. We will investigate cluster-wide allocation management mechanism in order to control resource consumption of services that are running on tens of thousands of nodes.

- *Consideration of availability of resource in the infrastructure during allocation decisions.* Currently, quotas are distributed based on demands from services without considering the availability of sufficient resources in the infrastructure. We want to consider infrastructure resource availability as another parameter during the decision process. This is because services may be relocated to other clusters when there are not enough allocations available locally which may in turn lead to further decision making.

- The experiment reported in Paper II [39] was to validate the theoretical analysis with uniform bundle size across clusters. We have realized that a rigorous experiment needs to be performed. We plan to perform an experiment that takes into account communication cost as well as

performance of the system for the different schemes presented in the paper with variable bundle size across clusters. Moreover, we will investigate the minimum threshold that should be kept at each cluster before applying credit redistribution so that the services are not disturbed in the process.

5.2 Other Areas of Future Work

We think that the work presented in this thesis can be the basis for addressing the following important problems in management of large-scale infrastructures.

5.2.1 Distributed Guaranteed Differentiated Quality of Service

Service providers and their customers often negotiate utility based Service Level Agreements (SLAs) to determine costs and penalties based on the achieved performance level. Ideally, resource demands for customers should not be specified per cluster but instead a limit should be set at a cloud level allowing customers to consume resources dynamically across all clusters subject to the specified agreed overall average cloud SLA level. Mechanisms to ensure that the overall average SLA level does not go above or below a certain threshold during service provision are required but expensive to achieve [21, 24, 29]. Current works either follow a centralized approach [3, 8] or do not provide guaranteed services [2, 42, 43].

In our future work we want to investigate distributed architectures for providing guaranteed differentiated quality of service for clouds composed of multiple clusters distributed across the globe. We aim to maximize the revenue while balancing the cost (penalty) of using the resources for different SLA class (e.g., gold, silver, and bronze customers). Each cluster provides differentiated quality of service for customers in a decentralized fashion while cooperating with other clusters to maintain the overall average SLA levels of each customer and maximizing revenues for the cloud service providers.

The idea is that overall average SLA levels are maintained through cooperative interaction and partial information sharing among different clusters. SLA violations at some clusters could be allowed as long as overall average SLAs are maintained while the service provider is able to make profit. Furthermore, no single decision-maker with a global view of the system is required to guide the decision process. All clusters should make local predictions using local information combined with partial information exchanged from other clusters.

5.2.2 Service Centric Load Balancing

Load balancing algorithms adopting a service centric objective function aim to optimize the performance of each individual service. When the services are geographically dispersed over multiple clusters, these services are connected

through relatively low-speed networks. Hence network delay for data transfer may become a significant factor in the load balancing strategy. It is important to have a decentralized load-balancing scheme that takes network delay into consideration, performs load balancing locally at each clusters, and cooperates regularly with other clusters to assure that the global goal is satisfied.

We plan to investigate how our approach on allocation redistribution could be adapted for load redistribution for services running across multiple clusters.

Bibliography

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

[2] M. Amirijoo, J. Hansson, S. Son, and S. Gunnarsson. Robust quality management for differentiated imprecise data services. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 265–275, 2004.

[3] M. Andreolini, E. Casalicchio, M. Colajanni, and M. Mambelli. A cluster-based web system providing differentiated and guaranteed services. *Cluster Computing*, 7(1):7–19, Jan. 2004.

[4] 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.

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

[6] 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.

[7] 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.

[8] K. Boloor, R. Chirkova, T. Salo, and Y. Viniotis. Analysis of response time percentile service level agreements in soa-based applications. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, 2011.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] E. Lakew, F. Hernandez-Rodriguez, L. Xu, and E. Elmroth. A synchronization mechanism for cloud accounting systems. To be submitted.

[24] 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.

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

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

[27] 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/.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Y. Zhang, S. Q. Ren, S. B. Chen, B. Tan, E. S. Lim, and K. L. Yong. DifferCloudStor: Differentiated quality of service for cloud storage. In *The Asia-Pacific Magnetic Recording Conference (APMRC), 2012 Digest*, pages 1–9, 2012.

[43] J. Zhao, H. Hassanein, J. Wu, and G. Gu. End-to-end QoS routing framework for differentiated services networks. *Comput. Commun.*, 26(6):566–578, Apr. 2003.