Christian Baun
Marcel Kunze
Jens Nimis
Stefan Tai

Cloud Computing

Web-Based Dynamic IT Services

Springer

Cloud Computing

Christian Baun • Marcel Kunze • Jens Nimis
Stefan Tai

Cloud Computing

Web-Based Dynamic IT Services

Springer

Christian Baun
Dr. Marcel Kunze
Karlsruhe Institute of Technology (KIT)
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
Germany
Christian.Baun@kit.edu
Marcel.Kunze@kit.edu

Jens Nimis
Hochschule Karlsruhe -
Technik und Wirtschaft
Fakultät für Wirtschaftswissenschaften
Moltkestraße 30
76133 Karlsruhe
Germany
jens.nimis@hs-karlsruhe.de

Stefan Tai
Karlsruhe Institute of Technology (KIT)
& FZI Forschungszentrum Informatik
Englerstraße 11
76131 Karlsruhe
Germany
stefan.tai@kit.edu

This is an English language translation of the German language edition:
Cloud Computing (2nd edn.) by C. Baun, M. Kunze, J. Nimis, and S. Tai
Published in the book series: Informatik im Fokus
Copyright © Springer-Verlag Berlin Heidelberg 2011
Springer-Verlag is part of Springer Science+Business Media.
All Rights Reserved.

ACM Codes : C.2, D.2, D.4, H.3

ISBN 978-3-642-20916-1 e-ISBN 978-3-642-20917-8
DOI 10.1007/978-3-642-20917-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011931673

Cover design: KünkelLopka, Heidelberg

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Cloud computing is on everyone's lips: as an innovative technology, as the next generation of the Internet, as a fundamental transformation of the entire IT landscape, and as an auspicious opportunity to introduce new business ideas. But what is actually at the heart of this notion?

As a result of the multi-faceted viewpoints and the interests expressed by the various stakeholders, cloud computing is perceived as a rather fuzzy concept. It is our ambition to try and clear some of the haze surrounding the cloud computing concept and thus pave the way both for its successful application and for advanced research work and innovation.

In only a few months, we were able to compile the available material on cloud computing – which has become quite comprehensive in the meantime – and to condense it to fit into a smart book. But this could only succeed with the help of many colleagues from research organizations, but especially from the industry. They contributed a never-ending stream of new input and helped us in numerous discussions to develop a more differentiated picture of this exciting topic.

As is common with most disruptive technologies, the opinions on cloud computing diverge strongly in terms of acceptance and risk assessment. There are companies that cannot relate yet to the concept of using and deploying dynamic IT services, while others are at the cutting edge of this technology, some of them writing success stories that make headlines. This book will help our readers to study cloud computing rather unemotionally in the first place before they can enthusiastically take up its challenges and opportunities.

Due the fact that the discussion about cloud computing always has to consider both its technological and economic aspects, this book will be of major interest to a broad audience. It is equally intended for the students of our courses at the Karlsruhe Institute of Technology (KIT) and at other universities, for interested software engineers, and for future-oriented decision-makers. We trust that the technical information compiled here is rich and coherent. But this book should also be an interesting read for top executives and managers.

We would like to take the opportunity to thank everybody who contributed to the making this book. We received continuing and substantial support by the KIT focus COMMputation that helped to realize this book. Our special thanks go to Anja Langner and Bianca Pagliosa for their invaluable help in creating the illustrations, and to our families and friends for their understanding and support, and their patience when, once again, we sacrificed our leisure time for this book.

Karlsruhe
March 2011

Christian Baun
Marcel Kunze
Jens Nimis
Stefan Tai

Contents

1 Introduction 1
1.1 What Is This Book About? 1
1.2 Definition 3
1.3 Outline of This Book 4

2 Cloud Basics 5
2.1 Virtualization 5
2.1.1 Benefits and Drawbacks of Virtualization 5
2.1.2 Virtualization Concepts 7
2.2 Service-Oriented Architectures 10
2.2.1 The Properties of SOA 10
2.2.2 How Is an SOA Implemented? 12
2.3 Web Services 13
2.3.1 Interoperability 13
2.3.2 SOAP Versus REST 14

3 Cloud Architecture 15
3.1 Public, Private, and Hybrid Clouds 15
3.2 The Technical Landscape of Cloud Services 16
3.3 Infrastructure as a Service 18
3.4 Platform as a Service 18
3.5 Software as a Service 20
3.6 Humans as a Service 21
3.7 Other Categories of Cloud Services 22

4 Selected Cloud Offerings 23
4.1 Amazon Web Services 23
4.1.1 Amazon Elastic Compute Cloud (EC2) 25
4.1.2 Amazon Simple Storage Service (S3) 28
4.1.3 Amazon Elastic Block Store (EBS) 29

4.1.4 Amazon Simple Queue Service (SQS) 29
4.1.5 Amazon SimpleDB 30
4.1.6 Amazon Relational Database Service 30
4.1.7 The Amazon Web Services as ‘Team Players’ 31
4.2 Google Cloud Services 33
4.2.1 Google App Engine 33
4.2.2 Google Storage 34
4.2.3 Google Cloud Print 35
4.3 Windows Azure 36
4.4 Salesforce.com 37
4.5 Cloud Gaming 38
4.6 Cloud Operating Systems 38

5 Cloud Management 39
5.1 Service Level Agreements (SLAs) 39
5.2 Lifecycle and Automation 40
5.3 Management Services and Tools 41
5.3.1 Monitoring 41
5.3.2 Control 41
5.3.3 Development 45
5.4 Security Management 46
5.5 Risk Management 47
5.6 Legal Compliance 48

6 Open Source Cloud Stack 49
6.1 Physical and Virtual Resources 49
6.2 Eucalyptus 51
6.2.1 Architecture and Components 51
6.3 OpenNebula 54
6.4 Nimbus 54
6.5 CloudStack 55
6.6 OpenStack 55
6.7 AppScale 56
6.8 TyphoonAE 56
6.9 Apache Hadoop 56
6.9.1 MapReduce 57
6.9.2 Hadoop Distributed File System 57
6.9.3 Pig 59
6.9.4 Hive 59
6.9.5 Hadoop as a Service 59
6.10 The OpenCirrus Project 60

7 Economic Considerations 63
7.1 Fields of Application 63
7.2 Evaluation Models 64
7.2.1 Cost Models 65
7.2.2 TCO Framework 66
7.3 Business Models 66

8 Opportunities and Risks 69
8.1 Market Development 69
8.2 Situational Evaluation 70
8.3 Conclusion 71

9 Appendix 73
9.1 Performing EC2 Tasks with the Amazon Tools 73
9.2 Performing EBS Tasks with the Amazon Tools 75
9.3 Performing RDS Tasks with the Amazon Tools 76
9.4 Performing S3 Tasks with s3cmd 77
9.5 Using Google App Engine 77
9.6 Using AppScale 79
9.7 Installing and Using Eucalyptus 79
9.8 Data Mining with Amazon Elastic MapReduce 82

Glossary 85

Bibliography 89

Index 93

Chapter 1
Introduction

With this book, we want to give our readers an overview of cloud computing architecture, services, and applications, but it is not intended to be exhaustive. It is our aim to bring all readers up to date on this technology and thus to provide a common basis for discussion. This book does not require any knowledge of the technological background.

1.1 What Is This Book About?

Cloud Computing is a buzz-word in today's information technology (IT) that nobody can escape. But what is really behind it? There are many interpretations of this term, but no standardized or even uniform definition. Cloud computing fosters the provision and use of IT infrastructure, platforms, and applications of any kind in the form of services that are electronically available on the Web. The term *cloud* hints at the fact that these services are provisioned by a provider on the Internet (or on the intranet of a larger organization). Users of cloud services, on the other hand, can propose their own offerings as services on the Internet or on an intranet.

Usually, cloud resources are virtualized: This way, cloud users always have the desired and required view on their infrastructure, and their applications are not subject to any systemic dependencies or constraints.

Moreover, cloud services can be scaled dynamically: If an application requires additional resources, these can be added immediately and without much effort by an automatic process. Thus, Web application developers with innovative ideas do not have to invest heavily in new hardware when founding a company. They can obtain the required resources flexibly from a provider while focusing on their business idea. With growing demand, the infrastructure can be adjusted automatically to the extended requirements.

Cloud computing adopts the ideas of utility computing. The required number of resources are made available and must be paid for. For unused resources,

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_1,

nothing will be charged. Cloud computing is therefore of economic significance since, due to its flexibility in provisioning and using its services, considerable savings are possible. The capacities which are available for use upon request are enormous, creating *economies of scale* with a very favorable price-performance ratio.

In the meantime, several commercial providers, such as Amazon, Google, or Microsoft have emerged. Their offerings, however, are of different types: Amazon offers virtualized resources for generic use while the clouds provided by Google and Microsoft allow application hosting. With every provider striving for a competitive edge, all current offerings are indeed proprietary and there are no standards so that, in general, a quick provider change is not an easy task.

Cloud computing critics like to allege this danger of vendor lock-in besides possible security concerns. Mainly established IT managers and department heads preach caution. But a closer look reveals that these arguments very often seem to aim at defending their ‘inherited’ vested rights in classical data centers. For this reason, mostly young companies (startups) that do not suffer from these dependencies take advantage of this new technology. But there are also established companies which have approached the subject of cloud computing and can already boast a sustainable rise in efficiency thanks to its use. Besides taking advantage of public clouds, companies are increasingly relying on their own private clouds.

From the customer’s perspective, there are a number of options that foster more productive and flexible work: By taking advantage of the offerings available in the market, purchasers of IT services are able to gain independence from the classical, static data center operations. In this context, cloud computing develops a *force of creative destruction* and old structures are replaced by new ones [25, 8].

The Corporate Executive Board (CEB) in Washington presented a study based on interviews held with 200 business and IT executives [62]. For the next years, the study predicts that the classical IT departments will lose a great deal of their responsibilities to business management with staff consequently shrinking to a quarter of their current numbers. The central drivers of these changes are cloud computing, social media, and an increase in the number of so-called knowledge workers – all this combined with a company IT which has used up most of its potential for efficiency enhancement. With increasingly specialized cloud services prevailing, business management is gaining the ability to purchase IT solutions independently from and even by bypassing the internal IT department.

Similarly, knowledge workers can use their smart end devices and their social networks to obtain information on IT systems and the associated services by themselves.

This also changes the role of the CIO who will either become the head of a slim, internal shared-services department or act as a purchasing agent and manager of external services in the future. In this scenario, the classical data center is bound to change into an IT service center.

1.2 Definition

Although there is no standardized, uniform definition of cloud computing, its basic concepts as well as its general objectives are undisputed: Cloud computing uses virtualization and the modern Web to dynamically provide resources of various kinds as services which are provisioned electronically. These services should be available in a reliable and scalable way so that multiple consumers can use them either explicitly upon request or simply as and when required.

From the cloud provider's perspective, this usually implies a multi-tenant architecture and a usage-based billing model. Thus, we can define the term *cloud computing* as follows:

> By using virtualized computing and storage resources and modern Web technologies, cloud computing provides scalable, network-centric, abstracted IT infrastructures, platforms, and applications as on-demand services. These services are billed on a usage basis.

This definition indeed does not specify whether the services are provided by a distributed system or a single, high-performance server, such as a mainframe. This is in contrast to *grid computing* which always uses a distributed system. Usually, cloud services also rely on a distributed infrastructure, but their management is typically determined in a central (and proprietary) manner by a single provider. This is another difference between cloud computing and grid computing where distributed nodes are usually autonomous. For an extensive discussion on this topic, see [30].

Another decisive factor for cloud computing is its economic impact. Due to its strict service-orientation and the utilization of Web standards and the Internet as an integrated technology and business platform, cloud computing is well-positioned for various kinds of applications. This includes in particular Web applications and modular services in distributed business networks and process chains.

An often-quoted definition of cloud computing was established by the National Institute of Standards and Technology (NIST) in the U.S. [105]. It specifies five essential characteristics of cloud computing, three different service models, and four different deployment models. The essential characteristics are:

- **On-demand self-service**: Services can be provided unilaterally and on demand to consumers without requiring human interaction.
- **Broad network access**: Services are available over the network in real-time through standard mechanisms.
- **Resource pooling**: The resources are pooled to enable parallel service provision to multiple users (multi-tenant model), while being adjusted to the actual demand of each user.
- **Elasticity**: Resources are rapidly provisioned in various, fine-grained quantities so that the systems can be scaled as required. To the consumer, the resources appear to be unlimited.

- **Measured quality of service**: The services leverage a quantitative and qualitative metering capability so that usage-based billing and validation of the service quality are possible.

  The service models and deployment models will be discussed in Chap. 3.

1.3 Outline of This Book

This book first introduces the foundation of cloud computing with its basic technologies, such as virtualization and Web services. Then, the cloud architecture and its service modules will be discussed. The following chapters cover selected cloud offerings and management tools; in particular, security and data privacy issues will be dealt with. Then, we will present current open source developments, such as Hadoop and Eucalyptus. The OpenCirrus^{TM} project of HP, Intel, and Yahoo!, in which the authors of this book take part as development partners, picks up on these topics. This project examines the foundation of cloud systems and cloud applications. Next, economic considerations, for instance cost and business models, will be discussed. Finally, an evaluation of the cloud market situation is given. The appendix contains some practical examples of how to use cloud resources or cloud applications and the glossary provides concise definitions of central terms.

We trust that our dear readers will enjoy this book from the first to the last page. Nevertheless, the individual chapters are written in such a way that each of them can be read and understood independently from the others. Readers who are less interested in the technical details might want to skip Chap. 2 and delve into the heart of the matter. If your time is tight and you would like to follow a path leading to technical architecture topics, Chaps. 1, 3, 5, 7, and 8 will be of interest to you. There is also a path for readers who rather want to focus on economic topics (Chaps. 1, 4, 6–8). These paths are shown schematically in Fig. 1.1.

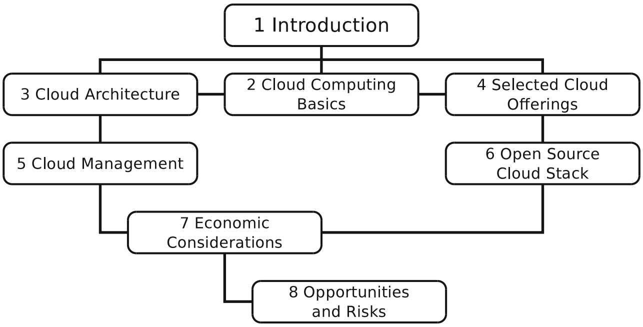

1 Introduction
3 Cloud Architecture
2 Cloud Computing Basics
4 Selected Cloud Offerings
5 Cloud Management
6 Open Source Cloud Stack
7 Economic Considerations
8 Opportunities and Risks

Fig. 1.1 How to find your way through this book

Chapter 2
Cloud Basics

One of the appealing aspects of cloud computing is that it hides the complexity of IT technology from users and developers. No need to know details of how a service is generated – it is the service provider's job to provide a corresponding abstraction layer. This chapter contains an overview of some technologies on which cloud computing depends: virtualization, service-oriented architectures (SOA), and Web services.

2.1 Virtualization

Resource virtualization is at the heart of most cloud architectures. The concept of virtualization allows an abstract, logical view on the physical resources and includes servers, data stores, networks, and software. The basic idea is to pool physical resources and manage them as a whole. Individual requests can then be served as required from these resource pools. For example, it is possible to dynamically generate a certain platform for a specific application at the very moment when it is needed. Instead of a real machine, a *virtual machine* is used.

2.1.1 Benefits and Drawbacks of Virtualization

For a provider of IT services, the use of virtualization techniques has a number of advantages [4]:

- **Resource usage:** Physical servers rarely work to capacity because their operators usually allow for sufficient computing resources to cover peak usage. If virtual machines are used, any load requirement can be satisfied from the resource pool. In case the demand increases, it is possible to delay or even avoid the purchase of new capacities.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_2,

- **Management:** It is possible to automate resource pool management. Virtual machines can be created and configured automatically as required.
- **Consolidation:** Different application classes can be consolidated to run on a smaller number of physical components. Besides server or storage consolidation, it is also possible to include entire system landscapes, data and databases, networks, and desktops. Consolidation leads to increased efficiency and thus to cost reduction.
- **Energy consumption:** Supplying large data centers with electric power has become increasingly difficult, and, seen over its lifetime, the cost of energy required to operate a server is higher than its purchase price. Consolidation reduces the number of physical components. This, in turn, reduces the expenses for energy supply.
- **Less space required:** Each and every square yard of data center space is scarce and expensive. With consolidation, the same performance can be obtained on a smaller footprint and the costly expansion of an existing data center might possibly be avoided.
- **Emergency planning:** It is possible to move virtual machines from one resource pool to another. This ensures better availability of the services and makes it easier to comply with service level agreements. Hardware maintenance windows are inherently no longer required.

Since the providers of cloud services tend to build very large resource centers (*IT factories*), virtualization not only leads to a size advantage, but also to a more favorable cost situation. This results in the following benefits for the customer:

- **Dynamic behavior:** Any request can be satisfied just in time and without any delays. In case of bottlenecks, a virtual machine can draw on additional resources (such as storage space, I/O capabilities).
- **Availability:** Services are highly available and can be used day and night without stop. In the event of technology upgrades, it is possible to hot-migrate applications because virtual machines can easily be moved to an up-to-date system.
- **Access:** The virtualization layer isolates each virtual machine from the others and from the physical infrastructure. This way, virtual systems feature multitenant capabilities and, using a roles concept, it is possible to safely delegate management functionality to the customer. Customers can purchase IT capabilities from a self-service portal *(customer emancipation).*

A drawback of virtualization is the fact that the operation of the abstraction layer itself requires resources. Modern virtualization techniques, however, are so sophisticated that this overhead is not too significant: Due to the particularly effective interaction of current multicore systems with virtualization technology, this performance loss plays only a minor role in today's systems. In view of possible savings and the quality benefits perceived by the customers, the use of virtualization pays off in nearly all cases.

Another problem is that after the consolidation, more systems need to be operated and managed: Besides the virtual machines, there is the physical infrastructure. By managing the resources using sophisticated management tools, the total balance is nevertheless positive because much less staff is required in practice.

2.1.2 Virtualization Concepts

Virtualization stands for a variety of different concepts and technologies, which differ by their implementation, relevance to practical use, and frequency of use.

2.1.2.1 Operating System Virtualization

The use of operating system virtualization or partitioning (such as IBM LPARs) in cloud environments may help to solve security and confidentiality problems, which would otherwise impair the acceptance of the cloud approach.

For this type of virtualization, which is also called ‘container’ or ‘jails’, the host operating system plays a major role. This is a concept where multiple identical system environments or runtime environments, which are completely isolated from each other, run under one operating system kernel (see Fig. 2.1). Seen from the outside, virtual environments appear as autonomous systems. All running applications use the same kernel, but they can only see the processes belonging to the same virtual environment.

Mainly Internet service providers (ISPs), who offer (virtual) root servers, prefer this kind of virtualization because it is associated with a minor performance loss and a high degree of security. The drawback of operating system virtualization is its reduced flexibility: While multiple independent instances of the same operating system can be used simultaneously, it is not possible to run different operating systems at the same time. Popular examples of operating system virtualization are the container technology from Sun Solaris, OpenVZ for Linux, Linux-VServer, FreeBSD Jails, and Virtuozzo.

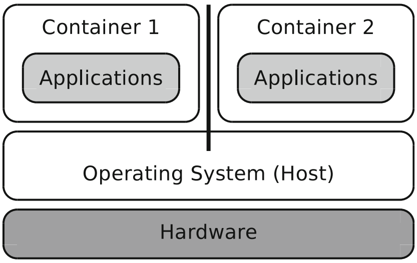


Fig. 2.1 The concept of operating system virtualization (container)

2.1.2.2 Platform Virtualization

Platform virtualization allows to run any desired operating systems and applications in virtual environments. There are two different models: Full virtualization and paravirtualization. Both solutions are implemented on the basis of a virtual machine monitor or hypervisor.

The hypervisor is a minimalistic meta-operating system used for distributing the hardware resources among the guest systems and for access coordination. A type-1 hypervisor is built directly on top of the hardware, a type-2 hypervisor runs under a traditional basic operating system (see Fig. 2.2).

Full virtualization is based on the simulation of an entire virtual computer with virtual resources, such as CPU, RAM, drives, network adapters, etc. including its own BIOS. Since the access to the most important resources, such as the processor and the RAM, is passed through, the processing speed of the guest operating systems nearly equals the speed to be expected if there was no virtualization. Other components, e.g. drives or network adapters, are emulated. While this decreases the performance, it allows to run unmodified guest operating systems.

Paravirtualization does not provide an emulated hardware layer to the guest operating systems, but only an application interface. For this purpose, the guest operating systems need to be modified because any direct access to hardware must be replaced by the corresponding hypervisor interface call. This is also referred to as hyper calls (just like system calls), which are used by the applications to call functions in the operating system kernel. Since this approach allows the guest system to participate actively in the virtualization (at least to some extent), a higher throughput than with full virtualization can be obtained, especially for I/O-intensive applications. Examples of full virtualization are the VMware products [135] or, specifically for Linux, the *Kernel-based Virtual Machine* (KVM). Under Linux, mostly Xen-based solutions are used for paravirtualization [14]. They play an important role, particularly in the realization of the Amazon Web Services [50].

2.1.2.3 Storage Virtualization

Cloud systems should also offer dynamically scalable storage space as a service. In this context, storage virtualization boasts a number of advantages. The fundamental

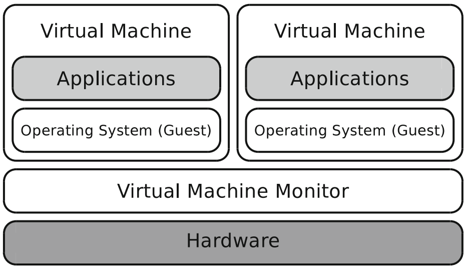


Fig. 2.2 Type-1 hypervisor (virtual machine monitor)

idea of storage virtualization is to separate the data store from the classical file servers and to pool the physical storage systems. Applications use these pools to dynamically meet their storage requirements. For the data transfers, a special storage area network (SAN) or a local company network (LAN) is used. Data for cloud offerings is mostly available in the form of Web objects that can be retrieved or manipulated over the Internet. An additional abstract administration layer is interposed between the clients and the storage landscape so that the representation of a datum is decoupled from its physical storage. This has a variety of advantages with respect to data management and access scalability. Central management also allows to operate the distributed storage systems at a lower cost.

Moreover, different categories of data storage can be organized in storage hierarchies (tier concept). This makes it possible to implement an automated lifecycle management for data sets, from tier-0 with the most stringent availability and bandwidth requirements to lower and cheaper tier levels with a correspondingly lower quality of service. The data can be migrated between these levels without affecting the service. By using snapshots, even large data quantities can be backed up without a special backup window. A further advantage of storage virtualization is that distributed mirrors may be created and managed in order to avoid service disruptions in case of malfunctions. Amazon, for instance, creates up to three copies in different data centers when storing data.

2.1.2.4 Network Virtualization

Techniques such as load balancing are essential in cloud environments because it must be possible to dynamically scale the services offered. The resources are usually implemented as Web objects. For this reason, it is recommended to apply the procedures commonly used for Web servers: Services can be accessed via virtual IP addresses. Through cluster technology, they realize load balancing as well as automatic failover in case of a failure. By forwarding DNS requests, it is also possible to integrate cloud resources into the customer's Internet namespace.

Network virtualization is also used for virtual local networks (VLANs) and virtual switches. In this case, cloud resources appear directly in the customer's network. Internal resources can thus be replaced transparently by external resources. VLAN technology has the following advantages:

- **Transparency:** Distributed devices can be pooled together in a single logical network. VLANs are very helpful when designing the IT infrastructure for geographically disparate locations.
- **Security:** Certain systems which require particular protection can be hidden in a separate virtual network.

On the other hand, VLANs involve more overhead for network administration and for programming active network components (switches, etc.).

2.1.2.5 Application Virtualization

Application virtualization is a software sales model where centrally managed applications are offered to the customers over a network. The advantages of application virtualization compared to traditional software installations are:

- Easier administration
- Automatic management of updates and patches
- Compatibility: all users work with the same software portfolio
- Global availability

There are two different methods for deploying virtual applications:

- **Hosted application:** The application is available on the Internet and is transmitted to the client, e.g. using a streaming protocol.
- **Virtual appliance:** The application can be downloaded and used on the customer's own computer.

In this case, a virtual environment provides all application files and components required by the program for it's execution. The virtual environment acts as a buffer between the application and the operating system, preventing conflicts with other applications or operating system components. In cloud environments, application virtualization is an important foundation of the SaaS concept (Software as a Service, see below) which is used for the dynamic provision of software components.

2.2 Service-Oriented Architectures

Besides virtualization, service-oriented architectures and Web services are to be considered as the fundamental prerequisites for cloud computing. Service-oriented architectures (SOA) are architectures whose components are implemented as independent services. They can be flexibly tied together and orchestrated and they can communicate via messages in a loosely coupled configuration. With cloud computing, virtualized IT infrastructures, platforms, and entire applications are implemented as services and made available for consumption in service-oriented architectures.

In the case of *public clouds* (see Chap. 3), services are offered on the Internet based on standardized Web protocols and interfaces. In this context, *Web services* and *RESTful services* have proven to be useful. For an SOA, however, the Web services technology is not mandatory.

2.2.1 The Properties of SOA

SOA is a style of software architecture, which defines how services are offered and used. These services may not only be used by customers, but can also be consumed by other services and applications. Often, services are orchestrated here as business processes, which means that a customer specifies a certain order for performing calls and data exchanges with various services. The process itself can be provisioned as a service. Thus, the promise of SOA is to promote IT architectures to the business process abstraction layer, or, conversely, to enable a uniform approach for the description and realization of business processes using IT services.

Typical properties of an SOA are:

- It consists of distributed components, i.e. the services.
- Heterogeneous service consumers and service providers are interoperable across platforms; different programming languages and platforms can be used to implement individual services.
- Services are loosely coupled and will be bound dynamically at runtime. An SOA consequently allows dynamic adjustments, which have a local (but no system-wide) effect.

A short and concise definition of service-oriented architectures can be found in [13]:

> An SOA is a system architecture that represents varied, different, and possibly incompatible methods or applications as re-usable and openly accessible services and thus allows to use and re-use them in a platform- or language-independent manner.

Figure 2.3 illustrates the basic, theoretical interaction between service provider, service consumer, and service directory for the dynamic binding of services at runtime. A service consumer can locate a suitable service in a service directory or learn of its existence by using a broker. If a suitable service has been found or brokered, the service consumer receives a reference (address, endpoint) for accessing the service, i.e. exchanging messages with it. Then, the service can be called, i.e. a message can be sent. The service provider replies by sending a message back.

In practice, service endpoints are often directly communicated as part of the message. Attempts to set standards for service directories, such as Universal

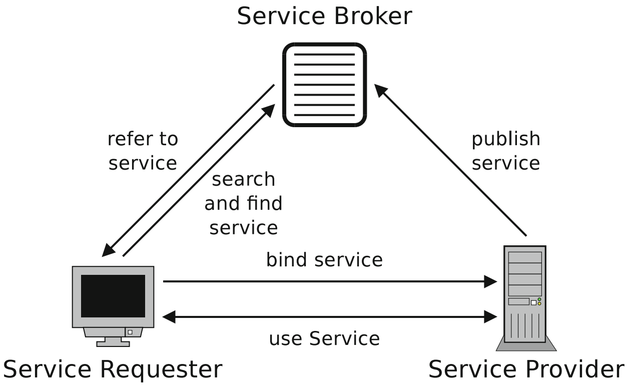


Fig. 2.3 SOA participants and actions

Description, Discovery and Integration (UDDI), were not successful. By sending endpoints directly in a message, it is possible to broker and bind services at runtime in a very dynamic, target-oriented way.

2.2.2 How Is an SOA Implemented?

An SOA may be implemented in many ways. The actual implementation depends primarily on the decisions taken with respect to communication and integration (coupling) between service provider and service consumer. Common approaches are above all Web services based on the Web Service Description Language (WSDL) and SOAP as well as RESTful services.

Services can be integrated into an SOA either by using point-to-point-connections or the hub-and-spoke approach. Each point-to-point-connection handles the connection between a service provider and a service consumer on an individual basis. For this purpose, the service consumer needs to know the endpoint of the desired service (IP address, URL). The service request is then addressed directly to the appropriate service provider.

With the hub-and-spoke approach, a broker acts as an intermediary between service providers and service consumers. The service consumer does not know the exact service provider address. For each service, a symbolic name exists, such as a URI. In the SOA context, the broker is called Enterprise Service Bus (ESB). Its tasks include routing, i.e. controlled, reliable forwarding of messages between the services across different systems and independently of the protocols in use. Another task is the transformation of data from one format to another. In its simplest case, this could refer to upgrading or downgrading different data types between 32-bit and 64-bit platforms, but also to the conversion between differing XML standards. Further tasks are the administration of a service directory and, depending on the implementation, the orchestration of messages.

Figure 2.4 contrasts point-to-point connections with the hub-and-spoke approach using an ESB.

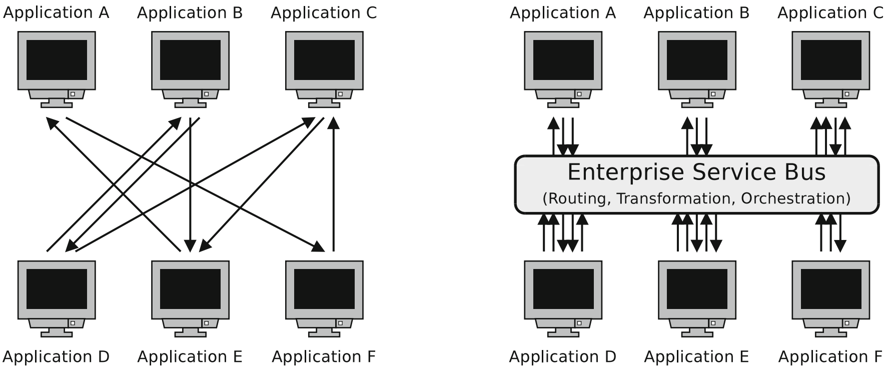


Fig. 2.4 Point-to-point-connections vs. Enterprise Service Bus

2.3 Web Services

Unlike distributed systems, which are interconnected over local networks, distributed cloud computing systems integrate heterogeneous resources which, in theory, could be located at any place on this earth where an Internet connection is available. The disadvantage of Internet connections compared to local networks is that they automatically entail problems, such as slow response times, low data rates, and potentially unreliable connections. In such environments, it is recommended to use a loosely coupled, asynchronous, message-based communication via Web services.

Just like with service-oriented architectures, a large number of different definitions for Web services exist. The Web Services Architecture Working Group of the W3C defines Web services as follows [137]:

> A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.

Wohlstadter and Tai [32] define Web services as distributed middleware, which enables machine-to-machine communication on the basis of Web protocols. Web services are defined not so much by the technology they use, but rather by their intended use. They propagate a compositional approach for application development. Functions can be integrated into an application using external, distributed services; and it is also possible to address legacy systems via Web services.

2.3.1 *Interoperability*

Web services describe the standards required to format and process messages as well as the standards for service interfaces. Two popular approaches are SOAP/WSDL-based Web services and RESTful (REpresentational State Transfer) services. SOAP is a messaging protocol and WSDL (Web Services Description Language) is an interface description language. Thus, SOAP/WSDL-based services have programmatic interfaces. REST, on the other hand, describes a style of software architecture, which is built on top of HTTP. RESTful services can only be invoked from the uniform HTTP interface. Both approaches use uniform resource identifiers (URIs) to identify the required services.

In their basic form, Web services describe only the primitives required to exchange documents (data) between service consumers and service providers. There are standards for transactional, reliable, and secure services for SOAP/WSDL. The Web Services Platform Architecture (WS-*) describes a set of extensions, which can be composed in a modular way; each of these extensions addresses a desired quality-of-service property.

The most common data exchange format is XML. XML is mandatory for SOAP/WSDL and commonly used with RESTful services. The conversion of XML data structures to programming language-specific data structures is done using standardized mapping mechanisms. An alternative data exchange format is JSON (JavaScript Object Notation), which has become quite popular, in particular when RESTful services are consumed directly in the Web browser by JavaScript.

The Web service interface for SOAP/WSDL is described in WSDL, an extension of the XML schema specification. It describes Web services both abstractly in terms of types, messages, operations and port types, and concretely in terms of transport protocol-specific, available addresses (endpoints) to invoke the service. REST uses the generic HTTP methods (i.e. GET, PUT, POST, DELETE, etc.) rather than specific operations. Thus, REST is derived directly from the principles of the WWW and also uses its advantages, such as simple error handling, through standardized HTTP error codes.

2.3.2 SOAP Versus REST

SOAP is a messaging standard, which defines an XML-based message format, specifies processing rules for messages, describes conventions, and allows mappings to different Internet transport protocols (including HTTP).

SOAP messages are always XML documents consisting of three parts: A virtual envelope, the *SOAP envelope*, contains two elements: the optional *SOAP header* that may contain information such as routing and security details (authentication and authorization), and the mandatory *SOAP body*. The body element accommodates the actual message. It may include information on the data exchange or instructions for a remote procedure call.

SOAP is based on the *chain of responsibility* pattern; it is possible to define a series of distributed steps to process a SOAP message. This results in interesting options for Web service intermediaries or for the integration of specialized middleware functionality to support different qualities of service as addressed by the Web Services Platform Architecture.

REST, however, uses the HTTP semantics and thus prescribes a stateless communication (i.e. the server does not maintain any client state information). This is why REST features point-to-point connections where any necessary information is encoded in the message itself. This facilitates, e.g. the interposition of caches or the replication of servers.

Chapter 3
Cloud Architecture

Cloud architectures can be analyzed from two different perspectives, i.e., from an organizational or from a technical point of view. The organizational view, which will be discussed in Sect. 3.1 below, makes a distinction based on the extent to which the users' and providers' organizational units are separated from each other, while the technical view in Sect. 3.2 is oriented towards functional features. Thus, the organizational view corresponds to the deployment model and the technical view to the service models as specified in the NIST definition [105] we presented in Sect. 1.2.

3.1 Public, Private, and Hybrid Clouds

A public cloud (also called 'external cloud') comprises all cloud offerings where the providers and the potential users do not belong to the same organizational unit. The providers make their cloud accessible to the public and usually offer a self-service Web portal where the users can specify their desired scope of services. For this purpose, no overall framework agreement is necessary, but the contractual obligations are entered within the scope of the performance specifications. The services are billed on the basis of the resources actually used in the corresponding period.

In contrast to this model, the providers and users of a so-called private cloud (also referred to as 'internal cloud' or 'IntraCloud') belong to the same organizational unit. The main reason why a private cloud would be preferred over a public cloud is usually security: In the private cloud, control over the data remains with the users or their organization.

This way, sensitive information, such as design plans or manufacturing data, can – supposedly – be better protected and regulatory measures, for example with respect to personal health data, can be complied with.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_3,

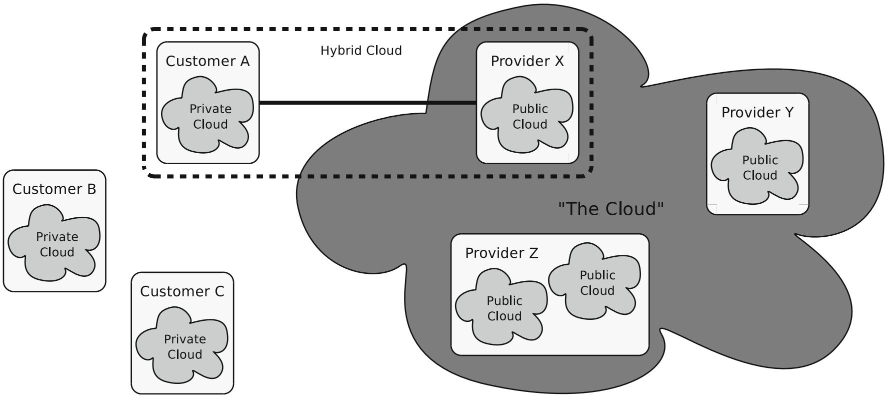


Fig. 3.1 Public cloud, private cloud, and hybrid cloud

While the services in a private cloud operate on resources belonging to the organization, some developers try to realize the interfaces by using the same technology as for the public cloud.

The objective is both to make sure that the tools which are available in the public cloud can also be used in the private cloud, and to keep the door open to be able to scale the applications that were first developed for the private cloud for later use in the public cloud.

Scenarios where services from the public cloud and from the private cloud are brought together, are referred to as *hybrid cloud*. With a hybrid cloud, certain functionalities or load peaks are usually transferred to the public cloud, while normal operation relies on the organization's private resources. According to the security considerations mentioned above, care has to be taken that only non-critical functions or data are transferred. Figure 3.1 shows how the three types, i.e., public cloud, private cloud, and hybrid cloud are positioned with respect to each other.

3.2 The Technical Landscape of Cloud Services

As described in the previous chapters, cloud computing allows the deployment and use of heterogeneous IT infrastructures, platforms, and applications as Web services. So it is no surprise that the existing cloud services landscape presents itself as correspondingly diverse, and, at the first glance, heterogeneous with respect to functionality and objectives. Based on a conceptual architecture, we will now draw up a cloud computing map which allows us to categorize the available cloud offerings and technologies and then compare the instances in each category. This enables prospective cloud users to determine the available choice of offerings or complementary technologies for their particular use case and find the optimum solution. The discussion in this section is based on [20].

The architecture depends on a stack or layer model, i.e., it consists of several layers which are arranged according to their degree of abstraction. In this context, the higher and more abstract layers may consume the services of the lower and more concrete layers in order to realize services of their own.

Layering within the stack is not strict, i.e., a higher layer may consume the services of all underlying layers and not only those of the one directly below it.

Each layer is characterized by properties of its own and some layers are subdivided into sub-layers. Cloud services having their own services and interfaces which make them eligible to be assigned to multiple layers, are mapped to the highest layer possible because this is usually the layer through which potential users are primarily addressed. With cloud computing being characterized by a dynamic evolution at the time this book was written, this classification is not intended to be complete, but rather represents a snapshot of the currently known or particularly archetypal cloud services.

The three main layers, which subdivide the resulting stack shown in Fig. 3.2, follow the *Everything-as-a-Service* paradigm (XaaS) with its four main representatives: Infrastructure as a Service (IaaS), Platform as a Service (PaaS),

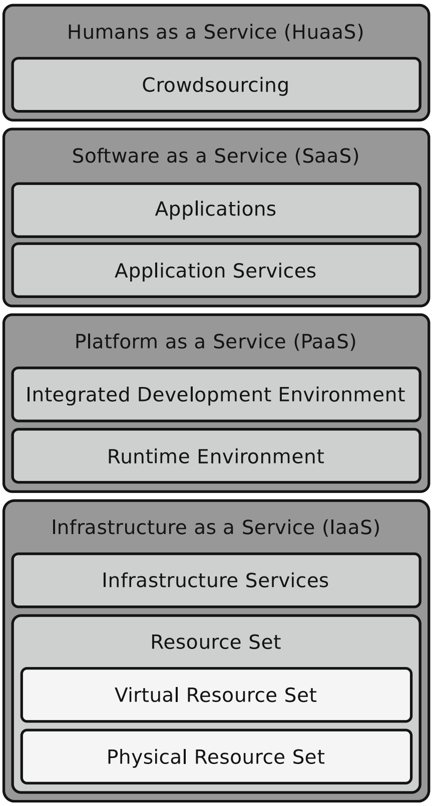

Humans as a Service (HuaaS)
Crowdsourcing
Software as a Service (SaaS)
Applications
Application Services
Platform as a Service (PaaS)
Integrated Development Environment
Runtime Environment
Infrastructure as a Service (IaaS)
Infrastructure Services
Resource Set
Virtual Resource Set
Physical Resource Set

Fig. 3.2 Cloud architecture stack diagram

Software as a Service (SaaS), and Humans as a Service (HuaaS). We will define and illustrate these in more detail in the following sections.

3.3 Infrastructure as a Service

The IaaS layer gives the users an abstracted view on the hardware, i.e., computers, mass storage systems, networks, etc. This is achieved by providing a user interface for the management of a number of resources in the resource set sub-layer (RS). It enables the users to allocate a subset of the resources for their own use. Typical functions available from the user interface include creating or removing operating system images, scaling required capacities, or defining network topologies. Moreover, the interface provides the required functionality for operations, such as starting and stopping operating system instances.

The items offered in the resource set can be further subdivided into the physical resource set (PRS) class, whose members represent and offer proprietary physical hardware, and the virtual resource set (VRS) class, which is built on top of virtualization technologies such as Xen [14] and thus make virtual instances available. Examples of PRS cloud services are Emulab [16] and iLO [15]. VRS cloud services include Amazon EC2 [40], Eucalyptus [22], Nimbus [17], and OpenNebula [26]. Even though the VRS services with their simplifying virtual resources are much more common, the PRS services also have their right to exist, e.g., when, for stability or performance reasons or due to specific hardware requirements, indirection by a hypervisor should be avoided, but at the same time the convenience and scope of functions for administering the cloud services are indispensable.

Besides the resource set sub-layer described here, the infrastructure services too belong to the IaaS layer. Compared to the resource set offerings, the infrastructure services have a narrower application focus. For example, there are dedicated infrastructure services for calculation tasks (Hadoop MapReduce [97]), for mass storage (Amazon S3 [46], Zumodrive [143], Dropbox [69]) or for networks (OpenFlow [24]). For an extended and commented list of IaaS cloud services and tools, see Table 3.1.

3.4 Platform as a Service

The cloud services provided on the PaaS layer are usually not targeted to end users, but rather to developers. These are programming environments (PE) and execution environments (EE) where proprietary software written in a specific programming language can be executed. Typical examples of these programming environments are Django Framework [68] or Sun Caroline [61]. They extend existing programming languages, e.g., by adding class libraries which have a specific

Table 3.1 Infrastructure as a service: offerings and tools

Organization	Cloud service	Reference	Description
Amazon	Elastic Compute Cloud (EC2)	[40]	Virtual servers
Amazon	Dynamo	[12]	Key-value pair store
Amazon	Simple Storage Service (S3)	[46]	Mass storage
Amazon	SimpleDB	[47]	Database as a Service (DaaS)
Amazon	CloudFront	[36]	Content Distribution Network (CDN)
Amazon	SQS	[48]	Message queues
AppNexus	AppNexus Cloud	[57]	Virtual servers
Bluelock	Virtual Cloud Computing	[60]	Virtual servers
Bluelock	Virtual Recovery	[60]	Virtual server recovery in case of failures
Cloud.com	CloudStack	[65]	Open source IaaS
Dropbox	Dropbox Cloud Storage	[69]	Mass storage
Emulab	Emulab Network Testbed	[16]	Emulation of logical networks for experimental purposes
ENKI	Virtual Private Data Centers	[73]	Need-based provisioning of virtual data centers
Reservoir	Open Nebula	[26]	Virtual open source server pools
FlexiScale	FlexiScale Cloud Computing	[76]	Virtual servers
GoGrid	Cloud Hosting	[82]	Virtual servers
GoGrid	Cloud Storage	[82]	Mass storage
Google	Google Big Table	[9]	Distributed storage of structured data
Google	Google File System	[89]	Distributed file system
HP	iLO	[15]	Lights-out management
HP	Tycoon	[126]	Market-based allocation of cluster resources
Joyent	Accelerator	[99]	Virtual servers
Joyent	Connector	[99]	Pre-configured virtual servers
Joyent	BingoDisk	[99]	Mass storage
University of Chicago	Nimbus	[108]	Open source IaaS
Nirvanix	Storage Delivery Network	[99]	Mass storage
Openflow	OpenFlow	[109]	Local network simulation
OpenNebula Project	OpenNebula	[114]	Open source IaaS
Rackspace	Mosso Cloud Sites	[120]	Pre-configured virtual servers
Rackspace	Mosso Cloud Storage	[120]	Mass storage
Rackspace	Mosso Cloud Servers	[120]	Virtual servers
Skytap	Skytap Virtual Lab	[127]	Hybrid cloud test environments
Terremark	Infinistructure	[131]	Virtual servers
todo GmbH	flexIT	[132]	Virtual servers
Eucalyptus Systems	Eucalyptus	[74]	Open source AWS implementation
Zimory	Zimory Public Cloud Market	[141]	Distributed, market-based allocation of IaaS resources
Zumodrive	Hybrid Cloud Storage	[143]	Mass storage
10gen	Mongo DB	[33]	Database for cloud storage

Table 3.2 Platform as a service: offerings and tools

Organization	Cloud services	Reference	Description
Akamai	EdgePlatform	[35]	Content, site, application delivery
Facebook	Facebook Platform	[75]	Environment for applications in the Facebook social network
Google	App Engine	[85]	Scalable execution environment for Web applications
Microsoft	Azure	[59]	Programming and execution environment for Windows applications
Microsoft	Windows SkyDrive	[103]	Platform for data synchronization between heterogeneous end devices
NetSuite	SuiteFlex	[106]	Business process development tool of NetSuite
Salesforce	Force.com	[78]	Development and operation of Salesforce CRM extensions
Sun	Project Caroline	[61]	Development and operation of distributed Web applications
Zoho	Zoho Creator	[142]	Development and operation of database-based Web applications

application focus. The application thus created then runs in an execution environment which is, from the project perspective, decoupled from the programming environment.

Well-known examples of cloud-based execution environments are Google App Engine [85], Azure by Microsoft [59] or Reasonably Smart by Joyent [99]. In the case of Microsoft Windows Azure, a variety of tools and different programming languages can be used on the basis of the Azure environment. Google App Engine supports the creation and execution of Web applications written in Python or Java.

An example of how services from different architectural layers interact or are built on top of each other, is the AppScale [58] programming environment, an open source re-implementation of Google App Engine. AppScale implements their functionality using the Eucalyptus infrastructure service mentioned above.

The interchangeability of parts of the stack described above, which is due to re-implementation in open source projects, not only fosters technical alternatives, but might, in some cases, prevent users from becoming too dependent on a particular commercial cloud vendor (so-called *vendor lock-in)*—an issue that has often been the target of criticism. Table 3.2 shows and describes further PaaS offerings and tools.

3.5 Software as a Service

Cloud software applications that directly address the end user belong to the SaaS layer. This model frees the customers from the need to install the software locally and thus to provide the required resources themselves. Seen from the cloud architecture perspective, the SaaS offering can be developed and operated by the provider on the basis of a PaaS or IaaS offering.

Table 3.3 Software as a service: offerings and tools

Organization	Cloud services	Reference	Description
Adobe	Photoshop Express	[34]	Online image processing
fluidOps	eCloud Manager SAP Edition	[77]	SAP Landscape as a Service
Google	Google Docs	[88]	Online office applications
Google	Google Maps API	[90]	Service for the integration of maps and geo-information
Google	OpenSocial	[115]	Generic programming interface for the integration of social networks into applications
OpenID Foundation	OpenID	[113]	Distributed cross-system user identity management system
Microsoft	Windows Live	[103]	Online office applications
Salesforce	Salesforce.com	[125]	Extensible CRM system

Within the SaaS offerings, we can distinguish application services, whose functionality is mainly based on a single, simple application, and full-fledged, complex applications. End users can directly access some of the application services, such as Google Maps [90]. Other services are provided as components, e.g., the OpenID [113] user management or the integration of social networks in applications through OpenSocial [115]. Linking to these application services can either be done by common static binding or by using so-called mash-ups [6] which allow for light-weight, flexible binding.

Applications, such as Google Docs [88], Microsoft Windows Live [103], or the central Salesforce.com [125] CRM system feature a much higher complexity and a wider range of functions. The latter also shows how a full-fledged application can be enhanced by adding other application services and how the SaaS and PaaS layers can interact here: Salesforce.com offers third-party providers a platform where they can develop and operate free or paid application services which add to and extend the functionality of the existing central CRM system. For further applications and application services, see Table 3.3.

3.6 Humans as a Service

HuaaS (Humans as a Service) is the top layer of the cloud computing stack. This shows that the cloud paradigm is not restricted to IT services, but can also be extended to include services provided by human beings as resources. Since humans have certain capabilities that outperform computer systems, their technical integration as resources is a matter of particular interest. Some tasks, such as translation or design services, where creativity is an important asset, are better accomplished by this special resource.

Within the HuaaS layer, the predominant sub-category is crowdsourcing where a group of human resources use the Internet to perform tasks of varying complexity and scope for a customer. A typical example of crowdsourcing again comes from Amazon: Amazon Mechanical Turk [44] offers an interface where – mainly fine-grained – tasks can be assigned to interested human resources who are paid an equally small amount for their efforts.

Amazon Mechanical Turk works like a marketplace for crowdsourcing offerings. However, so far only persons living in the U.S. and having a bank account there may place their HuaaS tasks. Persons outside the U.S. may only accept tasks from Amazon Mechanical Turk.

Typical crowdsourcing tasks require only little or no previous knowledge so that a potentially large, dynamically scalable workforce is available to complete the tasks posted there. In other crowdsourcing offerings, all participants can benefit from artifacts provided by others, and they can judge their quality based on other users' ratings. YouTube [140] is a typical example of such an offering. Prediction markets, for instance IOWA Electronic Markets [7], aggregate predictions of individuals related to future events. Based on the majority opinion, they predict the outcome of elections or sports events. Another interesting example of crowdsourcing is the investigation of the British MPs' expenses scandal in summer 2009 organized by The Guardian. The newspaper published more than 450,000 documents [96] and receipts online, so that the readers could view them and evaluate their relevance. The work done by so many people significantly contributed to uncover the facts of the scandal within a very short period of time.

3.7 Other Categories of Cloud Services

Besides the established service categories, i.e., IaaS, PaaS, SaaS, and HuaaS, there are a large number of special services that are partly derived from them. Some examples:

- High Performance Computing as a Service (HPCaaS). HPCaaS is intended to make high performance computing available as a service. The focus of high-performance computing is to minimize latency between the connected resources and to optimize the data throughput. For this purpose, it is necessary that the instances of a cloud infrastructure are physically located near each other. OpenNebula is an example of an IaaS solution meeting this constraint. Several companies offer corresponding solutions: Amazon Cluster Compute Instances [38], Gridcore Gompute [93], Penguin Computing on Demand [119], Sabalcore HPC on Demand [123], UNIVA UD UniCloud [134].
- Landscape as a Service (LaaS). Complex software with no or only limited multi-tenancy support, such as SAP R3, is offered as an SaaS. This offer is targeted at companies which aim at outsourcing their entire data center including hardware, software, maintenance, and deployment. Such solutions are offered by fluid Operations [77] and Zimory [141].

Chapter 4
Selected Cloud Offerings

In the preceding Chap. 3, we discussed the cloud services architecture from a technical point of view and thus mapped the landscape of cloud computing. Actual cloud offers were only mentioned briefly. The aim of this chapter is to give an overview of the offerings from vendors such as *Amazon*, *Google*, *Microsoft*, and *Salesforce* and to present their capabilities as well as their intended use.

Our shortlist is made up of the offerings mentioned because they are well-known and popular, and further, each of them plays the role of a prototype in its own cloud architecture category. However, it must be stressed at this point that the field of cloud computing is by far not limited to these major players. With their innovative developments, especially mid-size and start-up companies play an important role here. This is accounted for, among others, by the sections about cloud gaming and cloud operating systems which describe more recent application classes of cloud computing.

4.1 Amazon Web Services

Amazon Web Services (AWS) [50] is the umbrella name for all cloud offerings from Amazon. The reason why this company, which is still mainly perceived as an e-commerce shop, has committed itself to cloud computing is easy to see: Amazon is compelled to maintain a considerable amount of IT resources to cope with the seasonal peak periods, i.e. before Christmas or Thanksgiving. A large part of these resources, however, sits idle for the rest of the year. Hence, the business idea to offer the unused resources for money to third parties during low-usage times was born. Currently, Amazon offers the following cloud services:

- Amazon Elastic Compute Cloud (Amazon EC2) [40]: EC2 enables users to manage self-configured virtual servers running on Amazon's data centers using web services. For more details, see Sec. 4.1.1.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_4,

- Amazon Simple Storage Service (Amazon S3) [46]: Amazon S3 provides a mass storage infrastructure with an essentially flat structure to hold large data quantities. S3 will be discussed below in Sect. 4.1.2.
- Amazon Elastic Block Store (EBS) [39] is used for persistent data storage in the context of EC2 instances. This service will be dealt with in Sect. 4.1.3.
- Amazon Simple Queue Service (Amazon SQS) [48]: Amazon SQS implements a messaging system based on message queues which accept messages transmitted by sender applications (publishers) as input. Registered receiving applications (subscribers) can then read these messages asynchronously (see Sect. 4.1.4). SQS is a nice tool to easily develop scalable applications in the cloud (see Sect. 4.1.7).
- Amazon SimpleDB [47]: Amazon SimpleDB is a cloud database, which implements a simple database model reminiscent of a slimmed-down relational database model. For more information, see Sect. 4.1.5.
- Amazon Relational Database Service (RDS) [45]: The Relational Database Service implements a relational MySQL database in the Amazon cloud based on an EC2 database instance.
- Amazon CloudFront [36]: Frequently, Content Distribution Networks (CDN) are used for fast data delivery in web applications. CDNs provide a redundant and geographically distributed data storage system. Each user receives the requested data from the node which is best suited for that case. Amazon CloudFront implements a Content Distribution Network on top of Amazon S3.
- Amazon Elastic MapReduce [41]: Amazon Elastic Map-Reduce allows to distribute data-intensive computations across any number of Amazon EC2 instances so that they can be scaled flexibly.
- Amazon Virtual Private Cloud (VPC) [49]: VPC facilitates the transparent integration of AWS EC2 resources into the customer's IT infrastructure. The external resources are integrated into the customer's network via a secure, encrypted VPN connection so that it is possible to build a hybrid cloud. For this service, Amazon charges the usual prices plus the data volumes transferred over the VPN connection.
- AWS Import/Export [52]: The option to transfer very large data volumes by shipping physical storage devices through regular mail appears to be anachronistic at first glance. But since there are numerous practical problems when transferring large data volumes over a network, especially when in the order of several petabytes, postal mail deliveries are preferred because they are faster, more reliable and cheaper than transfers over the Internet. If a secure network connection cannot be established, sending the data by regular mail is also advantageous in terms of data privacy. The data are stored in encrypted form on the storage devices and the owner is identified by means of a certificate. Once the storage devices have been integrated into Amazon's data centers, only the certificate holder is able to make them operational, enter the key and process the data.

Besides these cloud offerings, which are an official part of the AWS, more cloud services are available. These, however, are only to be considered as supporting or as partial functionalities, which depend on the above offerings. They will not be discussed here any further. In the near future, additional AWS offerings are to be expected, as the demand for a larger variety of cloud services will likely grow over time.

4.1.1 Amazon Elastic Compute Cloud (EC2)

The virtual servers constituting the Amazon Elastic Compute Cloud are at the core of the Amazon Web Services. To set up and permanently operate such a server, the following steps are necessary:

- Selection of the region: Amazon defined four geographical regions for their services, with different pricing for resource usage. These are the regions:
  - `US East` ⇒ Virginia
  - `US West` ⇒ California
  - `EU West` ⇒ Ireland
  - `Asia Pacific` ⇒ Singapore
- AMI selection: The virtual servers are created from Amazon Machine Images (AMIs). These are like a blueprint to be used when creating new virtual servers. Amazon provides pre-built images, which differ by the operating system and the software packages installed. AMIs from Amazon are for example available for various UNIX derivatives and also for Windows operating systems with different environments installed, such as for web applications. Besides Amazon, a number of third-party vendors, such as IBM, Oracle, and Sun, provide AMIs including proprietary software packages. End users as well can create their own images for later reuse. These AMIs can be published and put on the market using a product ID (paid instances). The operating system and the installed software packages are the main criteria when selecting a suitable AMI. To do so, users can go to the AWS web pages, which give access to a separate Web portal that includes assessment functions.
- Selection of the resource configuration and the availability zone: Another preliminary consideration refers to the selection of the proper resource size for the desired virtual server. The available resource configurations differ in the processor performance as well as in the memory and hard disk sizes. The customer is also required to select an availability zone in which the virtual server should run. The regions mentioned above are subdivided into availability zones, which are independent from each other in that they do not share critical components and therefore cannot fail at the same time for the same reason. A careful selection of the availability zone can therefore pay off in reduced latencies and improved reliability for redundant servers. The `US East` region offers four availability zones, while the `US West`, `EU`, and `Asia Pacific` regions currently only comprise two availability zones each.

- Generating a key pair: Users who want to access an Amazon E2 instance use the so-called public key identification method. Prior to first-time use, a key pair consisting of a public and a private key must be generated. The public key is associated to the user account which is also required for billing purposes, and the private key remains in the user's working environment.
- Launching an instance: Once the above parameters have been defined accordingly, the desired instance can be launched. After the launch, both a public IP address and a private IP address are assigned dynamically to the new virtual server. The public address can then be used to reach the computer from the Internet, while the private address makes the computer visible to other Amazon Cloud instances. Since both the private and the public addresses are assigned dynamically each time an instance is launched, they are not suitable for permanent server operation if the server needs to be restarted repeatedly during its life cycle. For this case, Amazon provides static/elastic IP addresses, which – once they have been reserved – are allocated time and again to the different new server incarnations. Elastic IPs are therefore well suited for example for DNS entries.
- Defining and configuring a security group: For security reasons, new instances are not immediately accessible from the Internet after their generation. They must rather belong to a so-called security group which defines a common set of security settings. For such a group, it is for example possible to also enable the access via the secure shell (SSH) using the corresponding port 22. Now, the virtual server can be reached via SSH; for the authentication, the public key method with the keys generated as described above is used.
- Storing instance states: When the state of a running instance changes during operation, this change is not persistent. Once an instance has been terminated, all changes are lost. In order to make state changes persistent beyond the runtime of a virtual server, the state must consequently be stored outside the instance itself. While Amazon S3 can be used for storing large, weakly structured data sets, it is less suitable for typical file system operations. This is where Amazon Elastic Block Store (EBS) comes in: It allows block-based access to a storage medium. For the EC2 instance, EBS appears like an external hard disk. With the EBS command set, it is possible to create storage volumes, associate them with EC2 instances, mount them as drives there, and use them as any other drive. After the volume has been closed and unmounted in a controlled manner, e.g., when terminating the instance, it can be made available to other new EC2 instances. However, an EBS volume cannot be shared by multiple instances. A particularly handy feature of the Amazon EBS and EC2 combination is the possibility to take snapshots of the current EBS state as desired and to back them up in Amazon S3.
- Creating your own AMIs: The easiest way to build customized images is to use a suitable pre-configured AMI as the basis which is initially put into operation together with an EBS volume, as described above. The next step is to install the specific add-ons on this AMI, as usually. Using a number of special commands, which bundle the original image with the individual customizations, the user-defined image can be created and stored in S3, protected by certificates. Once the

image has been registered, the customized instances can be used from within Amazon S3.

- Tool support: All procedures described above can be executed using command line commands. Those who find this too cumbersome have the choice of a number of tools which are available from the AWS website. Among these tools, AWS Management Console, AWS Toolkit for the Eclipse development environment, and ElasticFox as a plug-in for the Firefox web browser are outstanding. Each of these tools conveniently provides certain subsets of the AWS interface, which are specific to the user group addressed (see Sect. 5.3 and Chap. 9).

The operating costs for on-demand EC2 instances are made up of several individual fees: The fee for the instance itself depends on its performance and is billed on a hourly basis. Table 4.1 shows the price per hour (in USD) as it was in winter 2010 for instances running in the `East` region of the U.S. As the table shows, Linux instances are 10% cheaper than their counterparts in Europe.

The sizes indicating the capacities of the instances refer to the *EC2 Compute Unit* (ECU) standard, and take further properties of the virtual system into account. An ECU is equivalent to the compute power of a 2007 AMD Opteron or Intel Xeon CPU with a processor speed from 1.0 to 1.2 GHz.

A *Small Standard On-Demand Instance* corresponds to a 32-bit computer system with a single virtual processor core (one ECU), 1.7 GB of memory, and 160 GB of disk space. A *High Memory Quadruple Extra Large On-Demand Instance*, at the high end of the scale, features a 64-bit platform with 26 ECUs (distributed to 8 virtual cores with 3.25 ECUs each), 68.4 GB of RAM, and 1,690 GB of disk space.

The *Cluster Compute Instances* are 10 Gigabit Ethernet HPCaaS offerings from Amazon with increased I/O performance. Up to now, such instances are only

Table 4.1 Pricing for Amazon EC2 on-demand instances (USD per hour)

Micro On-Demand Instance		*Linux/UNIX*	*Windows*
Micro	`t1.micro`	0.02	0.03
Standard On-Demand Instance		*Linux/UNIX*	*Windows*
Small (Default)	`m1.small`	0.085	0.12
Large	`m1.large`	0.34	0.48
Extra Large	`m1.xlarge`	0.68	0.96
High CPU On-Demand Instance		*Linux/UNIX*	*Windows*
Medium	`c1.medium`	0.17	0.29
Extra Large	`c1.xlarge`	0.68	1.16
High Memory On-Demand Instance		*Linux/UNIX*	*Windows*
Extra Large	`m2.xlarge`	0.50	0.62
Double Extra Large	`m2.2xlarge`	1.20	1.44
Quadruple Extra Large	`m2.4xlarge`	2.40	2.88
Cluster Compute On-Demand Instance		*Linux/UNIX*	*Windows*
Quadruple Extra Large	`cc1.4xlarge`	1.60	n.a.
GPU Quadruple Extra Large	`cg1.4xlarge`	2.10	n.a.

available in the `US East` region and under Linux. Each of them has two quad-core Intel Xeon-X5570 Nehalem processors and 1,690 GB of disk space. Every *Cluster GPU Quadruple Extra Large Compute On-Demand Instance* additionally includes two NVIDIA Tesla-M2050 graphics processing units (GPU).

32-bit architecture systems are only used by the two `m1.small` and `c1.medium` instance classes. All other instance classes use systems featuring a 64-bit architecture. An exception to this rule is the `t1.micro` instance class for which both 32-bit and 64-bit instances can be used.

Besides the hourly operating costs paid from the time an instance is launched until it is terminated, the data transfer (billed by volume), the cost for a permanent elastic IP (billed by the hour), and the Elastic Block Store volume used (billed by storage space) add to the total costs over time. Moreover, there are other chargeable features to which the user can subscribe. In order to make it easier for the user to estimate the potential costs – in the light of the rather complex pricing model – Amazon provides a billing form on the AWS website which calculates the monthly costs for a given configuration or based on a certain expected utilization profile.

Besides dynamic instances, Amazon also offers reserved instances for permanent use. They can be purchased for 1 or 3 year terms with corresponding discounts. Thus, a *Small Reserved Instance* in Europe costs $227.5 per year ($350 for 3 years).

Another possibility to obtain instances at a lower price are the so-called spot instances. These refer to an auction model which enables the customers to bid for unused EC2 capacities. The price for spot instances is variable and is adjusted periodically by Amazon depending on the supply and demand for capacity. To bid on spot instances, customers specify a maximum price they are willing to pay per instance hour, further indicating the desired instance type, the region, and the number of instances required. As long as a sufficient number of spot instances are available at a lower price, the order will be executed and the instances will be launched. If the spot price exceeds the given maximum price, the instances will be terminated. As soon as the spot price falls below the given maximum price again, the instances are re-launched automatically. The spot price is updated every 30 minutes. This offering is particularly interesting for industry and research projects which must stick to a tight budget and do not require 24/7 uptime.

An example of how to use Amazon EC2 is shown in the appendix in Sect. 9.1.

4.1.2 Amazon Simple Storage Service (S3)

Amazon S3 is a cloud-based mass storage system with a very simple storage structure. Stored web objects with a size of up to 5 GB are stored in so-called buckets. Buckets cannot be nested hierarchically, but each web object can be identified directly by its bucket name together with its object name. Consequently, Amazon S3 is no conventional file system, although there are third-party tools and cloud services, such as JungleDisk [100], that allow users to access storage resources in S3 just as they do with remote hard disks.

By definition, Amazon S3 is accessed via web services using either a SOAP or a REST API. The latter is preferable, especially if large objects are to be dealt with, since REST usually handles them better than SOAP. As with EC2, convenient graphical user interfaces are available for S3; these will be explained further in Sect. 5.3. For direct access to Amazon S3 from the command line, the s3cmd tool [121] is available; its functionality is illustrated in the appendix in Sect. 9.4.

4.1.3 Amazon Elastic Block Store (EBS)

The EBS storage service allows customers to create data stores, so-called storage volumes, with sizes from 1 GB to 1 TB within each availability zone.

A new EBS volume behaves just like an unformatted block device. Users can create any file system on top of the EBS volumes which can be handled like a USB stick.

It is important to mention that an EBS volume can only be mounted to one single instance, which, in turn, must be located in the same availability zone. An EBS volume implements persistent storage, i.e. it preserves the data after termination of the instance.

A usage example for Amazon EBS is shown in the appendix in Sect. 9.2.

4.1.4 Amazon Simple Queue Service (SQS)

The SQS message queue has special importance within the range of AWS cloud offerings because it can be used effectively to scale applications. A sender (publisher) can place messages in a queue, which can then be read out and processed by a registered recipient. To dissociate different components in an application, a service-consuming component can place its jobs as requests in the queue from where they are fetched by service-providing components. Skillful programming allows to operate critical components simultaneously on multiple EC2 instances, using the path thus defined. This way, bottlenecks existing with certain components can be eliminated flexibly at runtime and the system's overall performance is no longer limited by the bottleneck. The example in Sect. 4.1.7 illustrates the resulting system architecture.

The SQS interface provides the following services which are not usually started by a user by entering a command, but by corresponding web service calls issued by the associated components.

- CreateQueue: creates a queue in the AWS user context
- ListQueues: lists the existing queues
- DeleteQueue: deletes a queue
- SendMessage: places a message in a queue

- ReceiveMessage: reads one (or more) messages from a queue
- ChangeMessageVisibility: explicitly sets the visibility of a read message for other potential readers
- DeleteMessage: deletes a read message
- SetQueueAttributes: sets queue attributes, e.g., the interval between two read operations of the same message
- GetQueueAttributes: reads queue attributes, e.g., the number of messages currently in the queue
- AddPermission: enables shared access to a queue from multiple user contexts
- RemovePermission: disables the shared access by other user contexts

4.1.5 Amazon SimpleDB

Amazon SimpleDB is not designed for complex database schemes or transactional properties, but intended to provide simply structured, yet highly reliable data storage which is considered to be sufficient for a wide range of applications. The database administration and optimization tasks are thus reduced to a minimum. For applications which depend on the performance and the comprehensive functionality of today's commercial relational database systems (RDBMS), Amazon RDS is the better choice (see Sect. 4.1.6).

In line with the limited functionality of SimpleDB, its interface is also restricted to a few simple web service calls. This should ensure both ease of learning and a user-friendly behavior:

- CreateDomain, ListDomains, DeleteDomain: create, list or delete domains. Domains correspond to the tables existing in relational databases. Each command can only address one single domain at a time.
- DomainMetadata: reads metadata of a domain, such as the current storage space requirements
- PutAttributes: adds or updates a record based on a record identifier and attribute/value pairs
- BatchPutAttributes: simultaneously triggers multiple insert operations to increase the performance
- DeleteAttributes: deletes records, attributes, or values
- GetAttributes: reads an identified (partial) record
- Select: queries the database using an SQL-like syntax, but without being applied to multiple domains (as with Join)

4.1.6 Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) [45] is a PaaS that makes it easy to set up, operate, and scale a relational database in the cloud. Similar to the

EC2 operating model, this service provides elastic capacities and at the same time handles time-consuming database administration tasks. Thanks to automated database backups and snapshots, the Amazon RDS is highly reliable: a database instance can be recovered for any point in time or recovery point that lies within the agreed retention period.

Amazon CloudWatch allows users to monitor the utilization of the computing and storage capacities of their database instances and to scale the available resources vertically using a simple API call as needed. In connection with highly demanding applications involving many read operations, it is possible to scale out by launching so-called Read Replica instances. A corresponding high-availability offering allows the provisioning of synchronously replicated database instances without additional costs in multiple availability regions as a safeguard against failures at a single location. This way, it is possible to mask maintenance windows as RDS switches the database services transparently between the locations.

Amazon RDS enables access to all MySQL database functions so that it is a no-brainer to migrate existing applications while maintaining the preferred database tools and programming languages. If an existing application already uses a MySQL database, the data can be exported with `mysqldump` and then be piped directly into Amazon RDS. For larger databases of 1 GB or more, we recommend to create a database schema in RDS first, then convert the data into a flat file, and finally import it into the RDS instance using the mysqlimport utility. The same method can be used when exporting data from the database services.

Amazon RDS selects the optimum configuration parameters for database instances, taking the relevant computing resources and storage capacity requirements into account. However, it is also possible to change the default setting through configuration management APIs. Since RDS is implemented as a PaaS offering, it is not possible to set the database parameters by directly accessing the servers through the SSH.

For the management of its database services, Amazon not only offers command line tools and libraries for various programming languages [51], but also a convenient web-based management console [53].

A usage example for Amazon RDS is shown in the appendix in Sect. 9.3.

4.1.7 The Amazon Web Services as 'Team Players'

In the context of application development, the different AWS services can be used either as standalone utilities or in interaction with each other. [29] exemplifies an application which uses various AWS services to perform search requests against millions of web documents. This application, known as *GrepTheWeb*, is currently in production at Amazon.

This scenario (see Fig. 4.1) is characterized by the interaction of SQS, SimpleDB, EC2, and S3. A number of SQS queues store both incoming requests as well as intermediate results created when processing these requests. This

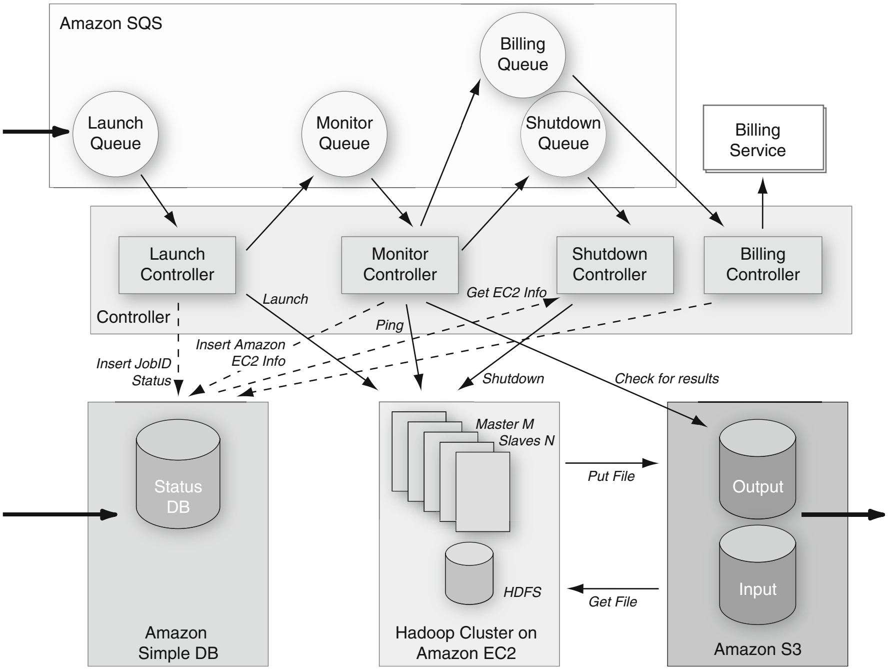


Fig. 4.1 AWS interaction example: GrepTheWeb (Source: Amazon)

decouples the processing phase from the receipt of the requests. Search requests are formulated as regular expressions which are placed as *launch messages* in an initial SQS *launch queue*.

For each processing step, there is a controller component *(launch controller, monitor controller, shutdown controller, billing controller)* which reads data from a queue and writes the results to the next queue. This means that the processing steps are executed asynchronously by the respective controller and that the controllers are independent from each other.

The controllers use SimpleDB, EC2, and S3. The former contains status information, logs, and user data required for requests. Thus, each controller queries and writes to SimpleDB. S3, on the other hand, holds the data to be searched (hundreds of terabytes of Web documents) and the results from the search request. S3 is accessed via a Hadoop cluster made of EC2 instances (see also Sect. 6.9). These are created by the *launch controller* monitored by the *monitor controller*, and terminated by the *shutdown controller*.

GrepTheWeb is an example of a modular application architecture which provides for processing steps, decouples them using queues and controlling them using independent controller components. All components are lightweight and communicate in a service-oriented manner via http requests using XML (Extensible Markup Language). This architecture illustrates how different basic services can be combined when developing scalable multi-tenant systems which are able to dynamically serve different workload situations and – above all – are tolerant, e.g., in the event of individual node failures.

4.2 Google Cloud Services

Like Amazon, Google offers a wide range of different cloud services. Some of them will be presented in the next sections.

4.2.1 Google App Engine

Google App Engine [85] is a PaaS that includes a programming environment, tool support, and an execution environment. This 'instrumentation' can be used to develop web applications for the scalable Google infrastructure. Google App Engine virtually frees web application designers from any tasks involving server administration so that they can focus on developing the required application functionality.

Google App Engine provides environments for the Python and Java programming languages. For these languages, a local execution environment is available where designers can test and trial their web applications during the development phase. Once the application is ready to go live, it can be moved to the Google infrastructure and run there. Thus, the web application benefits from the reliability and scaling technologies employed by Google for their own final products, such as the well-known search engine or the Google Docs online office suite. For private testing or basic, non-commercial systems App Engine is free of charge.

Further to the usual functions included in the Java libraries anyway, the platform gives access to a number of additional services (in the sense of cloud computing: application services). Part of them can be addressed via the standard Java interfaces in order to facilitate the portability and usability of the developed code even outside the App Engine context:

- Data storage: For persistent data storage, Google App Engine uses the so-called datastore, a schemaless, object-oriented database that includes a query engine and guarantees atomic transactions. The datastore implements a proprietary interface, but also the Java Data Objects (JDO) and Java Persistence API (JPA) standard interfaces.
- Email integration: The functionality for email communication can be embedded in Web applications using Google email accounts. This is another application service which can be used both though a proprietary interface and a subset of the standardized JavaMail interface.
- User management: Authentication of Web application customers is done through Google accounts for which a proprietary interface is available. The

users are redirected to the Google Accounts login page where they can enter their account information. A basic rights management system distinguishes normal users from administrators and thus controls the access to protected functions.
- There are more services which add to the ones mentioned above, e.g., services for fast data caching, for linking to other web applications using HTTP(S), or for efficient editing of images to be displayed.

Google App Engine not only supports Web application developers by providing the local runtime environment mentioned above and the associated automation of transfer and deployment on the Google servers. What is more, many other tools, e.g. Google Plugin for Eclipse are available for the development of Google Web Toolkit (GWT) applications. Even when it comes to the App Engine pricing system, Google has the needs of potential developers in mind: each developer benefits from free quotas for CPU load, storage use, data transfer, etc., which can be used up on a daily basis and which are usually sufficient to run developer systems and basic web applications. If it is predictable that more resources will be needed, they can be purchased as an option. Prices for commercial use are comparable to those for the Amazon Web Services.

In discussions about cloud computing, many people rightly object that major obstacles arise from the use of proprietary technologies. With many PaaS offerings, the code of the developed applications can only be reused to a very limited extent or not at all. The web application provider is permanently tied to the cloud operator and thus to their pricing and quality of service standards. For Google App Engine however, a compatible, open source platform called AppScale [58] has emerged which provides the ability to migrate Web applications to an EC2-compatible public or private cloud IaaS (see also Chaps. 6 and 9). The project, which was started at the University of California in Santa Barbara, meanwhile is actively supported by Google themselves.

Another free re-implementation of Google App Engine is TyphoonAE. It enables the use of App Engine applications in any Linux environment as well as under Mac OS X.

4.2.2 Google Storage

Google Storage [92] is a storage service which allows to store web objects in Google's data centers. As with Amazon S3, service access is based on the REST model.

Google Storage provides the same functionality as Amazon S3: objects are placed in buckets and a single bucket cannot contain nested buckets. Users can define the access to their own objects and buckets based on an access control list.

The GSUtil [94] command line tool offered by Google supports both Google Storage and Amazon S3. With GSUtil, users can create their own buckets, upload objects, and set access rights using a browser.

Currently (i.e. in November 2010), the service is still under development and can only be accessed by developers in the U.S.

Storing one GB of data is charged at $0.17 per month. Uploads cost $0.10 per GB and downloads range between $0.15 and $0.30 per GB (depending on the region). For 1,000 PUT, POST, and LIST requests, Google charges $0.01. The same rate applies to 10,000 GET and HEAD requests.

4.2.3 Google Cloud Print

Google Cloud Print [87] is another interesting cloud computing concept. This service allows any application to print to any output device on the Internet. Especially users of mobile end devices might greatly benefit from this option. While modern Internet-enabled devices, such as notebooks, touchpads, and mobile phones, are becoming more and more widespread, it is often difficult or even impossible to set up a local printer that can be used by these devices. The lack of suitable printer drivers, the partly insufficient devices resources and the variety of operating systems in use add to this issue.

With Google Cloud Print, print jobs are sent to a service which directly forwards them to a Cloud Print-compatible network printer; for this purpose, a special authorization or accounting procedure is used. As long as a printer is connected to the Internet, the service can be set up for worldwide use. If the printer is incompatible with the Cloud Print technology, it must be connected to a server where a suitable proxy is installed. In this case, the print job data is first converted to a compatible printer language. For this purpose, the required set of printer drivers is made available by the Cloud Print provider. Then, the print job is transmitted to the server which finally forwards the preprocessed print jobs to the selected printer. Google Cloud Print represents an unconstrained standard which can be implemented freely by the industry. Many manufacturers, such as HP, meanwhile offer a number of low-priced, compatible printers.

This development has the potential to allow a broad range of service offerings to spread around this technology, including services which go far beyond the simple replacement of centralized print job management in companies and universities. In the future, university students will have the possibility for example to directly upload their lecture notes from their touchpad to a print provider's site to have them printed and bound. Due to economies of scale, the provider is able to offer this service at a very good price, including free delivery. The standardized interface provides the ability to greatly automate and streamline certain tasks, particularly the cross-company processing of bulk mail, quality prints, etc. What is more, a new distribution model for publishing houses is beginning to materialize: The installation of machines in the public on which – controlled by a mobile device – electronic magazines, newspapers, or books can be printed on the spot.

4.3 Windows Azure

Windows Azure [59] is Microsoft's cloud computing platform for the execution of software in Microsoft's data centers. Microsoft operates the platform and provides resources with a guaranteed availability of 99.9% on redundant systems.

The Windows Azure platform comprises a compute service for running applications, a storage service for storing data, and an SQL service for providing highly available relational databases in the cloud. The storage service can be used to store large objects containing text or binary data. Based on this service, Windows Azure Drive allows to format a binary data object to be used as an NTFS volume.

A queue service ensures a reliable data exchange between the components. Finally, the virtual network service constitutes the basis for a transparent communication between local and remote resources. It can be used to integrate Windows Azure services into a local Active Directory.

In addition, the platform includes the Windows Azure AppFabric which uses secure connectivity to bridge traditional IT systems to cloud applications. With the Azure Service Platform, software products can be installed as cloud services on the Internet, or alternatively as applications in the in-house data center. The two methods can also be combined to implement a flexibly scalable hybrid cloud. For this purpose, web and business developers have the choice of a variety of established tools, such as Microsoft .NET, Visual Studio, or many other products which are available as commercial or open source software. Applications can be developed and tested locally before they are finally uploaded and published to the Azure cloud.

For the use of Windows Azure services, three different roles have been defined:

- The Web role supports the development and execution of web applications with Internet Information Server 7
- The Worker role provides supporting services to the Web role, e.g., in multi-tier applications
- The VM role allows the execution of a virtual machine on Windows Server 2008 R2. The image is stored on a virtual hard disk in the Azure storage service. This role further provides functionality for granting privileges and using Remote Desktop connections.

The usage-oriented Windows Azure pricing model is based on what is actually used so that IT solutions can be deployed without up-front investment in hardware and software and more compute power can be added later as required. Pricing is in line with Amazon's or Google's model (see Table 4.2).

Table 4.2 Fees for Windows azure compute instances (USD per hour)

Instance type	CPU (GHz)	Memory	Disk (GB)	I/O	Costs
Extra small	1.0	768 MB	20	Low	0.05
Small	1.6	1.75 GB	225	Moderate	0.12
Medium	2 × 1.6	3.5 GB	490	High	0.24
Large	4 × 1.6	7 GB	1,000	High	0.48
Extra large	8 × 1.6	14 GB	2,040	High	0.96

4.4 Salesforce.com

Salesforce.com [125] is a leading cloud provider for customer relationship management (CRM) software. In the sense of Chap. 3, Salesforce.com is an SaaS offering which has been complemented by a PaaS offering where independent third parties can develop and offer add-on software. The Salesforce.com portfolio consists of four major parts:

- The central part is a CRM SaaS offering called Salesforce. It provides a web-based solution for sales, marketing, customer service, partner management, and others. The central component is available in several bundles that reflect different capabilities and numbers of users; it is usually paid for on a monthly or yearly basis. As a classic SaaS offering, this multi-tenant application runs on the Salesforce.com servers and does not have to be installed locally.
- Force.com [78] is the name of a PaaS offering which allows customers or independent software vendors (ISVs) to develop their own web-based business applications and run them on the salesforce.com infrastructure. For more technical details on Force.com, see below.
- The business applications developed on Force.com can be obtained on the AppExchange [56] marketplace: the choice includes free and paid apps. Via Force.com, the applications are pre-integrated with the Salesforce.com CRM, and their functionality often complements the latter or is fine-tuned for particular industries.
- Both Force.com and Salesforce.com are accompanied by organized user communities named DeveloperForce [67] and Salesforce.com Community [124], which provide both user networking and professional consulting services offered by Salesforce.com and their partners.

The Force.com platform includes development tools and a programming environment suitable for example to develop the logic of an application in Apex, a programming language with a Java-like syntax. Other features are the support of user interface development with Visualforce, the integration of software testing procedures, or the connection of external Web services through a dedicated API.

For the development of custom web applications, a programming model with a highly data-centric approach is used: The development of an application usually starts with the creation of an object model which will later hold the application data. For the data elements, constraints can be defined which improve the data quality. Just like workflows and acceptance processes, these two initial steps can be defined directly from within the development environment, using the available data. Use of the Apex programming language is only required if a more complex application logic must be implemented. The developer tools are not only helpful in the essential steps mentioned above required to meet functional and non-functional requirements, but also in the context of the organizational workflows which are necessary to pack the developed application and deploy it as an offering to potential customers. Developers who wish to learn how to write programs on Force.com can obtain comprehensive material in the form of tutorials, manuals, reference documents, and code samples on the web-based community portals.

4.5 Cloud Gaming

The term ‘cloud gaming’ encompasses all offerings which provide the ability to play high-end video games on low-end devices, such as TV sets or computers with a low compute power, but also on mobile end devices, such as touchpads and mobile phones. With the recent introduction of 3-D TV sets, even a platform for 3-D games is now available.

The video games played in cloud gaming run on the provider’s powerful server farms. The only function of the target device is to display the game content which is transmitted as a compressed video stream. User input is sent to the provider and evaluated there. A prerequisite for cloud gaming is the availability of a broadband Internet connection with low latency to ensure short response times and make sure the flow of the game is not interrupted. A benefit of a centralized gaming system located on the provider’s servers is that it facilitates a faster interaction between the players in multi-user online games. The necessary compression, however, results in visualization trade-offs.

Cloud gaming customers benefit from the fact that they neither have to buy a gaming console nor a powerful computer nor the desired games. Instead, they pay a certain amount for the time they play the game. The game manufacturers, on the other hand, enjoy the beneficial side effect that with cloud gaming, they no longer have to deal with the problem of pirated copies. Popular providers of cloud gaming services are OnLive [110], Gaikai [79], and Otoy [118].

4.6 Cloud Operating Systems

The so-called web desktops which are on offer today are also called cloud operating systems. A popular product from this group is eyeOS [71], an open source solution. The operating system including the installed applications and the user data runs on the provider’s server farms. All the user needs is an end device with Internet access and a standard-compliant browser.

Even though products such as eyeOS are named ‘cloud operating systems’, this term is misleading in most cases. In fact, if users want to use a cloud operating system, they need a computer where a browser and an underlying operating system are installed. This means that the local operating system is in no way replaced, but rather complemented. Only the user applications and data are transferred to the cloud.

Google Chrome OS [86] is another offering from the class of cloud operating systems: A Chrome browser suitable for the efficient use of all Google web applications runs on a boiled-down Linux operating system. Its low resource requirements enable Chrome OS to run even on the smallest netbooks, the so-called nettops.

Chapter 5
Cloud Management

Setting up appropriate management procedures is vital both for operators and for users of cloud services. Services must be described, provisioned, and billed. In order to achieve the required service scalability and reliability, automated processes are employed. When services are transferred from the local context into the public cloud, security issues and risk assessment play an important role. This chapter deals with the related cloud management aspects.

5.1 Service Level Agreements (SLAs)

A *service level agreement* is an agreement between a service provider and a service consumer related to the service level (quality of service). Such an agreement can be reached by signing a formal and legally binding contract, or informally in case of different departments of a company using the services. This is referred to as an *operation level agreement* (OLA). In terms of quality, the SLA implies a mutual agreement with respect to security, priorities, responsibilities, guarantees, and billing modalities. In addition, the SLA specifies metrics such as availability, throughput, response times, and others. By nature, SLAs always consider the output side, i.e. they are drafted from the service consumer's perspective. A provider can stand out by delivering a service in a superior quality or in a particularly innovative way. From the business perspective, it is possible to agree on different quality levels, e.g. Basic, Silver, Gold, Platinum.

Cloud computing SLAs are interesting when it comes to controlling resource allocation and dynamic resource usage. There are two phases which are essential for *service level management*:

- Agreement on the quality of service
- Service monitoring at runtime

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_5,

With the current cloud offerings, agreeing and monitoring specific SLAs is only possible in a very basic way, and these offerings are usually made on a *best effort* basis. In case of failures or service disruptions, the provider issues a corresponding credit note.

With respect to the cloud architecture, developers are called upon to insert a layer into the cloud stack on which both aspects, i.e. service agreement and service monitoring, will be dealt with. SLA@SOI [128] is a project supported by the European Commission in the EU Seventh Framework Programme. It examines the aspects of multi-level SLAs in a market with competing offerings.

5.2 Lifecycle and Automation

Each cloud service goes through a well-defined lifecycle: The service provider defines the scope and quality of the services and describes their properties in a service catalog. The consumers select the desired services from the catalog and instantiate them as required, while SLAs need to be taken into account and monitored. At the end of the utilization period, service orders are closed, service modules are dissociated and resources are reset. An accounting procedure adds up the usage of all resources. Thus, it is possible to track the current status in a fine-grained and time-resolved manner. The consumers are billed either periodically or each time the costs incurred exceed a certain threshold. Most billing procedures use credit cards, while some providers, such as Zimory, also accept direct debiting or invoicing [141].

Services are often provisioned as so-called ensembles. An ensemble is a group of similar resources which are managed in a fully automatic way. This approach facilitates the scalable provisioning of services with a nearly constant management overhead. In this context, automation includes the following steps:

1. Monitoring
2. Analysis
3. Scheduling
4. Execution

A suitable monitoring procedure constantly checks the quality of service and an analysis component examines and evaluates the monitoring component output. In case of malfunctions or deviations from the agreed performance parameters, an appropriate troubleshooting process is selected from a previously defined portfolio (scheduling). The executing unit finally implements the process by providing additional resources, e.g. for an application or a request. The associated components form a closed loop which is cycled through. Process automation is an essential feature of nearly all cloud architectures because it allows for dynamic scalability and fault tolerance.

5.3 Management Services and Tools

For the management of their services, cloud providers offer a broad range of tools (see Table 5.1). Some are command line-based, others can be accessed from Web portals. From the virtually innumerable number of solutions, we selected some exemplary tools for different areas of application which will be presented below.

5.3.1 Monitoring

Periodic acquisition of performance data is important both to the cloud provider and to the cloud consumer who wants to judge the quality of service. CloudClimate, for example, gathers performance data of different cloud service providers and publishes them on a website [63], including performance metrics such as CPU, memory usage and hard disk access. These values are displayed in charts for the past month. This not only enables the users to directly compare the performance of different providers, but in addition, malfunctions and periods with unusually high loads can be identified.

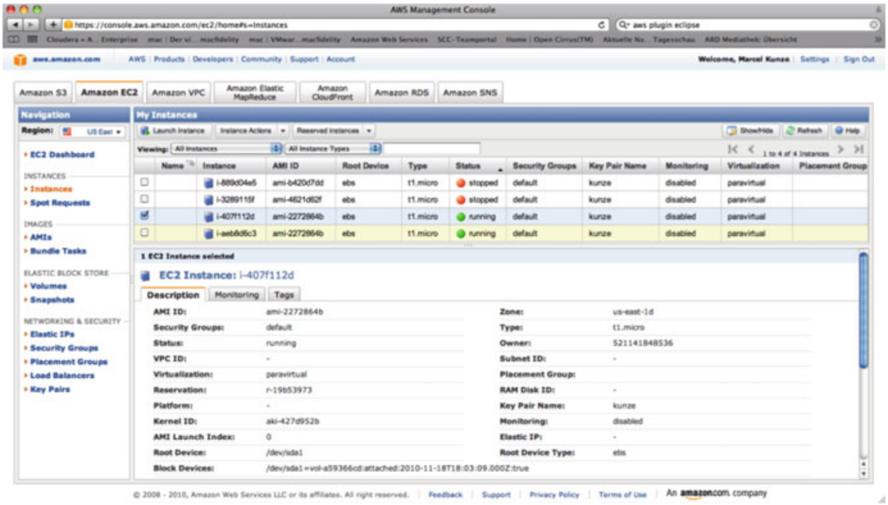
Amazon CloudWatch is a special Web service that monitors the Amazon Web Services [37] performance. CloudWatch allows to view resource usage and displays the current performance data as well as access patterns. This service collects data on CPU usage, disk access, and network traffic. To enable CloudWatch, the user allocates this service to an EC2 instance. The resulting monitoring data can be processed using either the Web service API or command line procedures.

5.3.2 Control

To support the management of its infrastructure services, Amazon not only offers a set of command line tools and libraries for various programming languages [51], but also a convenient Web-based management console [53] (see Fig. 5.1).

With this console, the following services can be controlled and monitored: Amazon S3, Amazon EC2, Amazon Virtual Private Cloud, Amazon Elastic MapReduce, Amazon CloudFront, Amazon Relational Database Service, and Amazon Simple Notification Service. This includes the following EC2 operations:

- **Instances:** Start, stop, restart, remove, access the console, view log files
- **Images:** Start, register, delete, define access rights
- **Memory:** Create, delete, allocate, release, take snapshots
- **Network:** Manage IP addresses
- **Structure:** Manage placement groups and load distribution
- **Security:** Manage security groups and keys



	Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name	Monitoring	Virtualization	Placement Group
□		i-889d04e5	ami-b420d7dd	ebs	t1.micro	● stopped	default	kunze	disabled	paravirtual	
□		i-3289115f	ami-4621d62f	ebs	t1.micro	● stopped	default	kunze	disabled	paravirtual	
☑		i-407f112d	ami-2272864b	ebs	t1.micro	● running	default	kunze	disabled	paravirtual	
□		i-aeb8d6c3	ami-2272864b	ebs	t1.micro	● running	default	kunze	disabled	paravirtual	

1 EC2 Instance selected

EC2 Instance: i-407f112d

AMI ID:	ami-2272864b	Zone:	us-east-1d
Security Groups:	default	Type:	t1.micro
Status:	running	Owner:	521141848536
VPC ID:	-	Subnet ID:	-
Virtualization:	paravirtual	Placement Group:	
Reservation:	r-19b53973	RAM Disk ID:	-
Platform:	-	Key Pair Name:	kunze
Kernel ID:	aki-427d952b	Monitoring:	disabled
AMI Launch Index:	0	Elastic IP:	-
Root Device:	/dev/sda1	Root Device Type:	ebs
Block Devices:	/dev/sda1=vol-a59366cd:attached:2010-11-18T18:03:09.000Z:true		

Fig. 5.1 Managing infrastructure with the AWS console

These management operations can be performed separately for the different availability zones. Besides the official EC2 command line tools from Amazon, there are a number of freely available tools that can be used to interact with AWS-compatible cloud services, such as s3cmd [121] and the Euca2ools from Eucalyptus.

An alternative graphical console is available as an open source plug-in for the Firefox browser: *ElasticFox* organizes infrastructure management in a similar way [72]. An advantage of ElasticFox is that it allows to manage not only EC2-based public clouds, but also Eucalyptus-based private clouds. A screenshot of the ElasticFox console is shown in Fig. 5.2.

There are multiple solutions which are targeted at using the Amazon S3 storage service. The Firefox S3Fox Organizer plug-in is of particular interest [122].

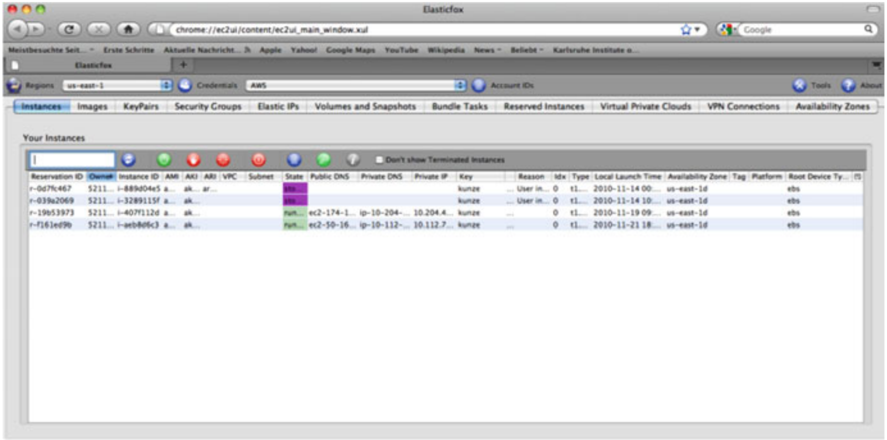
With this tool, a customer can

- Upload, download, delete, or hierarchically organize files,
- Control the visibility of data on the Internet (also temporarily),
- Manage access rights (especially set up access control lists), and
- Automatically synchronize local folders with S3.

The user interface is similar to a traditional FTP client (see Fig. 5.3). It contains two views showing the local and the cloud file structures side-by-side; files can be copied by simple drag-and-drop.

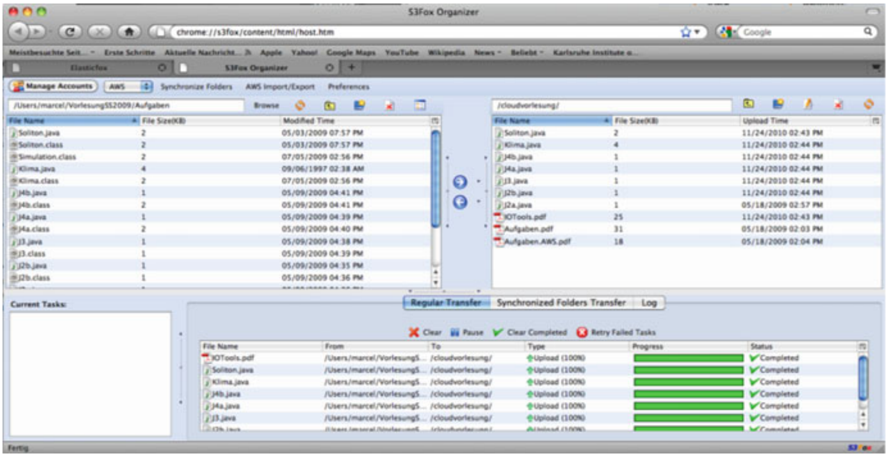
The existing tools for working with cloud services belong to one of the following three basic categories:

- Online tools (services)
- Browser plug-ins
- Command line tools



Reservation ID	Owner	Instance ID	AMI	AKI	ARI	VPC	Subnet	State	Public DNS	Private DNS	Private IP	Key		Reason	Idx	Type	Local Launch Time	Availability Zone	Tag	Platform	Root Device Ty...
r-0d7fc467	5211...	i-889d04e5	a...	ak...	ar...			sto...				kunze	...	User in...	0	t1....	2010-11-14 00:...	us-east-1d			ebs
r-039a2069	5211...	i-3289115f	a...	ak...				sto...				kunze	...	User in...	0	t1....	2010-11-14 10:...	us-east-1d			ebs
r-19b53973	5211...	i-407f112d	a...	ak...				run...	ec2-174-1...	ip-10-204-...	10.204.4...	kunze	...		0	t1....	2010-11-19 09:...	us-east-1d			ebs
r-f161ed9b	5211...	i-aeb8d6c3	a...	ak...				run...	ec2-50-16...	ip-10-112-...	10.112.7...	kunze	...		0	t1....	2010-11-21 18:...	us-east-1d			ebs

Fig. 5.2 Infrastructure management with ElasticFox


S3Fox Organizer
chrome://s3fox/content/html/host.htm
Google
Meistbesuchte Seit... Erste Schritte Aktuelle Nachricht... Apple Yahoo! Google Maps YouTube Wikipedia News Beliebt Karlsruhe Institute o...
Elasticfox
S3Fox Organizer
Manage Accounts
AWS
Synchronize Folders
AWS Import/Export
Preferences
Browse
Current Tasks:
Regular Transfer
Synchronized Folders Transfer
Log
Clear
Pause
Clear Completed
Retry Failed Tasks
Fertig

/Users/marcel/VorlesungSS2009/Aufgaben

File Name	File Size(KB)	Modified Time
Soliton.java	2	05/03/2009 07:57 PM
Soliton.class	2	05/03/2009 07:57 PM
Simulation.class	2	07/05/2009 02:56 PM
Klima.java	4	09/06/1997 02:38 AM
Klima.class	2	07/05/2009 02:56 PM
J4b.java	1	05/09/2009 04:41 PM
J4b.class	2	05/09/2009 04:41 PM
J4a.java	1	05/09/2009 04:39 PM
J4a.class	2	05/09/2009 04:40 PM
J3.java	1	05/09/2009 04:38 PM
J3.class	1	05/09/2009 04:39 PM
J2b.java	1	05/09/2009 04:35 PM
J2b.class	1	05/09/2009 04:36 PM

/cloudvorlesung/

File Name	File Size(KB)	Upload Time
Soliton.java	2	11/24/2010 02:43 PM
Klima.java	4	11/24/2010 02:44 PM
J4b.java	1	11/24/2010 02:44 PM
J4a.java	1	11/24/2010 02:44 PM
J3.java	1	11/24/2010 02:44 PM
J2b.java	1	11/24/2010 02:44 PM
J2a.java	1	05/18/2009 02:57 PM
IOTools.pdf	25	11/24/2010 02:43 PM
Aufgaben.pdf	31	05/18/2009 02:03 PM
Aufgaben.AWS.pdf	18	05/18/2009 02:04 PM

File Name	From	To	Type	Progress	Status
IOTools.pdf	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed
Soliton.java	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed
Klima.java	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed
J4b.java	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed
J4a.java	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed
J3.java	/Users/marcel/VorlesungS...	/cloudvorlesung/	Upload (100%)		✓Completed

Fig. 5.3 S3Fox organizer for storage services

The online tools, which are typically implemented as services, include tools such as the AWS Management Console [53] and a corresponding offering from Ylastic [139]. A drawback of these solutions is that customers must trust the tool provider with respect to data privacy and data security because the access data will always be stored with the provider. In addition, the AWS Management Console will only work with services that are part of Amazon Web Services. This means that private cloud services are excluded. Currently, Ylastic only supports Amazon Web Services and infrastructure services based on Eucalyptus.

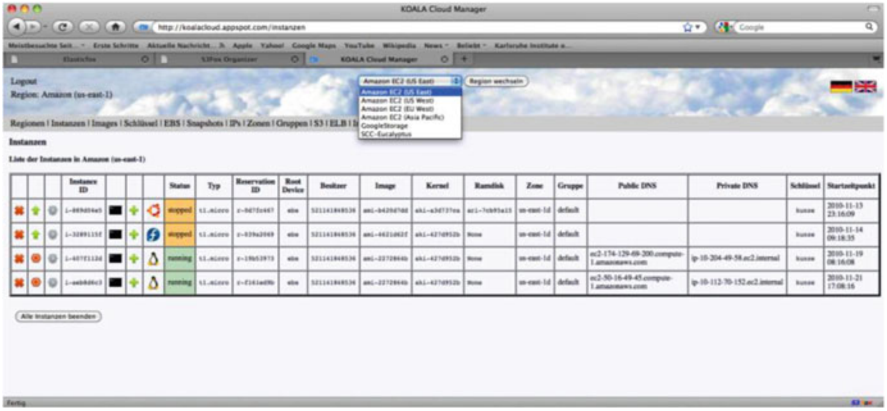
While browser plug-ins, such as ElasticFox and Hybridfox, boast many functions, they require a local installation and thus involve a certain maintenance effort. Each time a service provider switches to a new version, there is a certain risk of interface incompatibilities. What is more, the plug-ins cannot be used with alternative products, such as Chrome, Opera, Safari, Internet Explorer, etc.

Command line tools, such as the EC2 API Tools from Amazon and the Euca2ools from Eucalyptus, also require local installation. Since they have no graphical interface, they are less user-friendly. Just like the other Amazon tools, the EC2 API Tools only work with the AWS public cloud services. The Euca2ools are able to interact with public and private cloud services that are compatible with Amazon Web Services.

KOALA [101, 102] is a new solution for controlling cloud services, which overcomes the limitations of the existing tools. It is a software service capable of controlling public and private cloud infrastructures compatible with AWS interfaces (see Fig. 5.4). Cloud services from Amazon, Eucalyptus, Nimbus, and OpenNebula as well as the management of Google Storage are supported. The service itself can be operated in Google App Engine or alternatively in a compatible private cloud (AppScale and TyphoonAE). KOALA has been published under an open source license (Table 5.1).

Table 5.1 Cloud management tools

Name	Provider	Design	License	Costs	EC2	S3	EBS	ELB	Requirements
KOALA [101]	KIT	SaaS	Apache v2.0	Free	Yes	Yes	Yes	Yes	Browser
AWS console [53]	Amazon	SaaS	proprietary	Free	Yes	Yes	Yes	Yes	Browser
Ylastic [139]	Ylastic	SaaS	proprietary	$25/month	Yes	Yes	Yes	Yes	Browser
ElasticFox [72]	Amazon	Plug-in	Apache v2.0	Free	Yes	No	Yes	No	Firefox
S3Fox [122]	Suchi	Plug-in	proprietary	Free	No	Yes	No	No	Firefox
Euca2ools [74]	Eucalyptus	Shell	BSD	Free	Yes	No	Yes	No	Installation
API tools [50]	Amazon	Shell	Apache v2.0	Free	Yes	No	Yes	Yes	Installation
GSUtil [94]	Google	Shell	Apache v2.0	Free	No	Yes	No	No	Installation
s3cmd [121]	M. Ludvig	Shell	GPLv2	Free	No	Yes	No	No	Installation
AWS toolkit [54]	Amazon	Eclipse	Apache v2.0	Free	Yes	No	No	No	Eclipse

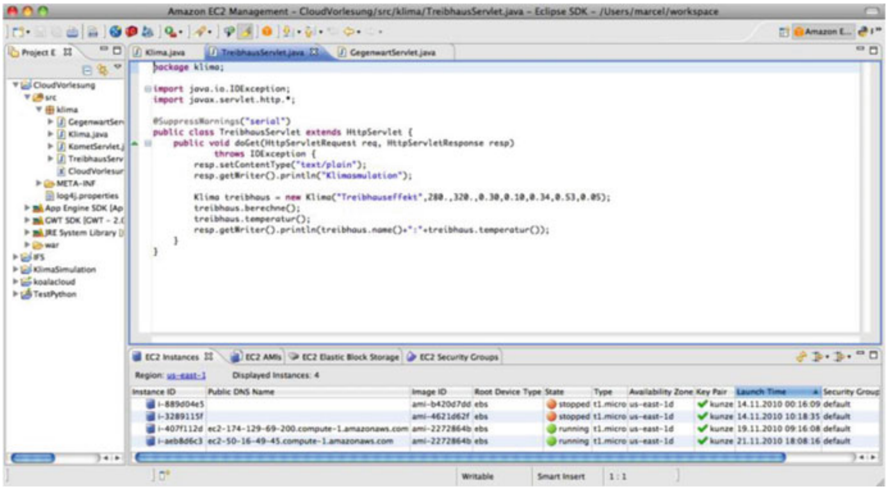

KOALA Cloud Manager
http://koalacloud.appspot.com/instanzen
Google
Meistbesuchte Seit... Erste Schritte Aktuelle Nachricht... Apple Yahoo! Google Maps YouTube Wikipedia News Beliebt Karlsruhe Institute o...
ElasticFox
S3Fox Organizer
KOALA Cloud Manager
Logout
Region: Amazon (us-east-1)
Amazon EC2 (US East)
Amazon EC2 (US East)
Amazon EC2 (US West)
Amazon EC2 (EU West)
Amazon EC2 (Asia Pacific)
GoogleStorage
SCC-Eucalyptus
Region wechseln
Regionen | Instanzen | Images | Schlüssel | EBS | Snapshots | IPs | Zonen | Gruppen | S3 | ELB | I
Alle Instanzen beenden
Fertig

Instanzen

Liste der Instanzen in Amazon (us-east-1)

			Instance ID				Status	Typ	Reservation ID	Root Device	Besitzer	Image	Kernel	Ramdisk	Zone	Gruppe	Public DNS	Private DNS	Schlüssel	Startzeitpunkt
✖	▲	○	i-889d04e5	■	+		stopped	t1.micro	r-8d7fc467	ebs	521141848536	ami-b429d7d8	aki-a3d737ca	ari-7cb95a15	us-east-1d	default			kunze	2010-11-13 23:16:09
✖	▲	○	i-32891151	■	+		stopped	t1.micro	r-039a2069	ebs	521141848536	ami-4621d62f	aki-427d952b	None	us-east-1d	default			kunze	2010-11-14 09:18:35
✖	●	○	i-407f113d	■	+		running	t1.micro	r-19b53973	ebs	521141848536	ami-2272864b	aki-427d952b	None	us-east-1d	default	ec2-174-129-69-200.compute-1.amazonaws.com	ip-10-204-49-58.ec2.internal	kunze	2010-11-19 08:16:08
✖	●	○	i-aeb8d6c3	■	+		running	t1.micro	r-f161ad9b	ebs	521141848536	ami-2272864b	aki-427d952b	None	us-east-1d	default	ec2-50-16-49-45.compute-1.amazonaws.com	ip-10-112-70-152.ec2.internal	kunze	2010-11-21 17:08:16

Fig. 5.4 KOALA cloud manager



```
package klima;

import java.io.IOException;
import javax.servlet.http.*;

@SuppressWarnings("serial")
public class TreibhausServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Klimasmulation");

        Klima treibhaus = new Klima("Treibhauseffekt",280.,320.,0.30,0.10,0.34,0.53,0.05);
        treibhaus.berechne();
        treibhaus.temperatur();
        resp.getWriter().println(treibhaus.name()+":"+treibhaus.temperatur());
    }
}
```

Region: us-east-1 Displayed Instances: 4

Instance ID	Public DNS Name	Image ID	Root Device Type	State	Type	Availability Zone	Key Pair	Launch Time	Security Group
i-889d04e5		ami-b420d7dd	ebs	● stopped	t1.micro	us-east-1d	✓ kunze	14.11.2010 00:16:09	default
i-3289115f		ami-4621d62f	ebs	● stopped	t1.micro	us-east-1d	✓ kunze	14.11.2010 10:18:35	default
i-407f112d	ec2-174-129-69-200.compute-1.amazonaws.com	ami-2272864b	ebs	● running	t1.micro	us-east-1d	✓ kunze	19.11.2010 09:16:08	default
i-aeb8d6c3	ec2-50-16-49-45.compute-1.amazonaws.com	ami-2272864b	ebs	● running	t1.micro	us-east-1d	✓ kunze	21.11.2010 18:08:16	default

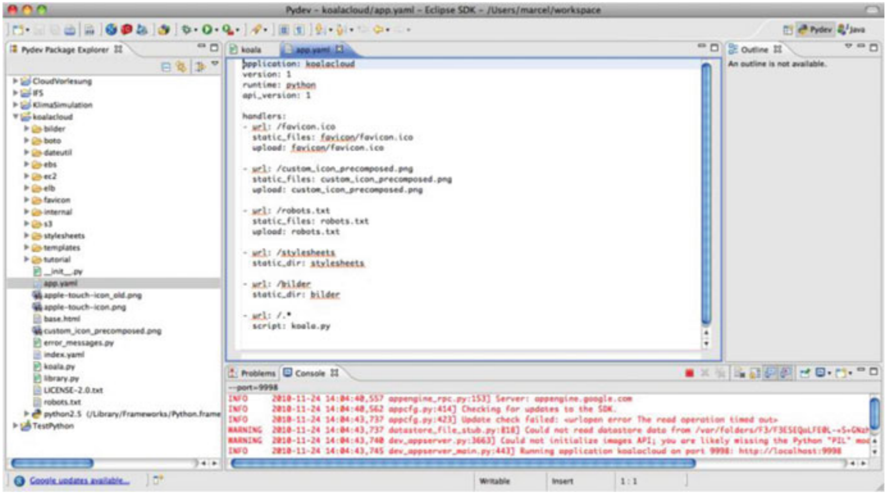
Fig. 5.5 Amazon web service plug-in for eclipse

5.3.3 Development

In a mainly service-oriented landscape, operation and development are often closely intertwined, so that besides the management tools, development tools play an important role. Let us take Eclipse [70] as an example of an integrated cloud development and management environment. Eclipse is a very popular application development platform for which a wealth of plug-ins exist, accommodating the most diverse programming environments. The gEclipse project, for example, features a comprehensive graphical interface for users, developers, and operators of grid and cloud infrastructures [81]. There are also specific Amazon Web Services extensions, such as the AWS Eclipse ToolKit [54] which enables developers to develop and test distributed and scalable Java applications based on Tomcat containers for Amazon EC2 (see Fig. 5.5). Tools for managing security groups, AMI libraries, EC2 instances, and EBS memory are included as well.

With its Google Web Toolkit (GWT), Google provides another way to support cloud application development. Google Web Toolkit is an SDK for the development of Java or Python programs to be run on the App Engine platform [84]. Besides the GWT, this development environment also includes a local Web server for testing the programs. Once developers are happy with the way an application works, they can package it into a so-called WAR file for upload to the scalable Google infrastructure. As an extension of the SDK, there is also a Google plug-in for Eclipse targeted at the interactive development of GWT applications in a graphical, integrated development environment [91].

With a simple click on the App Engine button, locally developed programs can be published to the Google infrastructure (see Fig. 5.6).



```
application: koalacloud
version: 1
runtime: python
api_version: 1

handlers:
- url: /favicon.ico
  static_files: favicon/favicon.ico
  upload: favicon/favicon.ico

- url: /custom_icon_precomposed.png
  static_files: custom_icon_precomposed.png
  upload: custom_icon_precomposed.png

- url: /robots.txt
  static_files: robots.txt
  upload: robots.txt

- url: /stylesheets
  static_dir: stylesheets

- url: /bilder
  static_dir: bilder

- url: /.*
  script: koala.py
```

```
--port=9998
INFO     2010-11-24 14:04:40,557 appengine_rpc.py:153] Server: appengine.google.com
INFO     2010-11-24 14:04:40,562 appcfg.py:414] Checking for updates to the SDK.
INFO     2010-11-24 14:04:43,737 appcfg.py:423] Update check failed: <urlopen error The read operation timed out>
WARNING  2010-11-24 14:04:43,737 datastore_file_stub.py:818] Could not read datastore data from /var/folders/F3/F3ESEQoLFE0L-+S+GNzH
WARNING  2010-11-24 14:04:43,740 dev_appserver.py:3663] Could not initialize images API; you are likely missing the Python "PIL" mod
INFO     2010-11-24 14:04:43,745 dev_appserver_main.py:443] Running application koalacloud on port 9998: http://localhost:9998
```

Fig. 5.6 Google app engine plug-in for eclipse

5.4 Security Management

Security is not only related to safely accessing resources, but also covers data privacy issues. A study by Berkeley University researchers shows that, for the so-called *cloud sourcing*, no specific challenges or problems exist that would require measures other than those currently implemented [1]. With respect to security, the same safety objectives apply as those common to the operation of services in a local data center [28]:

- Confidentiality
- Integrity
- Availability
- Authenticity
- Accountability
- Pseudonymity

Compliance with these safety objectives has to be considered as an integral part of the SLA negotiated between the service provider and the service consumer. The Zimory cloud marketplace, for example, allows to define SLAs with customized security requirements for cloud services [141].

Due to the openness of this approach, the various categories of cloud services have different characteristics:

- IaaS: Highest flexibility, the customer is responsible for security
- PaaS: Medium flexibility, both the customer and the provider are responsible for security
- SaaS: Lowest flexibility, the provider is responsible for security

Many providers simply identify customers by their credit card number.

This also ensures a smooth billing process. Keys or PKI-based processes usually allow controlled resource access and thus prevent unauthorized use. To guarantee the desired confidentiality, data encryption is used for transmission and storage. In this context, it is worth noting that storing encrypted data in the cloud may in some cases be safer than storing unencrypted data on a local PC or in a company data center. In addition, some providers use auditing processes to record all activities related to the use of their resources. This kind of monitoring enables the providers to comply even with the strictest legal provisions.

To guarantee information security, a certification according to ISO/IEC 27001 – or SAS 70 in the U.S. – is of great importance. Here, the processes required to establish and monitor security are stipulated in a binding manner. High-profile providers, such as Amazon or Microsoft, have undergone this certification for their resource centers and can thus offer a higher security standard than smaller company data centers.

There are scenarios where outsourcing of tasks into the cloud is indeed advantageous with respect to security issues: For a company intending to collaborate with external partners in a common project, the firewalls of the partner companies often become almost insurmountable obstacles to establishing common processes. If these processes, however, are transferred, e.g. to the Amazon cloud, it is possible for the cloud provider to autonomously adjust the firewall access rules in such a way that all project partners can collaborate smoothly and without compromising the security guidelines of the participating companies.

For more information on this topic, we refer to the comprehensive compendium on cloud security published by the *Cloud Security Alliance* [66].

5.5 Risk Management

With respect to risk assessment, cloud sourcing is hardly different from the classical "cloud-less" situation: When data has been transferred to the cloud, there is always a risk that access might no longer be available in case of a service disruption or bankruptcy of the provider. These issues, however, are basically the same as with any outsourcing process and can be handled by concluding suitable contracts or SLAs. To mitigate this problem, it is possible to call upon a second, independent cloud service provider as a backup solution to store copies of all business-critical data.

A further risk is the dependency on the proprietary technology and interfaces of a certain provider (vendor lock-in): This dependency problem is a minor one with IaaS and a major one with PaaS and SaaS: When developing a service, e.g. on top of Google App Engine, the developer is locked to the Google-specific infrastructure, and a platform change can be rather complex, time-consuming, and costly. In this context, it should be noted that similar dependencies on software manufactures, platforms, and infrastructures also exist when services are operated in a local data center. Conventional knowledge will therefore help to solve this problem:

- If possible, standardized processes should have priority over a proprietary solution.
- Software should be developed in a way to ensure the highest possible independence from a single platform.
- If dependencies are inevitable, proper encapsulation should be ensured for maximum flexibility.

For an evaluation of the opportunities and risks of cloud computing, see Chap. 8.

5.6 Legal Compliance

Neither cloud service offerings nor their use may infringe laws, social values, morals, or ethics. This is ensured by the concept of *compliance*. According to established case-law, the same laws are applicable to cloud computing as to renting (in case of paid services) or lending (with respect to unpaid services).

The geographical location of the cloud provider is decisive in the determination of the laws that will apply to the data stored. The data may have been replicated so that they exist in different places in the world at the same time and therefore might be subject to different data privacy laws: According to article 4 of the European Data Protection Directive, the pertaining national data protection legislation of the respective country is applicable [138]. For this reason, Amazon, for example, offers cloud services for different regions, such as the U.S., Europe, or Asia. Thus, it is possible, to specify the appropriate judicial area for the services by selecting the desired zone.

In this context, the legal aspects related to personal data processing deserve extra attention. Country-specific laws cover this topic in each country, and for the EU, directive 2000/31/EC is applicable. If data processing takes place outside of Europe, the customer should make sure that an appropriate data protection level exists (e.g. Safe Harbor Agreement in the U.S.). If international resources are involved in the commissioned processing of personal data in the cloud, the consent of the person concerned must always be obtained. From a legal point of view, this kind of data processing is still permitted, though, as soon as the personal character of the data is removed by measures such as encryption or anonymization.

For industries such as healthcare and finance explicit regulations regarding data protection exist such as the Health Insurance Portability and Accountability Act (HIPPA) or the Payment Card Industry Data Security Standard (PCI DSS) [130].

Chapter 6
Open Source Cloud Stack

This chapter explains how to build a cloud system based on open source components. A number of solutions suitable for creating a cloud architecture are already available. Thus, it is possible to design an open source cloud stack, as shown in Fig. 6.1.

The design not only includes components for building the hardware and software infrastructure, but also components for establishing application environments. The stack shown here is a specific variant of the cloud architecture represented in Chap. 3: The IaaS components introduced there are represented by the physical resource sets (PRS) which are used to partition the infrastructure, while the software infrastructure includes virtual machine and memory management components as well as procedures for monitoring and controlling the infrastructure. In addition, this layer in the stack comprises job control and accounting and billing components. The PaaS components form the framework layer, the SaaS components can be found on the application layer. In the following sections, we will present some of the design components by way of example.

6.1 Physical and Virtual Resources

The roots of the IaaS components lie in the Emulab project [16] where miniature data centers are made available for system development purposes. On the bottom layer, the infrastructure is organized in the form of physical resource sets (PRS). Each PRS comprises the resources (e.g. CPU, memory, networks) required for the implementation of a task. They are linked by a virtual LAN in a common domain. For domain management, a special PRS service is used which manages the resources over the network and allows to switch components on or off, roll out system images, and monitor the infrastructure. Figure 6.2 shows an example with four different domains:

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_6,

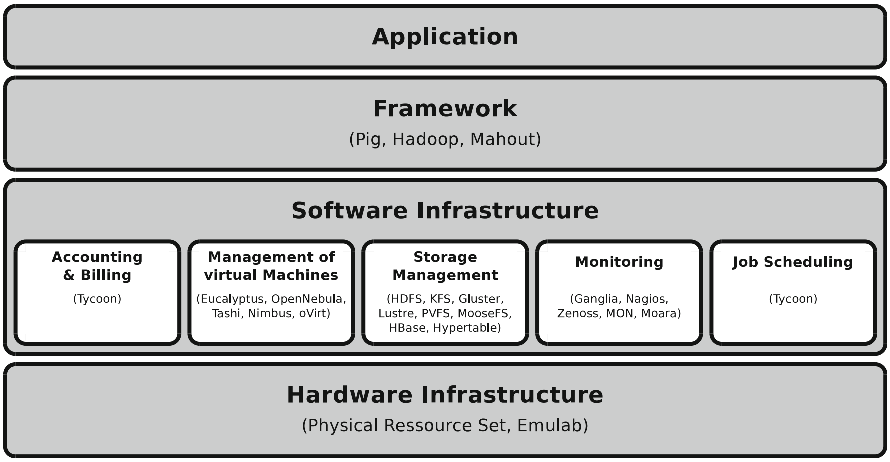

Application
Framework
(Pig, Hadoop, Mahout)
Software Infrastructure
Accounting & Billing
(Tycoon)
Management of virtual Machines
(Eucalyptus, OpenNebula, Tashi, Nimbus, oVirt)
Storage Management
(HDFS, KFS, Gluster, Lustre, PVFS, MooseFS, HBase, Hypertable)
Monitoring
(Ganglia, Nagios, Zenoss, MON, Moara)
Job Scheduling
(Tycoon)
Hardware Infrastructure
(Physical Ressource Set, Emulab)

Fig. 6.1 OpenCirrus open source cloud stack

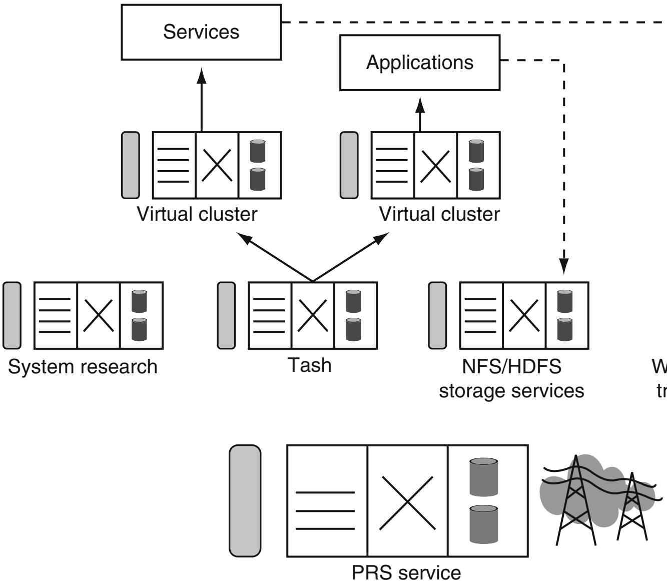


Fig. 6.2 Different domains associated with a physical resource set

Here, we find one domain for systems research, one for managing virtual machines, and further domains for storage services and monitoring. These services are consumed by applications which run on virtual clusters of the second domain.

The virtual clusters form virtual resource sets (VRS). In the example of Fig. 6.2, this is the Tashi management suite which is being developed jointly by Intel and Yahoo!. Tashi is a solution especially targeted at cloud data centers which have to process enormous data sets on the Internet. The seminal idea is that not only the CPU resources are subject to a scheduling process, but also the distributed storage resources. By scheduling CPU and data storage systems together, it is possible to optimize the system performance while keeping an eye on energy consumption [129].

Another very popular virtual resource management system is Eucalyptus which will be described below.

6.2 Eucalyptus

Cloud infrastructures from commercial providers, such as Amazon EC2, and S3, and PaaS offerings, such as Google App Engine, boast a high degree of usability and can be used at low cost (or even for free). In some cases, however, it is desirable to build a private cloud infrastructure. Situations where a private cloud would be preferred over a public cloud might be characterized by special security requirements or the need to store critical company data. It is also conceivable to set up an internal data mirror (RAID-0) in order to increase the availability of a commercial provider's cloud infrastructure.

Eucalyptus [74] is short for *Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*, and it was initially developed at the University of California in Santa Barbara (UCSB). Eucalyptus Systems has taken over the activities for further development of this system. Eucalyptus allows to set up and operate an independent IaaS cloud infrastructure. The Eucalyptus API is compatible with Amazon EC2, S3, and EBS [22]. For software development, a BSD license is used, i.e. it is open-sourced. Unlike Amazon EC2, which exclusively uses Xen for virtualization, Eucalyptus can co-operate with Xen und KVM (Kernel-based Virtual Machine). A prerequisite for using KVM is a CPU that supports hardware virtualization, i.e. AMD-V (Pacifica) or Intel VT-x (Vanderpool). The commercially available Enterprise Version offered by Eucalyptus Systems supports VMware vSphere/ESX/ESXi. It is not planned to integrate VMware support into the free Eucalyptus version.

6.2.1 Architecture and Components

As shown in Fig. 6.3, the Eucalyptus infrastructure consists of three components: the Cloud Controller (CLC), the Cluster Controller (CC), and the Node Controller (NC) [23]. All three components are implemented as Web services.

The NC must be installed on every node in the cloud where virtual instances should run. This, however, requires a functional Xen Hypervisor or KVM. Each NC sends information on the current state of its own resources to the CC, i.e. the number of virtual processors, free RAM and free disk space.

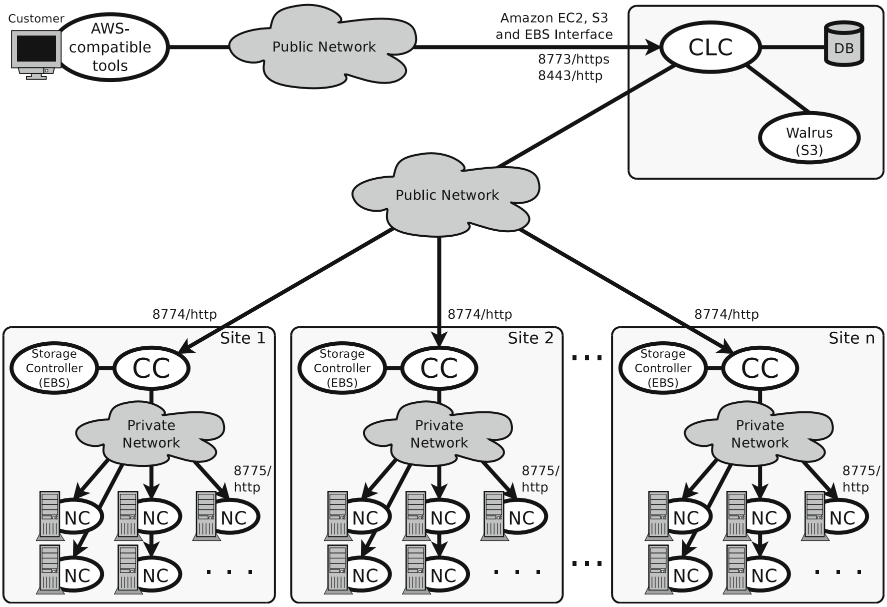


Fig. 6.3 Eucalyptus architecture and components

In each cluster, a CC performs the load-dependent distribution (i.e. scheduling) of the virtual machines to the NCs. For this purpose, the CC uses the resource information it receives from the NCs. Another task of a CC is to control the private network it uses to communicate with the NCs. Each CC sends information on the current state of the resources in its own cluster.

The CLC is responsible for meta-scheduling, i.e. the way how virtual machines are distributed between the connected clusters. For this purpose, it collects resource information submitted by the CCs. In each Eucalyptus infrastructure, exactly one CLC must be active. The CLC is the access point in the cloud, both for users and for administrators.

In Eucalyptus infrastructures with a small number of physical servers, it is a good idea to consolidate the CLC and the CC on a single server. If need be, all three components may be operated together on a physical server.

Eucalyptus further includes two storage services: Walrus is a storage service for Web objects compatible with the Amazon S3 REST API. In addition, there is a storage service called Storage Controller (SC) whose functionality and API are identical to the Amazon EBS service. Walrus and SC can be run on any computer in the cluster. In small and medium installations, Walrus and SC are usually located on the CLC.

Eucalyptus uses Walrus to store the images. But it is also possible to install and use Walrus as a standalone service, independently from Eucalyptus.

Virtual Distributed Ethernet (VDE) is used to create the private network. For this purpose, virtual VDE switches run on the individual Eucalyptus components. The switches are linked by virtual VDE cables. The virtual network ensures that a homogeneous subnet is available to the virtual machines within a cloud cluster [3].

In a Eucalyptus private cloud, the instance class identifiers are the same as with Amazon EC2. They differ, however, by the resources which are allocated to them by default, as shown in Tables 6.1 and 6.2. While the resource allocation for the respective instance classes within a Eucalyptus private cloud can be configured, there is no way to define additional or rename existing instance classes.

The *new* Amazon Web Services instance classes (which include the `m2.xlarge`, `m2.2xlarge`, and `m2.4xlarge` high-memory instances and the `t1.micro` micro instances) have not yet been introduced in Eucalyptus.

In a Eucalyptus cloud, all instance classes can be placed on all nodes. Thus, it is not possible to differentiate the instance classes by architectures as with Amazon EC2. In EC2, the `m1.small` and `c1.medium` instance classes are exclusively available to instances with a 32-bit architecture, while all other instances are based on the 64-bit architecture. An exception to this is the `t1.micro` instance class which can be used for 32-bit and for 64-bit instances. In a Eucalyptus cloud, in contrast, all instance classes have the same architecture.

Table 6.1 Computing power comparison of the Eucalyptus and Amazon EC2 instance classes

Category	Eucalyptus	Amazon EC2
`t1.micro`	n/a	1 virtual core with 2 ECUs max.
`m1.small`	1 virtual CPU	1 virtual core with 1 ECU
`m1.large`	2 virtual CPUs	2 virtual cores with 2 ECUs each ⇨ 4 ECUs
`m1.xlarge`	2 virtual CPUs	4 virtual cores with 2 ECUs each ⇨ 8 ECUs
`m2.xlarge`	n/a	2 virtual cores with 3.25 ECUs each ⇨ 6.5 ECUs
`m2.2xlarge`	n/a	4 virtual cores with 3.25 ECUs each ⇨ 13 ECUs
`m2.4xlarge`	n/a	8 virtual cores with 3.25 ECUs each ⇨ 26 ECUs
`c1.medium`	1 virtual CPU	2 virtual cores with 2.5 ECUs each ⇨ 5 ECUs
`c1.xlarge`	4 virtual CPUs	8 virtual cores with 2.5 ECUs each ⇨ 20 ECUs
`cc1.4xlarge`	n/a	8 Intel Xeon Nehalem cores ⇨ 33.5 ECUs
`cg1.4xlarge`	n/a	8 Intel Xeon Nehalem cores ⇨ 33.5 ECUs

Table 6.2 RAM comparison of the Eucalyptus and Amazon EC2 instance classes

Category	Eucalyptus	Amazon EC2
`t1.micro`	–	613 MB RAM
`m1.small`	128 MB RAM	1.7 GB RAM
`m1.large`	512 MB RAM	7.5 GB RAM
`m1.xlarge`	1 GB RAM	15 GB RAM
`m2.xlarge`	–	17.1 GB RAM
`m2.2xlarge`	–	34.2 GB RAM
`m2.4xlarge`	–	68.4 GB RAM
`c1.medium`	256 MB RAM	1.7 GB RAM
`c1.xlarge`	2 GB RAM	7 GB RAM
`cc1.4xlarge`	–	23 GB RAM
`cg1.4xlarge`	–	22 GB RAM

A further difference between Amazon Web Services and Eucalyptus lies in the performance of the CPU cores offered. For the definition of its computing power, Amazon uses the *EC2 Compute Units* (ECU) metric. With respect to computing power, an EC2 Compute Unit is equivalent to a 1.0–1.2 GHz Opteron or Xeon processor from 2007 or a 1.7 GHz Xeon processor from spring 2006 [42].

The virtual CPU cores in Amazon Web Services feature different performances which depend on the instance class they are assigned to. The reason is that Amazon has discrete physical hardware available for provisioning in the different instance classes. While in the EC2 `m1.small` instance class, the computing power of a virtual CPU core corresponds to one EC2 compute unit, each virtual core has a computing power of two EC2 compute units in the other standard instances, i.e. `m1.small` and `m1.large`. In the `m2.xlarge`, `m2.2xlarge`, and `m2.4xlarge` high-memory instances, each virtual core has a computing power of 3.25 EC2 compute units, and in the `c1.medium` and `c1.xlarge` high CPU instances, a computing power of 2.5 EC2 compute units. The cluster compute instances are equipped with two quad-core Intel Xeon-X5570 Nehalem processors each.

6.3 OpenNebula

Just like Eucalyptus, OpenNebula [114] is an IaaS solution for building private clouds. OpenNebula supports the Xen Hypervisor, KVM, and VMware vSphere virtualization approaches. Unlike Eucalyptus, OpenNebula allows to move running instances between the connected nodes. To date, however, OpenNebula only provides basic support for the EC2 SOAP and EC2 Query APIs. It is possible to retrieve a list of images and instances and to start, restart, and stop instances. In addition, OpenNebula can be used to control Amazon EC2 resources.

A cutting-edge feature of OpenNebula is its node grouping capability, so that it enables High Performance Computing as a Service (HPCaaS).

Contrary to Eucalyptus and Nimbus, OpenNebula does not include a storage service which is compatible with the S3 or EBS API. OpenNebula is available under an open source license.

6.4 Nimbus

Nimbus [17] is a private cloud IaaS solution developed by the Globus Alliance. Nimbus supports the Xen Hypervisor and KVM virtualization solutions. For virtual machine scheduling, Nimbus can rely on systems such as Portable Batch System (PBS) or Sun Grid Engine (SGE). Nimbus features basic support for the EC2 SOAP and EC2 Query APIs that allows users to retrieve a list of images and instances. It is possible to start, restart, and stop instances and to create key pairs. Amazon Web Services resources can be addressed via a back-end.

Version 2.5 and higher of Nimbus include the Cumulus storage service whose interface is compatible with the S3 REST API. Nimbus uses Cumulus to store the images. Cumulus may be installed and operated as a standalone service without Nimbus. Nimbus does not feature an EBS-compatible storage service. It is available under an open source license.

6.5 CloudStack

Another private cloud IaaS solution is CloudStack [65], jointly developed by Cloud.com and Rackspace. CloudStack supports the Xen Hypervisor, KVM, and VMware vSphere virtualization approaches. The architecture comprises two components: the Management Server and the Compute Nodes. The Management Server features a Web user interface for administrators and users. Other Management Server tasks are to control and manage the resources when distributing the instances to the Compute Nodes.

This software is available in a Community Edition, an Enterprise Edition, and a Service Provider Edition. Only the Community Edition can be used under an open source license.

Table 6.3 summarizes the most popular open source solutions for the implementation of virtual resource sets.

6.6 OpenStack

In summer 2010, NASA and Rackspace jointly launched an open source project called OpenStack [117]. Many renowned companies support this project, such as AMD, Intel, Dell, and Cloud.com. On the basis of CloudStack, OpenStack provides the *Compute* and *Object Storage* components. The Compute service allows to

Table 6.3 Comparison of open source virtual resource sets

Name	License	Interface	EC2	S3	EBS	Hypervisor	Enterprise
Eucalyptus	GPL v3	AWS	Yes	Yes	Yes	KVM, Xen, VMware	Yes
OpenNebula	Apache v2.0	OCCI, AWS	Partially	No	No	KVM, Xen, VMware, VirtualBox	No
Nimbus	Apache v2.0	WSRF, AWS	Partially	Partially	No	KVM, Xen	No
CloudStack	GPL v3	CloudStack, AWS	Partially	No	No	KVM, Xen, VMware	Yes
OpenStack	Apache v2.0	OpenStack, AWS	Yes	No	No	KVM, Xen, VirtualBox, UML	No

manage large groups of virtual servers, Object Storage makes redundant, scalable storage space available. Microsoft announced that they will adapt the OpenStack software to their Hyper-V virtualization technology. The objective is to be able to use Windows and open source programs together in cloud systems.

6.7 AppScale

AppScale [58] offers an open source re-implementation of the Google App Engine functionality and interfaces. AppScale is being developed at the University of California in Santa Barbara. With AppScale, it is possible to run and test Google App Engine-compatible applications within a private cloud (Eucalyptus) or within a public cloud (EC2). Moreover, AppScale can be implemented directly on the Xen Hypervisor, without the need to interpose an IaaS. AppScale supports Python and Java applications and emulates the Google Datastore, Memcache, XMPP, Mail, and Authentication infrastructure services.

6.8 TyphoonAE

TyphoonAE [133] is another open source re-implementation of Google App Engine. Here as well, developers can run Google App Engine-compatible applications in a local environment. Unlike AppScale, TyphoonAE works well with any Linux environment and with Mac OS X. Thus, it is not only suitable for private and public clouds, but also for virtual machines.

Another difference compared to AppScale is that TyphoonAE exclusively supports applications developed in Python. The software is based on the App Engine SDK and on popular open source packages, such as NGINX, Apache2, MySQL, memcached, and RabbitMQ, which are used to emulate the Google infrastructure services.

6.9 Apache Hadoop

Hadoop [31] is an open source software platform which allows to easily process and analyze very large data sets in a computer cluster. Hadoop can for example be used for Web indexing, data mining, logfile analyses, machine learning, finance analyses, scientific simulations, or research in the bioinformatics field.

The Hadoop system features the following properties:

- **Scalability:** It is possible to process data sets with a volume of several petabytes by distributing them to several thousand nodes of a computer cluster.
- **Efficiency:** Parallel data processing and a distributed file system allow to manipulate the data quickly.

- **Reliability:** Multiple copies of the data can be created and managed. In case a cluster node fails, the workflow reorganizes itself without user intervention. Hence, automatic error-correction is possible.

Hadoop has been designed with scalability in mind so that cluster sizes of up to 10,000 nodes can be realized. The largest Hadoop cluster at Yahoo! currently comprises 32,000 cores in 4,000 nodes, where 16 petabytes of data are stored and processed. It takes about 16 h to analyze and sort a one petabyte data set on this cluster.

Cloudera is a company which offers packetized versions of the Hadoop system. Various Hadoop distributions are available on the Internet for download [64].

6.9.1 MapReduce

Hadoop implements the MapReduce programming model which is also of great importance in the Google search engine and applications [11]. Even though the model relies on massive parallel processing of data, it has a functional approach. In principle, there are two functions to be implemented:

- **Map function:** Reads key/value pairs to generate intermediate results which are then output in the form of new key/value pairs.
- **Reduce function:** Reads all intermediate results, groups them by keys and generates aggregated output for each key.

Usually, the procedures generate lists or queues to store the results of the individual steps. As an example, we would like to show how the vocabulary in a text collection can be acquired (see also the example in the appendix): The Map function extracts the individual words from the texts, the Reduce function reads them in, counts the number of occurrences, and stores the result in a list. In parallel processing, Hadoop distributes the texts or text fragments to the available nodes of a computer cluster. The Map nodes process the fragments assigned to them and output the individual words. These outputs are available to all nodes via a distributed file system. The Reduce nodes then read the word lists and count the number of words. Since counting can only start after all words have been processed by the Map function, a bottleneck might arise here. Figure 6.4 shows the schematic workflow of the MapReduce function in a Hadoop cluster.

6.9.2 Hadoop Distributed File System

In order to implement the MapReduce functionality in a robust and scalable way, a highly available, high-performance file system is required. For data processing, Hadoop uses a specific distributed file system called Hadoop Distributed File System (HDFS). Its architecture is based on a master node (name node) which

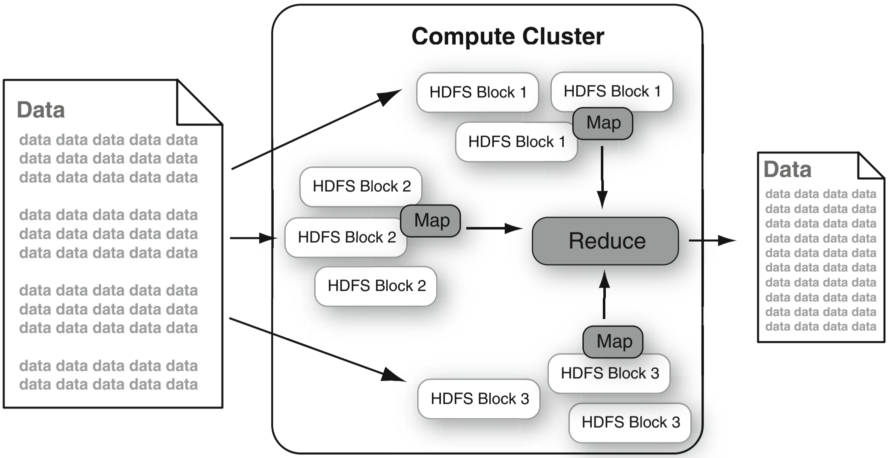


Fig. 6.4 MapReduce programming model based on a distributed file system (HDFS)

manages a large number of data nodes. This master node processes external data requests, organizes the storage of files and saves all metadata for the description of the system status. In practice, the number of files that can be stored in the HDFS is limited by the RAM available on the master node, since, for performance reasons, all data should be transferred to the memory cache. It should be possible to realize systems accommodating several hundred millions of files on current hardware.

Hadoop splits the files and distributes the fragments to multiple data blocks in the cluster, thus enabling parallel access to them. In addition, HDFS saves multiple copies of the data blocks in the cluster. This increases the reliability and guarantees a faster access. Data integrity is ensured by optional checksum calculations: Thus, it is possible to detect potential data corruption, and the read operation can be redirected to an alternative, uncorrupted block. Since the master node is a single point of failure, it is wise to provide for its replication.

Contrary to the widely used RAID systems, HDFS uses a flat storage model. This is mainly with respect to fault tolerance: If a disk fails, a rebuild process takes place creating new distributed copies of the affected blocks. It is important to keep this time as short as possible in order to minimize the risk of a data loss caused by multiple faults. In case of a failure, HDFS needs only about half an hour for rebuilding a terabyte disk (with RAID, this may take several days due to system constraints). If a data node or even an entire rack fails, the master node redelegates the corresponding subtasks immediately.

If at all possible, tasks in the cluster are performed where the corresponding data resides. For efficiency reasons, data access over the network is avoided whenever possible. This is assessed by a distance function criterion describing the access costs: The distance is shortest when data is accessed on the same node, it increases for access operations within the same rack, and further grows with increasing distances on the network.

6.9.3 Pig

In connection with Apache Hadoop, a platform called Pig has been created [31], a special high-level programming environment that can be used to define data analyses (Pig Latin). This environment is coupled with a suitable infrastructure for performing analyses. The salient property of Pig programs is that their structure is amenable to substantial parallelization. A particular feature of the platform is a compiler which generates MapReduce program sequences for Hadoop. The programming language has the following key properties:

- **Ease of programming:** Supports concurrent as well as parallel applications. Complex coupled systems are implemented as data flow sequences.
- **Optimization:** The execution of tasks is automatically optimized so that programmers can remain focused on the semantics of their programs.
- **Extensibility:** It is possible to embed self-developed functions in order to solve domain-specific problems.

Using Pig is particularly helpful in cases where batch processing of large data sets is required. Pig is less suitable for environments where only small subsets of large data sets have to be processed.

6.9.4 Hive

Hive is a central data warehousing application built on top of Hadoop [98]. Hive uses the QL query language to analyze large, structured data sets stored in the Hadoop file system. Derived from SQL, this language allows users to transfer their existing database applications to the Hadoop platform. An interesting feature in this context is the capability to combine database queries with the MapReduce programming model.

6.9.5 Hadoop as a Service

Installing and operating a Hadoop cluster involves considerable overhead which might not pay off, especially if the cluster is only used every once in a while. For this reason, several approaches have emerged to offer Hadoop as a cloud service.

Especially noteworthy in this regard is Amazon Elastic MapReduce [41] which is available as an Amazon Web Services component. Amazon Elastic MapReduce is based on EC2 and is able to provision clusters of nearly any size, depending on the current demand. The Hadoop distributed file system interacts with the S3 service, as shown in Fig. 6.5. The Elastic MapReduce service works as follows:

1. The user uploads the desired data and the Map and Reduce executable to S3.
2. Elastic MapReduce generates and starts an EC2 Hadoop cluster (master + slaves).
3. Hadoop creates a job flow that distributes data from S3 to the cluster nodes to process them there.

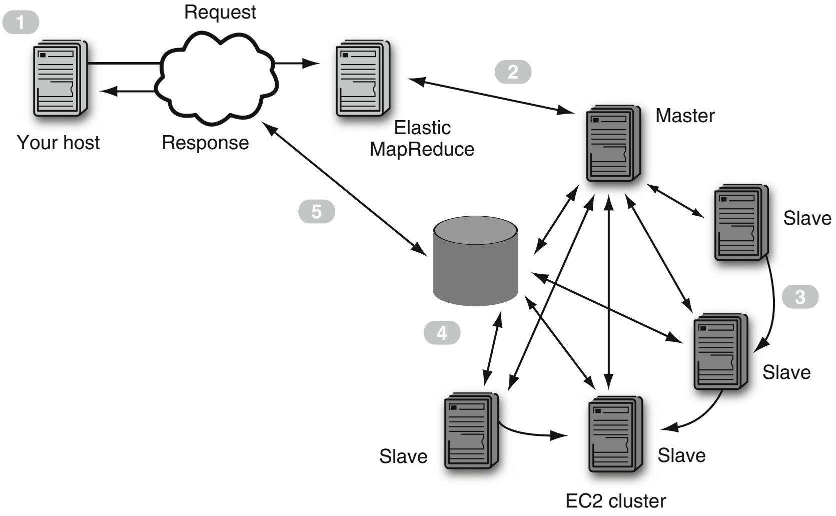


Fig. 6.5 Amazon elastic MapReduce (Source: Amazon)

4. After processing, the results are copied to S3.
5. Notification when the job is finished: The user retrieves the results from S3.

It is possible to control Amazon Elastic MapReduce by specifying parameters in the command line or graphically using a Web console [53]. The appendix of this book contains an example of how to use Elastic MapReduce.

Besides Elastic MapReduce, there is another option to use Hadoop as a flexible service within the Amazon Web Services framework: A dedicated Hadoop cluster of a size suitable to solve the problem can be instantiated in the Amazon infrastructure. For this purpose, a predefined Amazon Machine Image may be used which is provided by Cloudera as a public AMI in the Amazon Web Services context.

6.10 The OpenCirrus Project

The objective of the OpenCirrus™ project is to build and operate an international cloud computing testbed in support of the open source cloud system research [2]. HP, Intel, and Yahoo! started the project in July 2008 together with their academic partners IDA[1], KIT[2], and UIUC[3]. Since 2010, new members ETRI[4], MIMOS[5],

[1] Infocom Development Authority, Singapore

[2] Karlsruhe Institute of Technology, Germany

[3] University of Illinois Urbana Champaign, USA

[4] Electronics and Telecommunications Research Institute, South Korea

[5] Malaysian Institute for Microelectronic Systems, Malaysia

RAS[6], CESGA[7], CMU[8], CERCS[9] as well as China Mobile and China Telecom have joined the consortium. The partners operate federate resource centers, providing up to 1,024 CPU cores and up to one petabyte of storage space each. Their activities include development work on the infrastructure layer as well as on the platform and application layers (see Chap. 3). Unlike with other cloud environments, such as Google App Engine or Amazon Web Services, researchers and developers have full access to all OpenCirrus system resources. This facilitates the further development of the open source cloud stack. All system users must register in the project portal [111]. The portal not only provides general information on the project, but also allows researchers to request access to the resources they need for their work.

Running a cloud system which is distributed over multiple locations requires the deployment of comprehensive shared services. These global basic services are:

- **Identity management:** Identity management is the basis on which all activities can be assigned to a user profile. In this context, it is desirable to have uniform user profiles available at all distributed locations (single sign-on). This can be ensured by using the SSH public key authentication [116]. The public key of an RSA key pair is kept with the resource and the user transmits the private key over a secure connection. By indicating the private key, a client can then access the distributed resources once the operator has registered the public key at the corresponding location. A slightly different version of the OpenCirrus procedure is used with Amazon Web Services.
- **Monitoring:** Another global service monitors the distributed resources, thus enabling the management of the distributed infrastructures and helping to localize and troubleshoot faults. OpenCirrus uses the Ganglia open source project [80] for monitoring. Ganglia gathers information on the resource status and usage for each component and aggregates them hierarchically. For this purpose, the operators install a daemon process which transmits an XML stream containing the resource status and all associated data to a central Web server where this information is collected and consolidated [112].

This forms the foundation for the development and introduction of other global services, i.e. services for common data storage and distributed application development.

Within the scope of this project, the partners will continue the development of the open source components discussed above, namely PRS, VRS, Hadoop, and Tashi. Another goal is to tackle unsolved problems in cloud system research, such as:

- Standardization of interfaces
- Security techniques

[6] Russian Academy of Sciences, Russia

[7] Centro de Supercomputacion Galicia, Spain

[8] Carnegie Mellon University, USA

[9] GeorgiaTech, USA

- Dynamic transfer of workloads (cloud bursting)
- Realization of Service Level Agreements (SLAs)

The test environment is particularly useful to enable large-scale scaling tests and open new horizons for cloud computing. The KIT, for example, examines how to leverage cloud techniques in high-performance computing environments. The idea is to benefit from the dynamic properties of cloud computing for handling computationally intensive parallel applications and to design a corresponding elastic service (High Performance Computing as a Service, HPCaaS). Particular challenges with regard to systems are the provisioning of ensembles of tightly coupled CPU resources and the virtualization of high-speed connections based on Infiniband. On the platform side, the following issues are still open:

- Development of a scheduling service
- Development of MPI services
- Management of software licenses

Further information on OpenCirrus can be found on the project website [111].

Chapter 7
Economic Considerations

The economic impact of cloud computing is often seen as its key success factor. Cloud computing is associated with the potential to fundamentally and sustainably change the way in which IT resources are deployed and used, not only in individual companies, but also in the IT industry as a whole. Experts often mention dramatic time savings, lower risks, and fewer obstacles to the introduction of new applications as well as significant cost savings in general while implementing IT projects as the main benefits of cloud computing. This chapter gives an overview of some fields of application relevant to cloud computing and the economic aspects and issues associated with them.

7.1 Fields of Application

In the early times of cloud computing, the focus was on scalable Web applications developed by companies having only a limited IT infrastructure. Well-known examples are the TimesMachine project initiated by the New York Times [107] or the Animoto [55] video services. In both cases, public cloud services were used to lower the barrier of entry to this world. Thus, an easily scalable infrastructure was available at low cost.

The TimesMachine project was motivated by the desire of making six million articles from the New York Times archives available to the readers as PDF files. Conventional approaches to generate the PDF files by scanning the articles to TIFF format proved to be very costly and time-consuming; the expected data volume of four terabytes would have meant to buy new servers and the project would have taken several months. Instead, the entire set of articles could be made available within only 3 days and at a fraction of the costs simply by using public cloud services.

In the case of Animoto, cloud services helped to cope with an unexpectedly high demand by end users. Animoto offers video services over the Web: End users can upload separate music and image files and then receive an animated video.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_7,

After linking Animoto to the Facebook.com social network, the company was surprised by a quick and dramatic surge in demand: Up to 25,000 new users registered every hour and it took only a short while until the number of new users had grown to more than 250,000. To satisfy the enormous demand for video services, Animoto required a 100-fold of the previous IT infrastructure: 3,500 servers had to be added in only 3 days. To be able to process these requests at all – and without excessive delays – Animoto had to use public cloud services. Later, when the demand slackened again, it was possible to scale down the use of cloud services easily and cost-efficiently.

The question whether cloud computing pays off for a company or an application cannot be answered generally, but must be considered on a case-by-case basis. The answer depends on the properties and goals of the application concerned. The TimesMachine and Animoto examples describe two different classes of applications: TimesMachine was a one-time batch job; in the case of Animoto, there is a continuous demand behavior which varies dynamically. Both cases feature an enormous need for scaling, and for both tasks, parallelization is the perfect option.

Cloud offerings are generally not restricted to specific applications. In principle, each and every application can run in a cloud environment or be combined with cloud services. The reasons for using cloud services vary accordingly; they range from a one-time, temporary demand and the (sometimes periodical) handling of hardly predictable (and sometimes extreme) load peaks to the management of seasonal demand effects or to the outsourcing of functionalities and services to third parties in general. The use of cloud services can make sense for testing purposes as well as in production environments. The provider company may conceal the use of one or more cloud services to the end users, as in the Animoto case, or these services may be directly visible, as in the case of Web-based Office applications.

7.2 Evaluation Models

By dynamically scaling IT resources in cloud computing and by adopting a usage-dependent billing system, companies are enabled to transform fixed expenditure into operating expenditure. *Pay as you go* billing models are based on the actual use of resources, billed over time. This contrasts to purchasing or leasing models where resources are definitely bought or rented and paid for over a fixed period (contract term), independently of their actual use.

The advantage of usage-dependent billing models for the application and for the company lies in the *elasticity* and in the fact that there is little risk of overprovisioning and underutilization (too many resources are available and not used sufficiently) or of underprovisioning and saturation (too few resources are available and peak loads cannot be accommodated). Elasticity refers to the capability of being able to add or remove resources with fine granularity within minutes (and not within weeks or even months) and thus to be able to better meet the real requirements of an application.

In practice, overprovisioning and underutilization are the rule in conventional data centers, and an optimum resource utilization is hardly achievable. If there is an unexpectedly high demand of resources, however, it cannot be met immediately, and the reverse case of an unexpected drop in demand even adds to the degree of underutilization. The financial consequences are not to be neglected. What is more, it is often not possible to reliably predict and schedule the resource demand. Many e-commerce applications face a highly varying demand which is not only due to seasonal changes, but may be influenced strongly by temporary trends and effects (for instance when triggered by social networks).

7.2.1 Cost Models

To evaluate cloud services from an economic point of view, cost models can be used. In comparative cost calculations, the costs of the actual usage of cloud offerings (e.g. in hours and server units) are contrasted with the data center costs (or costs of an IT infrastructure which has to be purchased and maintained by the operator).

A very simple model is suggested in [1], where the costs of a cloud service are compared to the costs of a data center, factorizing the average utilization of the data center and assuming that the customer's profit is proportional to the number of resource-hours. It has to be noted, however, that a data center always has a fixed capacity while (public) cloud services theoretically have no upper capacity limit.

The costs incurred by a data center are manifold, because operating expenses such as power (in particular for computer cooling systems), rental and real estate costs as well as costs for system administration staff have to be taken into account besides the hardware acquisition costs.

In this context, the energy costs make up for the major part of the total cost of ownership (TCO). Generally, efficient use of energy can be achieved more easily in large data centers than in small ones. Owners of large data centers are able to realize substantial cost savings by negotiating lower energy purchase prices or by distributing or transferring data centers to regions with reduced energy costs. In addition, they benefit from significantly lower hardware purchase prices so that the money saved here can be invested in failsafe and data security systems. A Microsoft study accordingly identifies disadvantages for small corporate data centers and forecasts a long-term trend towards multi-tenant utilization of very large data centers [104]. Currently, data centers of unparalleled dimensions are emerging all over the world; they take up enormous surface areas and represent investments in the order of several hundred million US dollars. Global data centers will comprise more than 100,000 servers and thus can dramatically reduce the TCO per server. According to this study, the costs will not pay off for organizations having less than

100 servers. For organizations that operate about 1,000 servers, the costs of using a private cloud will be ten times higher than those for a public cloud.

For consumers, likewise, it is advisable to take a closer look at the calculation of cloud service costs. Current cloud offerings distinguish, for example, the costs of provisioning a computing or storage unit (in the case of an IaaS) from the costs of resource interaction, i.e. the number of invocations or the data volume transmitted over the Internet. Thus, cost determination largely depends upon the properties and the requirements of the application in question.

7.2.2 TCO Framework

Currently, a number of costing models and decision support systems are being developed both in the field and in application-oriented research. An example of a TCO (Total Cost of Ownership) framework specifically developed for cloud computing is presented in [18]. Starting with a qualitative analysis of the application (business scenario), a quantitative cost analysis in a second step contrasts the costs for a cloud service with the costs for a reference architecture. By comparing the opportunity costs, an assessment is obtained.

Figure 7.1 illustrates the different cost estimation steps which can be supported by corresponding tools. The first step is to create a model of the application with all its relevant requirements and properties. Later, the required factors for a quantitative cost analysis are selected and weighted based on qualitative requirements and properties. The simplest tool can be a matrix listing the available cloud services and alternative architectures in rows and the different requirements and properties in columns. This matrix can then be evaluated like a questionnaire (or *checklist*).

In [21], this idea has been extended into a decision support system which helps consumers to relate different criteria, requirements, and preferences of various kinds as well as technical, economic, or legal aspects in a systematic way so that they can use an iterative process to decide whether to accept or reject a cloud offering.

7.3 Business Models

In the wake of cloud computing, new business models have emerged which vary depending on the position of the service offering in the cloud computing stack. In this context, the dynamic character of the Internet as a continually changing technology, business, and collaboration platform plays a key role. Due to the acceptance and popularity of the Internet which encompasses all facets of daily life and of business, many novel IT-based services are emerging that lend themselves to a service-oriented society.

In the IaaS area, for example, open cloud architectures allow to add a number of third-party services, thus forming the basis of an ecosystem of providers who offer

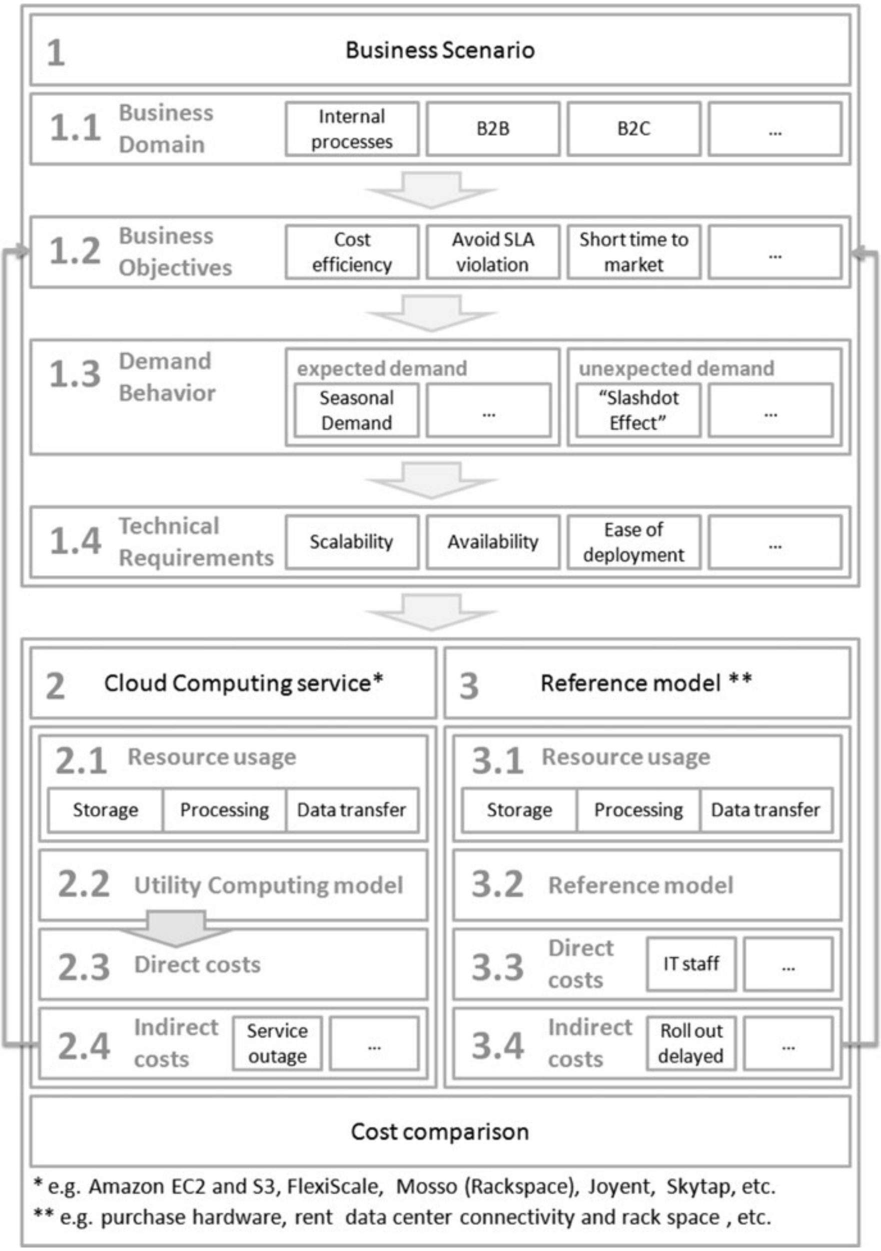


Fig. 7.1 TCO framework

added value, such as for cloud application monitoring and management. This spawns market and business models that operate in the context of combined cloud services from different providers.

In the PaaS and SaaS areas, however, many of the existing business models have been coined primarily by a single provider. *Software-Plus-Services* strategies, such as pursued by Microsoft, combine the classical license-based software approach with the flexibility and the advantages of Internet-based cloud services. Thus, it is possible to supplement *on-premise* software products with Internet services or to replace them entirely with *on-demand* solutions.

Chapter 8
Opportunities and Risks

Cloud computing is still a very young and dynamic field characterized by a buzzing industry. Virtually every organization in the industry and even parts of the public sector are taking on cloud computing today, either as a provider or as a consumer. Primarily US-based companies, such as Amazon, Google, or Microsoft are currently shaping the cloud services market. But many smaller companies also show a lot of commitment in this field. There is no doubt that cloud computing is a *disruptive technology* which has the potential to change our understanding of how to provision and leverage IT services in a fundamental and sustainable way. The effects might even be comparable to the introduction of the personal computer some 25 years ago.

8.1 Market Development

There is already a rich ecosystem of cloud services and providers. The most diverse services can be developed, tested, and operated. On the one hand, these are commercial *public cloud* offerings on the infrastructure, platform, and application levels, on the other hand, *private clouds* can be built and deployed using products like VMware vSphere [136] or open source developments such as Eucalyptus [74] and Hadoop. This way, a dynamic landscape is created where, in the near future, it will be possible to transparently compose and orchestrate services from public and private sources. This opens up a variety of opportunities to dynamically optimize both the quality of service and the costs.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_8,

8.2 Situational Evaluation

Cloud computing offers a multitude of opportunities, but also entails some risks. For one thing, developers and users are faced with the difficult task of selecting a suitable service from this vast choice of cloud offerings which sometimes differ considerably as to their properties and conditions. For another thing, there are concerns about the safety and privacy of data as well as the vendor lock-in issue. In this context, it is important to know that the desired degree of security can be achieved by data encryption: In some cases, encrypted data stored in the cloud are safer than unencrypted data located in the user's data center. With respect to vendor lock-in, it has to be said that data access can be controlled by establishing corresponding agreements between the business partners. Amazon, for instance, offers corresponding import and export services for large-volume data sets.

Armbrust et al. [1] name ten risks which at the same time are opportunities for the adoption and growth of cloud computing (see Table 8.1). The list includes questions on the availability of the services and other quality properties, on data lock-in and potential performance, scalability, and troubleshooting problems. For the risks which have been identified, solutions are conceivable, even if some of them are currently still under research. Please refer to [1] for a more extensive discussion of these risks and opportunities.

There are already signs that security-related issues can be solved by technological progress, that legal issues are also being taken into account and, what is more, that the attitude and perception of consumers are moving towards a greater confidence in cloud technologies.

Table 8.1 Risks and opportunities of cloud computing (selection)

	Risks	Opportunities
1	Service availability	Addressing multiple cloud providers to ensure service continuity
2	Data lock-in	Standardization of interfaces (APIs)
3	Data privacy and traceability	Use of data encryption, virtual networks (VLANs), and firewalls Compliance with national legislation by geographical data storage
4	Data transfer bottlenecks	Shipment of hard disks via third parties (e.g. FedEx, UPS[a])
5	Poor performance predictability	Better support of virtual machines. Use of flash memory.
6	Scalability of persistent storage space	Development of scalable storage space technologies
7	Errors in large, distributed systems	Development of debuggers for distributed machines
8	Quick scalability	Development of automatic scaling tools based on machine learning Resource and cost awareness on the part of users and providers
9	Reputation and liability	Use of third-party services, e.g. for confidential e-mails
10	Software licenses	Pay-for-use licenses. Software and services are sold as a package

[a] As described in Sect. 4.1, this is already common practice

8.3 Conclusion

From an economic point of view, cloud computing has become one of the strongest emerging markets in the IT industry: According to Gartner, 2009 cloud services sales already reached 56 billion dollars; Meryll Lynch analysts predict a market volume of up to 160 billion dollars for 2011. In the light of these findings, CIOs and IT managers should continue addressing the innovative topic of cloud computing and make their IT landscapes fit for this trend.

Cloud computing is and obviously remains an exciting, promising topic. We hope that, with this little compendium, we succeeded in giving you a short and concise overview of the current developments, technologies, and trends.

Chapter 9
Appendix

In this appendix, we will finally present some cloud offerings and their basic tools in real-world examples so the our dear readers can get started quickly with their own first cloud exercises.

9.1 Performing EC2 Tasks with the Amazon Tools

The following examples show how typical steps in EC2 can be performed using the Amazon EC2 command line tools [43]. First, you must create a key pair for the desired region. These keys can be reviewed and deleted at any time.

- Create a `secret` key pair in the `eu-west-1` region
  ```
  ec2-add-keypair secret --region eu-west-1
  ```
- Review the key pairs
  ```
  ec2-describe-keypairs --region eu-west-1
  ```
- Delete the key pair
  ```
  ec2-delete-keypair secret
                        --region eu-west-1
  ```

When working with different EC2 applications, it is recommended to set up and use dedicated security groups. Each of these security groups may have different firewall settings so that any requirements with respect to openness and security can be met.

- In the `eu-west-1 region`, create a security group named `west_group` and add the description `This is a new group`
  ```
  ec2-add-group west_group
                       --region eu-west-1
                       -d "This is a new group"
  ```
- Open TCP port 80 for the security group
  ```
  ec2-authorize west_group -P tcp -p 80
  ```

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8_9,
© Springer-Verlag Berlin Heidelberg 2011

- Review the firewall rules for the new security group
  ```
  ec2-describe-group west_group
                           --region eu-west-1
  ```
- Delete the security group
  ```
  ec2-delete-group west_group
                          --region eu-west-1
  ```

The following commands demonstrate how you typically work with images and instances. These commands are used to review and start images and to review, stop, start and remove instances.

- List all images in the `eu-west-1` region whose owner is Amazon itself
  ```
  ec2-describe-images -o amazon
                          --region eu-west-1
  ```
- Create two instances of the image named `ami -13042f67` in the `eu-west-1` region in the `eu-west-1a` availability zone with the `secret` key pair and the `m1.small` instance class within the `west_group` security group
  ```
  ec2-run-instances ami-13042f67
                          --region eu-west-1
                          -z eu-west-1a
                          -k secret
                          -g west_group
                          -n 2 -t m1.small
  ```
- Review the instances in the `eu-west-1` region
  ```
  ec2-describe-instances --region eu-west-1
  ```
- Stop the `i- f1e1a786` instance
  ```
  ec2-stop-instances i-f1e1a786
                           --region eu-west-1
  ```
- Start the instance
  ```
  ec2-start-instances i -f1e1a786
                           --region eu-west-1
  ```
- Remove the instance
  ```
  ec2-terminate-instances i -f1e1a786
                           --region eu-west-1
  ```

When you start an instance, its private and public DNS names are generated. In order to use the EC2 resources efficiently, you often need a fixed IP address. Amazon ensures this by providing elastic IP addresses.

- Create a new elastic IP address in the `eu-west-1` region
  ```
  ec2-allocate-address --region eu-west-1
  ```
- Review the elastic IP addresses
  ```
  ec2-describe-addresses --region eu-west-1
  ```
- Associate the elastic IP address `79.125.121.3` with the `i-f1e1a786` instance
  ```
  ec2-associate-address 79.125.121.3
                           -i i-f1e1a786
                           --region eu-west-1
  ```

- Delete the elastic IP address
  ```
  ec2-disassociate-address 79.125.121.3
                           --region eu-west-1
  ```
- Release the elastic IP address
  ```
  ec2-release-address 79.125.121.3
                           --region eu-west-1
  ```

9.2 Performing EBS Tasks with the Amazon Tools

The following commands from the Amazon EC2 command line tools can be used to perform typical tasks with EBS volumes, i.e. creating, reviewing, mounting, unmounting, and deleting volumes.

- Create a 1 GB volume in the `eu-west-1b` availability zone within the `eu-west-1` region
  ```
  ec2-create-volume --size 1
                          --region eu-west-1
                          -z eu-west-1b
  ```
- Review the volumes in the `eu-west-1` region
  ```
  ec2-describe-keypairs --region eu-west-1
  ```
- Attach the volume named `vol-e466838d` to the `i-58859c2c` instance and the `/dev/sdf` device ID
  ```
  ec2-attach-volume vol-e466838d
                          --region eu-west-1
                          -i i-58859c2c
                          -d /dev/sdf
  ```
- Use commands to work with the volume within the instance
  ```
  mkdir /mnt/ebsvolume1
  yes | mkfs.ext3 /dev/sdf
  mount /dev/sdf /mnt/ebsvolume1
  umount /mnt/ebsvolume1
  ```
- Detach the volume from the instance
  ```
  ec2-detach-volume vol-e466838d
                          --region eu-west-1
                          -i i-58859c2c
  ```
- Delete the volume
  ```
  ec2-delete-volume vol-e466838d
                          --region eu-west-1
  ```

It is also possible to create so-called snapshots of EBS volumes which can later be used for building new volumes.

- Create a snapshot of the volume named `vol-e466838d`
  ```
  ec2-create-snapshot vol-e466838d
                          --region eu-west-1
  ```
- Retrieve the status of snapshot `snap-820de2eb`
  ```
  ec2-describe-snapshots snap-820de2eb
                          --region eu-west-1
  ```
- Create a volume from this snapshot
  ```
  ec2-create-volume --snapshot snap-820de2eb
                          --region eu-west-1
                          -z eu-west-1b
  ```

9.3 Performing RDS Tasks with the Amazon Tools

The following commands from the Amazon RDS command line tools [51] illustrate how to work with the Relational Database Service:

- Create an 20 GB RDS instance named `mydb` of the `db.m1.large` instance type
  ```
  rds-create-db-instance --engine mysql5.1
              --db-instance-identifier mydb
              --db-instance-class db.m1.large
              --allocated-storage 20
              --master-username dbroot
              --master-user-password dbpass
  ```
- Display the instance status
  ```
  rds-describe-db-instances
  ```
- Grant access to certain IP address ranges (CIDR notation)
  ```
  rds-authorize-db-security-group-ingress default
               --cidr-ip 0.0.0.0/0
  ```
- In the RDS instance, you can now use tools or programs, e.g. in order to import a mysql database which can then be manipulated interactively
  ```
  mysql -h mydb.cpjgp4subzas.us-east-1.rds.amazonaws.com
              -u dbroot --password=dbpass
  mysql>CREATE DATABASE world;
  mysql>USE world;
  mysql>SOURCE world.sql;
  mysql>SELECT * FROM world.City LIMIT0, 100; mysql>quit
  ```
- If more space is required, you can increase the instance size dynamically
  ```
  rds-modify-db-instance mydb
              --apply-immediately -s 50
  ```
- Create a snapshot named `mySnapshot`
  ```
   rds-create-db-snapshot mydb -s mySnapshot
  ```
- Delete the snapshot
  ```
  rds-delete-db-snapshot mySnapshot
  ```

- Delete the instance without keeping the data in a final snapshot
  ```
  rds-delete-db-instance mydb --skip-final-snapshot
  ```
- If you intend to re-use the database, however, it is wise to create a final snapshot
  ```
  rds-delete-db-instance mydb
     -final-db-snapshot-identifier myWorldDatabase
  ```
- From the snapshot, you can re-instantiate the database later
  ```
  rds-restore-db-instance-from-db-snapshot mydb
     -db-snapshot-identifier myWorldDatabase
  ```

9.4 Performing S3 Tasks with s3cmd

`s3cmd` is a helpful command line tool for working with Amazon S3.

- Configure the access data
  ```
   s3cmd --configure
  ```
- Display a list of my buckets
  ```
  s3cmd ls
  ```
- Create a bucket
  ```
  s3cmd mb s3://Bucket
  ```
- Upload objects to a bucket
  ```
  s3cmd put localFile s3://Bucket/remoteFile
  ```
- Retrieve the contents of a bucket
  ```
  s3cmd ls s3://Bucket
  ```
- Download objects from a bucket
  ```
  s3cmd get s3://Bucket/remoteFile localFile
  ```
- Delete objects from a bucket
  ```
  s3cmd del s3://Bucket/remoteFile
  ```
- Delete an (empty) bucket
  ```
  s3cmd rb s3://Bucket
  ```

The `s3cmd` configuration is stored in a file named `.s3cfg`. If you want to use an alternative configuration, you can specify it in the command line as follows: `-c s3cfg.alternative`.

9.5 Using Google App Engine

In Google App Engine, each user can launch up to 10 applications. New application names may be reserved on the App Engine website [83]. Please note, however, that each application name must be unique in App Engine, i.e. it may be used one time only.

The applications within Google App Engine can be accessed by entering the following address:

```
http://<application>.appspot.com
```

The so-called dashboard [83] holds all relevant information on the user's applications. To facilitate the development process, multiple versions of each application may be uploaded, each of them being directly accessible:
`http://<#>.latest.<application>.appspot.com`

For developing and running your own applications, you need a Web browser environment (any Web browser will do) and the App Engine SDK [84] for Python or Java. The SDK is available for the following operating systems: Linux/UNIX, Mac OS X, and Windows. In addition, the Google Eclipse plug-in [91] can be used.

App Engine exclusively supports Python 2.5.2 and Java 6.

The following steps illustrate how to develop and run a simple Python application from the Linux command line.

- Review the Python version number
```
python --version
```
- Install the App Engine SDK
```
unzip google_appengine_1.3.1.zip
export PATH=$PATH :~/google_appengine
appcfg.py -h
```
- Create an application directory
```
mkdir gae-example
cd gae-example
```
- Create the `app.yaml` configuration file
```
application: gae-example
version: 1
api_version: 1
runtime: python
handlers:
- url: .*
  script: main.py
```
- Create a miniature application in `main.py`
```
#!/usr/bin/env python
print 'Hello world'
```
- Using the development server, start the application in the App Engine SDK
```
dev_appserver.py gae-example/
```
- Test the application in the Web browser by entering the following URL:
`http://localhost:8080`
- Transfer the application to App Engine
```
appcfg.py --passin --email = <email>
    update gae-example/
```
- Request the application log data
```
appcfg.py --passin --email=<email>
    request_logs gae-example/ logs.txt
```

9.6 Using AppScale

The AppScale tools are a collection of Ruby scripts that can be used to control the AppScale private cloud PaaS.

- Create the SSH keys for the instances
  `appscale-add-keypair`
- Start the AppScale private cloud instances
  `appscale-run-instances`
- Check the status of all instances
  `appscale-describe-instances`
- Upload the application to the instances
  `appscale-upload-app.`
- Set the administrator password
  `appscale-reset-pwd`
- Remove the application from the instances
  `appscale-remove-app`
- Stop the AppScale private cloud instances
  `appscale-terminate-instances`

9.7 Installing and Using Eucalyptus

Eucalyptus can be installed from binary packages or by compiling from source. For a detailed description of the installation procedure, please refer to the project website [74]. It is recommended to use the latest Ubuntu server version because it already contains all Eucalyptus packages and thus allows for a fast installation. The Eucalyptus cloud infrastructure can be controlled using the Euca2ools which are a component of the Eucalyptus distribution.

As an alternative, it is possible to use the Amazon EC2 command line tools.

To list the resources that can be accessed by Eucalyptus, the `euca-describe-availability-zones verbose` command can be used.

The output includes the number of existing and free resources. Five categories of virtual machines with different capabilities in terms of computing power, RAM, and disk capacity have been defined. These five categories are:

- Small instance `(m1.small)`
- High CPU medium instance (`c1.medium`)
- Large instance `(m1.large)`
- Extra large instance `(m1.xlarge)`
- High CPU extra large instance `(c1.xlarge)`

The list also indicates the connected NCs and the cluster names. The category identifiers for Eucalyptus and Amazon EC2 are identical. They differ, however, in the resources allocated to the instance classes, as shown in Tables 6.1 and 6.2. Unlike Amazon AWS, resource allocation within a Eucalyptus private cloud can be defined freely in the different instance classes. However, it is not possible to define additional or to rename existing instance classes.

Before you can start virtual machines (instances) in the cloud, at least one file system including an installed operating system, a kernel, and, if applicable, a RAM disk need to be registered in the Eucalyptus system as images. For this purpose, the `euca-bundle-image`, `euca-upload-bundle` and `euca-register` commands are used.

During the registration of the file system/kernel/RAM disk images, unique identifiers are assigned to the Eucalyptus Machine Image `(emi-xxxxxxxx)`, the Eucalyptus Kernel Image `(eki-xxxxxxxx)`, and the Eucalyptus RAM Disk Image `(eri-xxxxxxxx)`. The `euca-describe-images` command gives an overview of the installed file system/kernel/RAM disk images.

Before instantiating the virtual machines, a key pair should be generated that can be used later to access the resource. Invoking the `euca-add-keypair` command returns the private key and saves it in a local file for later use:

```
# euca-add-keypair mykey > mykey.private
```

Then, the new private key is protected against unauthorized use:

```
# chmod 0600 mykey.private
```

To list the available keys, enter the `euca-describe-keypairs` command:

```
# euca-describe-keypairs
KEYPAIR mykey 33:da:6e:13:96:e6:f7: 3b:b7:34:a6:28:
ba:2f:64:ab:83:70:ef:70
```

Free resources and available instance categories

```
# euca-describe-availability-zones verbose
AVAILABILITYZONE   SCC_Cloud      141.52.167.65
AVAILABILITYZONE   |- vm types    free / max    cpu   ram    disk
AVAILABILITYZONE   |- m1.small    0026 / 0032   1     384    4
AVAILABILITYZONE   |- c1.medium   0026 / 0032   1     768    6
AVAILABILITYZONE   |- m1.large    0025 / 0030   1     1280   10
AVAILABILITYZONE   |- m1.xlarge   0022 / 0024   1     2048   16
AVAILABILITYZONE   |- c1.xlarge   0012 / 0015   2     2048   16
```

Register the kernel image

```
# euca-bundle-image -i vmlinuz --kernel true
# euca-upload-bundle -b kernelbucket -m /tmp/vmlinuz.manifest.xml
# euca-register kernelbucket/vmlinuz.manifest.xml
IMAGE eki-803516F2
```

Register the RAM dis

```
# euca-bundle-image -i initrd.img --ramdisk true
# euca-upload-bundle -b rambucket -m /tmp/initrd.img.manifest.xml
# euca-register rambucket/initrd.img.manifest.xml
IMAGE eri-E76C1849
```

Register the operating system (file system) image

```
# euca-bundle-image -i ubuntu.img --kernel eki-803516F2
--ramdisk eri-E76C1849
# euca-upload-bundle -b imagebucket -m /tmp/ubuntu.img.manifest.xml
# euca-register imagebucket/ubuntu.img.manifest.xml
IMAGE emi-8D1F1369
```

Display an overview of the installed file system/kernel/RAM disk images

```
# euca-describe-images
IMAGE emi-8D1F1369            imagebucket/ubuntu.img.manifest.xml
                              admin available public x86_64 machine
IMAGE eki-803516 F2           kernelbucket/vmlinuz.manifest.xml
                              admin available public x86_64 kernel
IMAGE eri-E76C1849            rambucket/initrd.img.manifest.xml
                              admin available public x86_64 ramdisk
```

Start two instances

```
# euca-run-instances emi-8D1F1369          --kernel eki-803516F2
                                           --ramdisk eri-E76C1849
                                           -k mykey -n 2 -t m1.small
```

Overview of the currently running instances

```
# euca-describe-instances
RESERVATION   r-3DDE07D9   admin          default
INSTANCE      i-4901084 F  emi-8D1F1369   141.2.3.160   141.2.3.160
running       mykey        0              m1.small
2010-09-01 T13:54:28.917Z  SCC_Cloud      eki-803516F2  eri-E76C1849
RESERVATION   r-42FA0732   admin          default
INSTANCE      i-463B08BE   emi-8D1F1369   141.2.3.161   141.2.3.161
running       mykey        0              m1.small
2010-09-01 T13:54:28.917Z  SCC_Cloud      eki-803516F2  eri-E76C1849
```

Terminate both instances

```
# euca-terminate-instances i-4901084 F i-463B08BE
```

Then, the images can be started as instances with the `euca-run-instances` command. If more resources are required than the most basic category `(m1.small)` can provide, you can select a higher instance class (e.g. `m1.large)` when starting the instance. In the example above, two instances of the previously registered Debian 5.0 images with Kernel 2.6.26 and a suitable RAM disk are created. It is agreed that the instances can be accessed by entering the `mykey` key.

To monitor the instances, you can use the `euca-describe-instances` command. Each instance has an IP address and a unique instance number `(i-xxxxxxxx)`.

Via Secure Shell, you can directly log on to a virtual machine by indicating the key that was created upon instance startup:

```
# ssh -i mykey.private 141.52.166.160
```

The `euca-terminate-instances` command terminates one or more instances.

The command line tools presented here help to automate certain tasks. They are intended for system administrators rather than for normal users. For simpler applications, there are appropriate tools with graphical user interfaces, such as ElasticFox [72].

9.8 Data Mining with Amazon Elastic MapReduce

The MapReduce programming model provides a very elegant way to solve statistical problems in conjunction with very large data collections. The following simple task will be used as an example:

- How large is the vocabulary in Shakespeare's works?
- Which word is used most often there?
- Which protagonist is mentioned most often?

We will solve this problem by using the Amazon Elastic MapReduce console [53], as described in Chap. 6. Three steps are required:

1. Upload Data to Amazon S3 Bucket: First, the Shakespeare texts are required. They are available for instance in the Gutenberg project [95] from where they can be downloaded as an ASCII file. In order to be used as input, these texts must be transferred to Amazon S3 and stored in a common directory (e.g. with S3Fox in `< YourBucket>/input`).
2. Create a Job Flow on Amazon Elastic Map Reduce: After clicking the *Create New Job Flow* button, the job properties can be specified.

   - Define Job Flow: Enter a flow job name (e.g. *Shakespeare*) and
   - Select the *Word Count (Streaming)* application from *Run a sample application*
   - Specify Parameters: location of the input and output directories (e.g. `<YourBucket>/input, <YourBucket>/output`); mapper: word Splitter.py; reducer: *aggregate*
   - Configure EC2 Instances: Specify the number and size of the resources required
   - Bootstrap Actions: Proceed with no bootstrap actions
   - Review: Check the settings

   Then, the job flow can be created and processed *(Create Job Flow)*. The setup may take some minutes, then the *WordCount* example is run using the Shakespeare texts. The mapping function is executed simultaneously on the specified EC2 instances. It splits the texts from each file and supplies output in the form of word lists. These word lists are evaluated by the Reducer nodes which also run simultaneously and the results are then stored as lists in the selected S3 output directory `(part-00000, part-00001,...)`.
3. Get Results from Amazon S3 Bucket: The results can then either be processed directly in an EC2 instance of the Amazon cloud or they can be downloaded to a

local machine. For this purpose, you can use either S3Fox or a command line command (if the required file shares have been defined):

```
# wget -r
http://<YourBucket>.s3.amazonaws.com/output/part-00000
```

Next, the intermediate results provided by the Reducer function must be aggregated and sorted alphanumerically:

```
# cat output/part-*      | awk'{print $2, $1}'
                         | sort -n -r > result
```

Now, the initial questions can be answered:
How large is the vocabulary in Shakespeare's works?

```
# wc result
23688              47376              247338              result
```

Which word is used most often there?

```
# head  result
29854  the
27554  and
23357  i
21075  to
18520  of
15523  a
14264  you
12964  my
11955  that
11842  in
```

Which protagonist is mentioned most often?

```
# grep henry result
1361  henry
```

Glossary

Amazon Online retailer and Web service provider with a strong presence in cloud computing. Amazon's Web address: http://www.amazon.com; Amazon's Web services: http://aws.amazon.com

App Engine Programming environment for the Google infrastructure used to develop Web applications in Python or Java.

AppScale Free private cloud implementation of Google App Engine.

AWS Amazon Web Services. Collection of various cloud computing Web services provided by Amazon.

Azure Windows Azure platform. PaaS from Microsoft.

Cloud Provisioning of scalable IT services as Web services with usage-dependent billing.

Cloudera Free, easy to install Hadoop distribution.

Cloud gaming Service which leverages cloud computing to make high-end video games available on low-end devices, e.g. TV sets, older PCs/MACs, and mobile end devices such as smart phones.

Cloud Print Printing from within the cloud. With this concept devised by Google, print jobs submitted by (mobile) end devices are sent to a printer on the Internet, either directly or via a proxy which handles the necessary conversion.

CloudStack Free private cloud IaaS from Cloud.com with an EC2-compatible interface.

Cluster Group of closely coupled computers in a network which are managed and used jointly.

CRM Customer Relationship Management. System that supports communication with customers. Ensures that standardized workflows are used for marketing, sales, and customer service.

Crowdsourcing See Humans as a Service (HuaaS).

Cumulus Free private cloud storage service for Web objects with an S3-compatible interface. Cumulus is a Nimbus component.

EBS Amazon Elastic Block Store. Provides block-oriented storage space for the virtual EC2 servers.

C. Baun et al., *Cloud Computing*, DOI 10.1007/978-3-642-20917-8,

EC2 Amazon Elastic Compute Cloud. A Web service which allows to run virtual instances on the Amazon servers.

Elasticity In cloud computing, resources can be added and removed with fine granularity and within minutes in order to satisfy the actual demand of an application.

Emulator Functional duplicate of the entire hardware of a computer system. Applications or operating systems for a different hardware architecture can be used without making any changes.

Eucalyptus Free implementation of the EC2, S3, and EBS Amazon Web Services.

Google Storage Web services-based storage service for Web objects.

Grid Technology for integrating and sharing distributed, heterogeneous resources independently from their actual location.

Hadoop Platform for working with programs that handle large data sets, based on MapReduce. This is an Apache project powered by Yahoo! It was named after a yellow stuffed elephant owned by the son of chief developer Doug Cutting.

Hive Hive is a central data warehousing application built on top of Hadoop.

HPCaaS High Performance Computing as a Service.

HuaaS Humans as a Service. Adopts the principle of crowdsourcing, i.e. human creativity is offered as a resource.

Hybrid cloud Combination of public and private clouds.

Hypervisor Meta-operating system in virtualization which distributes the hardware resources among the guest systems and is responsible for access coordination.

IaaS Infrastructure as a Service. Implements an abstract view on hardware with the purpose of offering virtual IT components in a cloud.

LaaS Landscape as a Service. Complex software systems with no or only limited multi-tenancy support, such as SAP R3, offered as SaaS.

MapReduce Programming model for parallel data analysis. This algorithm was published by Google in 2004.

Multi-tenancy Capability of managing multiple tenants (customers) on the same server or software system, while none of the tenants can see the other tenants' data and applications.

Nimbus Free private cloud IaaS with an EC2-compatible interface, based on the Globus 4 grid middleware.

OpenNebula Free private cloud IaaS with an EC2-compatible interface.

Open Source Software whose source is publicly available and subject to an open source license approved by the Open Source Initiative (OSI), a body which is dedicated to promoting the development of open source software.

PaaS Platform as a Service. Virtual programming and execution environment for applications which enables transparent scaling. Billing is based on the actual resource consumption and usage time.

Paravirtualization Virtualization technique allows to provide an application interface to the guest operating systems which, in turn, need to be modified accordingly because each direct hardware access must be replaced by the corresponding hypervisor interface call.

Pig Programming environment which includes an optimizing MapReduce compiler built on top of Hadoop. The associated programming language is called *Pig Latin*.

Private Cloud Cloud services are operated *in-house*, i.e. within a company.

Public Cloud Cloud services are operated by an external cloud provider.

RDS Relational Database Service. This service provided by Amazon allows the deployment and operation of a relational database.

REST Representational State Transfer. Style of software architecture based on the World Wide Web.

S3 Amazon Simple Storage Service. Web services-based storage service for Web objects.

SaaS Software as a Service. Software is operated by a provider and can be used over the Internet. The services are billed for the time they were actually used.

SDC Secure Data Connector. Applications within App Engine can be integrated into the user's infrastructure.

Service See Web service.

SimpleDB Distributed database system in AWS which provides a simple, relational database model.

SLA Service Level Agreement. Agreement on the quality of service.

SOA Service-oriented architecture. A technical, organizational, and business architecture based on services.

SOAP Messaging standard for the communication in networked systems.

SQS Simple Queue Service. A messaging service in AWS which provides a simple message queue.

TCO Total Cost of Ownership. Total cost of a service.

TyphoonAE Free private cloud implementation of Google App Engine.

URI Uniform Resource Identifier. String which uniquely identifies a resource.

Virtualization Abstraction layer between the services and the IT infrastructure.

VPC Virtual Private Cloud. Allows to integrate AWS EC2 resources into an existing IT infrastructure using an encrypted VPN tunnel.

Walrus Free private cloud storage service for Web objects with an S3-compatible interface. Walrus is a Eucalyptus component.

Web service A Web service is a software application uniquely identified by a URI, whose interfaces can be defined, described, and located as XML artifacts. A Web service supports the direct interaction with other software agents through XML-based messages which are exchanged via Internet protocols.

XML Extensible Markup Language. Standardized language used to represent hierarchically structured information in the form of text files.

Bibliography

1. Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, and Zaharia M. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report No. UCB/EECS-2009-28. Electrical Engineering and Computer Sciences. University of California at Berkeley. USA. 2009
2. Avetisyan A, Campbell R, Gupta I, Heath M, Ko S, Ganger G, Kozuch M, O'Hallaron D, Kunze M, Kwan T, Lai K, Lyons M, Milojicic D, Lee HY, Soh YC, Ming NK, Luke JY, Namgoong H. Open Cirrus: A Global Cloud Computing Testbed. IEEE Computer, Vol 43, No 4, pp. 42–50, 2010.
3. Baun C und Kunze M. Infrastrukturen für Clouds mit Eucalyptus selbst aufbauen. iX 4/2009. S.128–130. Heise Zeitschriften Verlag
4. Baun C, Kunze M und Ludwig T. Servervirtualisierung. Informatik-Spektrum 3/2009. S.197–205
5. Bengel G, Baun C, Kunze M und Stucky K-U. Masterkurs Parallele und Verteilte Systeme. Grundlagen und Programmierung von Multicoreprozessoren, Multiprozessoren, Cluster und Grid. Vieweg und Teubner, Wiesbaden. 2008
6. Benslimane D, Dustdar S, and Sheth A. Services Mashups: The New Generation of Web Applications. IEEE Internet Computing. http://dx.doi.org/10.1109/MIC.2008.110. IEEE Educational Activities Department. Piscataway. NJ. USA
7. Berg J, Forsythe R, Nelson F, and Rietz T. Results from a dozen years of election futures markets research. Handbook of Experimental Economic Results. 2001
8. Carr N. The Big Switch. Rewiring the World, from Edison to Google. W.W.Norton. 2008
9. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A und Gruber RE. Bigtable: A Distributed Storage System for Structured Data. Symposium on Operating System Design and Implementation. 2006
10. Chisnall D. The Definitive Guide to the Xen Hypervisor. Prentice Hall. USA. 2008
11. Dean J and Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. OSDI2004. San Franzisko. 2004.
12. DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, and Vogels W. Dynamo: Amazon's Highly Available Key-Value Store. ACM Symposium on Operating Systems Principles. 2007
13. Dostal W, Jeckle M, Melzer I und Zengler B. Service-orientierte Architekturen mit Web Services. Spektrum. 2005
14. Dragovic B, Fraser K, Hand S, Harris T, Ho A, Pratt I, Warfield A, Barham P, and Neugebauer R. Xen and the Art of Virtualization. Proc. ACM Symposium on Operating Systems Principles. 2003
15. HP Integrated Lights-Out 2 User Guide. HP

16. Hibler M, Ricci R, Stoller L, Duerig J, Guruprasad S, Stack T, Webb K, and Lepreau J. Large-scale Virtualization in the Emulab Network Testbed. Proceedings of the 2008 USENIX Annual Technical Conference. 2008
17. Keahey K and Freeman T. Contextualization: Providing One-Click Virtual Clusters. eScience 2008
18. Klems M, Nimis J, and Tai S. Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. Proc. 7th Workshop of e-Business (WeB 2008). Springer LNBIP
19. Lai K, Rasmusson L, Adar E, Sorkin S, Zhang L, and Huberman BA. Tycoon: an Implementation of a Distributed Market-Based Resource Allocation System. Multiagent and Grid Systems. 2005
20. Lenk A, Sandholm T, Klems M, Nimis J, and Tai S. What's inside the Cloud? An Architectural Map of the Cloud Landscape. ICSE 2009 Workshop on Software Engineering Challenges of Cloud Computing. 2009
21. Menzel M, Schönherr M, Nimis J, and Tai S. (MC2)2: A Generic Decision-Making Framework and its Application to Cloud Computing. Proc. International Conference on Cloud Computing and Virtualization. 2010
22. Nurmi D,Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, and Zagorodnov D. The Eucalyptus Open-source Cloud-computing System. October 2008
23. Nurmi D,Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, and Zagorodnov D. Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems. August 2008
24. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, and Turner J. OpenFlow: Enabling Innovation in College Networks. 2008
25. Schäfer A. Die Kraft der schöpferischen Zerstörung. Campus. 2008
26. Sotomayor B, Montero RS, Llorente IM, and Foster I. Capacity Leasing in Cloud Systems using the OpenNebula Engine. CCA08: Cloud Computing and its Applications. 2008
27. Sprang H, Benk T, Zdrzalek J, und Dehner R. Xen. Virtualisierung unter Linux. Open Source Press. München. 2007
28. Streitberger W, Ruppel A. Cloud Computing Sicherheit - Schutzziele. Taxonomie. Marktübersicht, FhG SIT Sept. 2009
29. Varia J. Cloud Architectures, White Paper, Amazon. 2009
30. Wang L. Virtual environments for Grid computing. Universitätsverlag Karlsruhe. 2009
31. White T. Hadoop – The Definite Guide. O'Reilly Verlag. 2009
32. Wohlstadter E und Tai S. Web Services. Reference Entry. Encyclopedia of Database Systems (EDBS). Springer. 2009

Online References

33. 10gen Scalable High Performance Data Storage for Web Applications http://www.10gen.com
34. Adobe Photoshop Express https://www.photoshop.com
35. EdgePlatform http://www.akamai.com/en/html/technology/edgeplatform.html
36. Amazon CloudFront http://aws.amazon.com/cloudfront/
37. Amazon CloudWatch http://aws.amazon.com/cloudwatch/
38. Amazon Cluster Compute Instances http://aws.amazon.com/ec2/hpc-applications/
39. Amazon Elastic Block Store (EBS) http://aws.amazon.com/ebs/
40. Amazon Elastic Compute Cloud (EC2) http://aws.amazon.com/ec2/
41. Amazon Elastic Map Reduce http://aws.amazon.com/elasticmapreduce/
42. Amazon EC2 Instance Types http://aws.amazon.com/ec2/instance-types/
43. Amazon Elastic Compute Cloud. Getting Started Guide http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/
44. Amazon Mechanical Turk (MTurk) http://www.mturk.com
45. Amazon Relational Database Service http://aws.amazon.com/rds/
46. Amazon Simple Storage Service (S3) http://aws.amazon.com/s3/

47. Amazon SimpleDB http://aws.amazon.com/simpledb/
48. Amazon Simple Queue Service (SQS) http://aws.amazon.com/sqs/
49. Amazon Virtual Private Cloud (VPC) http://aws.amazon.com/vpc/
50. Amazon Web Services (AWS) http://aws.amazon.com
51. Amazon Web Services Developer Tools http://aws.amazon.com/developertools/
52. Amazon Web Services Import/Export http://aws.amazon.com/importexport/
53. Amazon Web Services Management Console http://console.aws.amazon.com
54. Amazon Web Services Toolkit for Eclipse http://aws.amazon.com/eclipse/
55. Animoto http://animoto.com
56. AppExchange http://sites.force.com/appexchange/home
57. AppNexus Cloud http://www.appnexus.com
58. Chandra Krintz: Appscale – An open-source research framework for execution of Google AppEngine applications and investigation of scalable cloud computing fabrics http://www.cs.ucsb.edu/ ckrintz/abstracts/appscale.html
59. Azure Services Platform http://www.microsoft.com/azure/
60. Bluelock http://www.bluelock.com
61. Project Caroline http://research.sun.com/projects/caroline
62. The Future of Corporate IT, Corporate Executive Board, 2010 http://www.executiveboard.com/it/pdf/The_Future_of_Corporate_IT.pdf
63. Cloud Hosting and Cloud Storage Performance Dashboard http://www.cloudclimate.com
64. Cloudera http://www.cloudera.com
65. CloudStack http://www.cloud.com
66. Security Guidance for Critical Areas of Focus in Cloud Computing, Cloud Security Alliance, 2009 http://www.cloudsecurityalliance.org/guidance/csaguide.pdf
67. DeveloperForce http://developer.force.com
68. Django Web Framework http://www.djangoproject.com
69. Dropbox https://www.dropbox.com
70. Eclipse http://www.eclipse.org
71. eyeOS http://www.eyeos.org
72. ElasticFox: Firefox Extension for Amazon EC2 http://sourceforge.net/projects/elasticfox/
73. Virtual Private Datacenters http://www.enkiconsulting.net/virtual-private-data-centers/
74. Eucalyptus http://open.eucalyptus.com
75. Facebook Platform http://developers.facebook.com
76. FlexiScale Cloud Computing http://www.flexiscale.com
77. fluidOps eCloudManager http://www.fluidops.com/ecloudmanager.html
78. Force.com http://www.salesforce.com/de/platform/
79. Gaikai http://www.gaikai.com
80. Ganglia Monitoring System http://ganglia.info
81. gEclipse http://www.geclipse.org
82. GoGrid Cloud Hosting http://www.gogrid.com
83. Google App Engine http://appengine.google.com
84. Google App Engine SDK http://code.google.com/intl/de/appengine/downloads.html
85. Google Apps http://www.google.com/apps/
86. Google Chrome OS http://www.chromium.org/chromium-os
87. Google Cloud Print http://code.google.com/apis/cloudprint/
88. Google Docs http://docs.google.com
89. The Google File System http://labs.google.com/papers/gfs.html
90. Google Maps API http://code.google.com/apis/maps/
91. Google Eclipse Plugin http://code.google.com/intl/de/appengine/docs/java/tools/eclipse.html
92. Google Storage http://code.google.com/apis/storage/
93. Gridcore Gompute http://www.gridcore.se/ondemand-com/
94. GSUtil http://code.google.com/intl/en/apis/storage/docs/gsutil.html
95. Projekt Gutenberg http://www.gutenberg.org

96. Guardian: Investigate your MP's expenses http://mps-expenses.guardian.co.uk
97. Hadoop Homepage http://hadoop.apache.org
98. Hive! http://hadoop.apache.org/hive/
99. Joyent Reasonably Smart http://www.joyent.com
100. JungleDisk http://www.jungledisk.com
101. KOALA http://koalacloud.appspot.com
102. KOALA Cloud Manager http://code.google.com/p/koalacloud/
103. Microsoft Windows Live http://explore.live.com/
104. Microsoft – The Economics of the Cloud http://www.microsoft.com/presspass/presskits/cloud/docs/The-Economics-of-the-Cloud.pdf
105. NIST: Cloud Computing: Cloud Computing http://csrc.nist.gov/groups/SNS/cloud-computing/
106. NetSuite SuiteFlex http://www.netsuite.com
107. New York Times Archives + Amazon Web Services = TimesMachine http://open.blogs.nytimes.com/2008/05/21/the-new-york-timesarchives-amazon-web-services-timesmachine/
108. Nimbus http://www.nimbusproject.org
109. Nirvanix Storage Delivery Network http://www.nirvanix.com
110. OnLive – Cloud Gaming Service http://www.onlive.com
111. OpenCirrus http://opencirrus.org
112. OpenCirrus (Global) Monitoring https://opencirrus.org/content/global-monitoring/
113. OpenId http://www.openid.net
114. OpenNebula http://www.opennebula.org
115. OpenSocial API http://code.google.com/apis/opensocial/
116. OpenSSH http://www.openssh.com
117. OpenStack http://www.openstack.org/
118. Otoy http://www.otoy.com
119. Penguin Computing on Demand http://www.penguincomputing.com
120. Rackspace Managed Hosting http://www.rackspace.com
121. Michal Ludvig, s3cmd: S3-Client für die Kommandozeile http://s3tools.org/s3cmd/
122. S3Fox Organizer(S3Fox) http://www.s3fox.net
123. Sabalcore HPC on Demand http://www.sabalcore.com
124. Salesforce Community http://www.salesforce.com/community/
125. Salesforce CRM http://www.salesforce.com/de/crm/
126. Thomas Sandholm, Hadoop User Group Presentation http://tycoon.hpl.hp.com/ tycoon/grid/HadoopUserGroupSept08.ppt
127. Skytab Virtual Lab http://www.skytap.com
128. SLA@SOI Empowering the Service Industry with SLA-aware Infrastructures http://sla-at-soi.eu
129. Tashi Cloud Computing on Big Data http://www.pittsburgh.intel-research.net/projects/tashi/
130. PCI DSS https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf
131. Terremark Infinistructure http://www.terremark.com/services/managed-hosting.aspx
132. todo flexIT http://www.todo.de/Produkte/flexIT.php
133. TyphoonAE http://code.google.com/p/typhoonae/
134. UNIVA UD UniCloud http://www.univaud.com
135. VMware http://www.vmware.com
136. VMware vSphere http://www.vmware.com/products/vsphere/
137. Web Services Architecture Requirements. W3C Working Draft 29 April 2002 http://www.w3.org/TR/2002/WD-wsa-reqs-20020429
138. World Privacy Forum: Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing http://http://www.worldprivacyforum.org
139. Ylastic http://www.ylastic.com
140. YouTube http://www.youtube.com
141. Zimory Public Cloud Marketplace http://www.zimory.de
142. Zoho Creator http://www.zoho.com
143. ZumoDrive – Hybrid Cloud Storage http://zumodrive.com

Index

A
Accountability, 46, 48
Accounting, 22, 23, 26, 27, 31, 33–35, 40, 49, 65, 70
Amazon Web Services (AWS), 8, 23–34, 41, 43–45, 53, 54, 59–61
  Amazon Machine Image (AMI), 25, 45, 60
  availability zone, 25
  CloudFront, 19, 24, 41
  CloudWatch, 31, 41
  Cluster Compute Instances, 22
  control, 41, 44
  Dynamo, 19
  EC2, 18, 19, 23–28, 41, 44, 45, 51, 53, 54, 59, 73–75, 79
  EC2 Compute Unit (ECU), 53, 54
  Elastic Block Store (EBS), 24, 26, 29, 44, 51, 52, 55, 75–76
  Elastic MapReduce, 24, 41, 59, 60, 82–83
  Import/Export, 24
  job flow, 82
  Machine Image, 80
  Management Console, 27, 31, 41, 43
  Mechanical Turk, 22
  Relational Database Service (RDS), 24, 30–31, 41, 76–77
  security group, 45
  SimpleDB, 19, 24, 30
  Simple Queue Service (SQS), 19, 24, 29–30
  Simple Storage Service (S3), 18, 19, 24, 26–29, 34, 41, 42, 51, 52, 77, 82
  spot instances, 28
  VPC, 24, 41
Animoto, 63–64
AppNexus, 19
AppScale, 20, 34, 44, 56, 79
Auditing, 47
Authenticity, 46
Automation, 6, 34, 35, 40, 82
Availability, 6, 9, 10, 25, 29, 31, 36, 38, 39, 42, 46, 51, 70, 74, 75, 79, 80
AWS. *See* Amazon Web Services
Azure, 20, 36

B
Billing, 3, 4, 26, 28, 32, 39, 40, 47, 49, 64
Bioinformatics, 56
Bucket, 28, 34, 77, 82–83
Business models, 4, 66–68

C
Case-law, 48
Cloud computing, 1–2
  adoption, 70
  architecture, 15–22, 40, 66–67
  basic technologies, 4
  billing models, 64
  business models, 66–68
  cloud bursting, 62
  Cloudera, 57, 60
  CloudFront, 19, 24
  cloud gaming, 38
  Cloud Print, 35
  Cloud Security Alliance, 47
  cloud sourcing, 46, 47
  CloudStack, 19, 40, 49–62
  cost models, 65–66
  Cumulus, 55
  customer relationship management (CRM), 20, 21, 37

Cloud computing (*cont.*)
data center, 2, 6, 9, 19, 22–24, 34, 36, 46, 47, 49–51, 65, 70
definition, 3
development, 20, 23, 27, 31, 33–37, 45–46, 51, 61, 62, 69, 70, 78
economic considerations, 4, 63–68
evaluation, 4, 64–66, 70
external cloud, 15
fields of application, 63–64
growth, 70
hybrid cloud, 15–16, 19, 24, 36
layers, 6, 8, 9, 11, 17, 18, 20–22, 40, 49, 61
legal compliance, 48
management, 2–4, 6, 7, 9, 10, 18, 21, 27, 31, 33–35, 37, 39–51, 55, 61, 62, 64, 66–67
management services, 41–46
market development, 69
marketplace, 22, 37, 46
monitoring, 31, 32, 39–41, 47, 49, 50, 61, 66–67, 81
more cloud services, 24–25, 64
offerings, 1, 2, 4, 9, 15, 16, 18–38, 40, 44, 48, 51, 64–66, 69, 70, 73
operating systems, 7, 8, 10, 18, 25, 34, 35, 38, 56, 78, 80, 81
opportunities, 69–71
private cloud, 2, 15, 16, 34, 42–44, 51, 53–56, 66, 69, 79, 80
programming, 9, 11, 14, 18, 20, 21, 29, 31, 33, 37, 41, 45, 57–59, 82
providers, 1–3, 6, 15, 20, 21, 34, 35, 37, 38, 41, 43, 47, 48, 51, 64, 66–68, 70
public cloud, 2, 10, 15, 16, 19, 42, 44, 51, 56, 63–66, 69
risk management, 47–48
risks, 44, 58, 64, 69–71
security, 2, 4, 7, 9, 14–16, 26, 39, 41, 43, 45–48, 51, 61, 65, 70, 73, 74
service consumer, 11–13, 39, 46
service provider, 5–7, 11–13, 39–41, 44, 46, 47, 55
services, 10–14, 16–35, 41–46, 59–60
stack, 17, 20, 21, 40, 49–62, 66
users, 1, 3, 10, 15–18, 20–35, 37–39, 41, 42, 44, 45, 52, 54, 55, 57, 59–61, 63–64, 70, 77, 78, 82
Commissioned data processing, 48
Confidentiality, 7, 46, 47

D
Data
analysis, 56, 57, 59
mining, 56, 82–83
privacy, 4, 24, 43, 46, 48, 70
Disruptive technology, 69
Django Framework, 18
Dropbox, 18, 19

E
EC2. *See* Elastic Compute Cloud
EC2 command line tools, 42, 73, 75, 79
EE. *See* Execution environment
Elastic Block Store (EBS) volume, 26, 29, 75
Elastic Compute Cloud (EC2), 18, 19, 23–32, 34, 41, 42, 44, 45, 51, 53–56, 59, 73–75, 79, 82–83
ElasticFox, 27, 42–44, 82
Elasticity, 3, 64–65
Emulab, 18, 19, 49
Encryption, 24, 47, 48, 70
Ensembles, 40, 62
Enterprise Service Bus (ESB), 12
Eucalyptus, 4, 18–20, 42–44, 51, 54, 55, 69
architecture, 51–54
Cloud Controller (CLC), 51, 52
Cluster Controller (CC), 51, 52
components, 51–54
Euca2ools, 42, 44, 79
file system image, 81
how to use, 79–82
infrastructure, 51, 52
installation, 79–82
instance number, 81
instances, 51, 53, 54, 80, 81
kernel image, 80, 81
key pair, 80
Node Controller (NC), 51, 52, 79
RAM disk image, 80, 81
resources, 51–53, 79–81
Storage Controller (SC), 52
Walrus, 52
Euca2ools, 42, 44, 79
Everything as a Service (XaaS), 17–18
Execution environment (EE), 18, 20, 33
Extensible Markup Language (XML), 12–14, 32, 61
External cloud, 2, 10, 15, 16, 19, 42, 44, 51, 56, 63–66, 69
eyeOS, 38

F
Facebook, 20, 64
Fault tolerance, 40, 58
Financial analyses, 56
FlexiScale, 19
Fluid Operations, 21, 22
Force.com, 20, 37

G
Gaikai, 38
Ganglia, 61
GoGrid, 19
Google
App Engine, 20, 33–34, 44, 46, 47, 51, 56, 61, 77–78
Docs, 21, 33
Maps, 21
Storage, 34–35, 44
Google Web Toolkit (GWT), 34, 45
GrepTheWeb, 31, 32
Grid computing, 3
Gridcore Gompute, 22
GSUtil, 34, 44
GWT. *See* Google Web Toolkit

H
Hadoop, 4, 18, 32, 56–61, 69
Hadoop as a Service, 59–60
Hadoop Distributed File System (HDFS), 57–59
HDFS. *See* Hadoop Distributed File System
High Performance Computing as a Service (HPCaaS), 22, 27, 54, 62
Hive, 59
HPCaaS. *See* High Performance Computing as a Service
HTTP. *See* Hypertext Transfer Protocol
HuaaS. *See* Humans as a Service
Hub-and-spoke approach, 12
Humans as a Service (HuaaS), 17–18, 21–22
Hybrid cloud, 15–16, 19, 24, 36
Hybridfox, 43–44
Hyper call, 8
Hypertext Transfer Protocol (HTTP), 13, 14, 32, 34
Hyper-V, 56
Hypervisor, 8, 18, 51, 54–56

I
IaaS. *See* Infrastructure as a Service
Identity management, 21, 61
iLO. *See* Integrated Lights-Out
Infiniband, 62
Information technology (IT) service center, 2
Infrastructure as a Service (IaaS), 17–20, 22, 34, 46, 47, 49, 51, 54–56, 66–67
Integrated Lights-Out (iLO), 18, 19
Integrity, 46, 58
IOWA Electronic Markets, 22
ISO 27001, 47

J
Jails, 7
JungleDisk, 28

K
Kernel-based Virtual Machine (KVM), 8, 51, 54, 55
Knowledge worker, 2
KOALA, 44
KVM. *See* Kernel-based Virtual Machine

L
LaaS. *See* Landscape as a Service
Landscape as a Service (LaaS), 19, 22
Lifecycle, 9, 26, 40
Linux-VServer, 7
Logfile analyses, 56

M
Machine learning, 56, 70
Map function, 57
MapReduce, 18, 24, 41, 57–60, 82–83
Mash-up, 21
Message Passing Interface (MPI), 62
Monitoring, 8, 31, 32, 39–41, 47, 49, 50, 61–62, 66–67, 81
MPI. *See* Message Passing Interface

N
New York Times, 63
Nimbus, 18, 19, 44, 54–55

O
OLA. *See* Operation level agreement
On-demand solutions, 68
OnLive, 38
On-premise solutions, 68
OpenCirrus^{TM}, 4, 61, 62

OpenID, 21
OpenNebula, 18, 19, 22, 44, 54, 55
OpenSocial, 21
Open source cloud stack, 49–62
OpenStack, 55–56
  Compute, 55–56
  Object Storage, 55–56
OpenVZ, 7
Operating system kernel, 7, 8
Operation level agreement (OLA), 39
Otoy, 38
Overprovisioning, 64, 65

P
PaaS. *See* Platform as a Service
Pay as you go, 64
PE. *See* Programming environment
Penguin Computing on Demand, 22
Personal computer, 69
Physical resource set (PRS), 18, 49, 50, 61
Pig, 59
Pirated copies, 38
Platform as a Service (PaaS), 17–22, 30, 31, 33, 34, 37, 46, 47, 49, 51, 68, 79
Private cloud, 2, 15, 16, 34, 42–44, 51, 53–56, 65–66, 69, 79, 80
Programming environment (PE), 18, 20, 33, 37, 45, 59
PRS. *See* Physical resource set
Pseudonymity, 46
Public cloud, 2, 10, 15, 16, 19, 39, 42, 44, 51, 56, 63–66, 69

Q
Quality of service, 4, 9, 13, 34, 39–41, 69

R
Real estate costs, 65
Reduce function, 57, 83
Relational Database Service (RDS) command line tools, 76
Resource set, 18
REST, 13, 14, 29, 34, 52, 55

S
SaaS. *See* Software as a Service
Sabalcore HPC on Demand, 22
Safe Harbor, 48
Salesforce.com, 21, 37
SAS, 47
Saturation, 64
Scalability, 9, 39, 40, 56, 57, 70
Scheduling, 40, 51, 52, 54, 62, 65
Scientific simulations, 56
s3cmd, 29, 42, 44, 77
Security techniques, 61
Service, 1, 5, 15, 23, 39, 49, 63, 69, 76
  catalog, 40
  level management, 39
  lifecycle, 40
  monitoring, 39–41, 50, 61
Service level agreement (SLA), 6, 39–40, 46, 47, 62
Service-oriented architectures (SOA), 5, 10–13
S3Fox, 42–44, 82–83
Simple Object Access Protocol (SOAP), 12–14, 29, 54
  body, 14
  envelope, 14
  header, 14
  message, 13, 14
Single sign-on, 61
SLA. *See* Service level agreement
Snapshots, 9, 17, 26, 31, 41, 75–77
SOA. *See* Service-oriented architectures
SOAP. *See* Simple Object Access Protocol
Social media, 2
Software as a Service (SaaS), 10, 17–18, 20–22, 37, 44, 46, 47, 49, 68
Software licenses, 62, 70
Software-Plus-Services, 68
Staff costs, 65
Standardization, 1, 3, 10, 14, 33, 35, 48, 61, 70
Storage service, 29, 34, 36, 42, 43, 50, 52, 54, 55, 61
System call, 8, 57

T
Tashi, 50–51, 61
TCO. *See* Total cost of ownership
TimesMachine, 63, 64
Topic, 3, 4, 47, 48, 71
Total cost of ownership (TCO), 65–67
Tycoon, 19
TyphoonAE, 34, 44, 56

U
UDDI. *See* Universal Description, Discovery and Integration
Underprovisioning, 64

Underutilization, 64, 65
Uniform resource identifier (URI), 12, 13
UNIVA DU UniCloud, 22
Universal Description, Discovery and Integration (UDDI), 11–12
URI. *See* Uniform resource identifier
Utility computing, 1, 51

V

VDE. *See* Virtual Distributed Ethernet
Vendor lock-in, 2, 20, 47, 70
Virtual Distributed Ethernet (VDE), 52–53
Virtualization, 3–10, 18, 51, 54–56, 62
  application virtualization, 10
  benefits, 5–7
  container, 7
  drawbacks, 5–7
  hyper call, 8
  hypervisor, 8, 54, 55
  network virtualization, 8–10
  operating system virtualization, 7
  paravirtualization, 8
  platform virtualization, 8
  storage virtualization, 8–9
Virtual machine, 5–8, 36, 49–54, 56, 70, 79–81
Virtual Private Cloud (VPC), 24, 41
Virtual resource set (VRS), 18, 50, 55, 61
Virtuozzo, 7
VMware, 8, 51, 54, 55, 69
  ESX, 51
  vSphere, 51, 54, 55, 69
Volume, 24, 26, 28, 29, 36, 56, 63, 66, 70, 71, 75, 76
VPC. *See* Virtual Private Cloud
VRS. *See* Virtual resource set

W

Walrus, 52
Web
  indexing, 56
  protocols, 10, 13
  services, 4, 5, 8, 10, 12–14, 16, 23, 29, 30, 37, 41, 51
Web Service Description Language (WSDL), 12–14
Windows Azure, 20, 36
  Active Directory, 36
  AppFabric, 36
  compute, 36
  drive, 36
  queue, 36
  roles, 36
  SQL, 36
  storage, 36
WSDL. *See* Web Service Description Language

X

XaaS. *See* Everything as a Service
Xen, 8, 18, 51, 54–56
XML. *See* Extensible Markup Language

Y

Ylastic, 43, 44
YouTube, 22

Z

Zimory, 19, 22, 40, 46
Zumodrive, 18, 19