CHAPTER

The cloud ecosystem

2

The statement that cloud computing is a disruptive technology can be easily supported by the arguments presented in the final section of this chapter where cloud computing's impact on enterprise computing is quantified in billions, possibly trillions, of dollars. The argument that cloud computing is a major contributor to the large CO_2 footprint of the IT industry is more subtile,[1] as we have seen in Section 1.2.

Even more subtile is the transformative effect of cloud computing on virtually all areas of computing, including algorithms, computer software and hardware, database systems, networking, etc. This is the main theme of this text, discussed in the next eleven chapters, which track how concepts, ideas, and technologies from the precloud computing era have changed to accommodate the qualitative and quantitative challenges brought about by the cloud ecosystem on hardware, software, and applications.

A case in point is the evolution of the *load balancing* concept used for three decades or longer to suggest that, when you have a system with a large number of components, it is best to divide the load if you can and then assign each unit about the same amount of work. The new twist in cloud computing is that load balancing should attempt to minimize energy consumption, and this means concentrating the workload on the smallest number of servers, while switching the others to an energy-saving mode.

Previously, computer architecture supported only the privileged kernel mode and the non-privileged user mode, but system resources in a virtual environment are managed by a hypervisor rather than one operating system. Consequently, the hardware needs to support different levels of protection. Moreover, the hardware must provide a trusted execution mode because clouds are target-rich environments.

The *scale of the cloud computing infrastructure* has profound implications on cloud resource management forced to satisfy conflicting requirements; on one hand, the system must be elastic, i.e., able to accommodate sudden spikes of workload and, on the other hand, minimize energy consumption. All these complications must be hidden from users who should be presented with simple and easy-to-use interfaces for low-cost, high-level services. This is the subject of this introductory chapter which presents the cloud ecosystem as of mid 2021.

The major players in this ecosystem are Amazon, Google, and Microsoft and now IBM, which is poised to challenge the dominance of the first three on this list. The chapter starts with an overview of the cloud ecosystem and an in-depth discussion of cloud delivery models and services in Section 2.1, followed by the analysis of cloud computing at Amazon, Google, Microsoft, and IBM in Sections 2.2, 2.3, 2.4, and 2.5, respectively. Sections 2.6 and 2.7 cover the pervasive issue of vendor lock-in and the prospects of cloud interoperability. The presentation of Service Level Agreements (SLAs) in Section 2.8 is followed by a discussion of responsibility sharing between cloud users and cloud service providers in

[1] The IT industry may one day end up competing with the transportation industry as far as its CO_2 footprint is concerned, unless the Bitcoint craze is stopped or at least limited, see Section 3.18.

Cloud Computing. https://doi.org/10.1016/B978-0-32-385277-7.00009-9

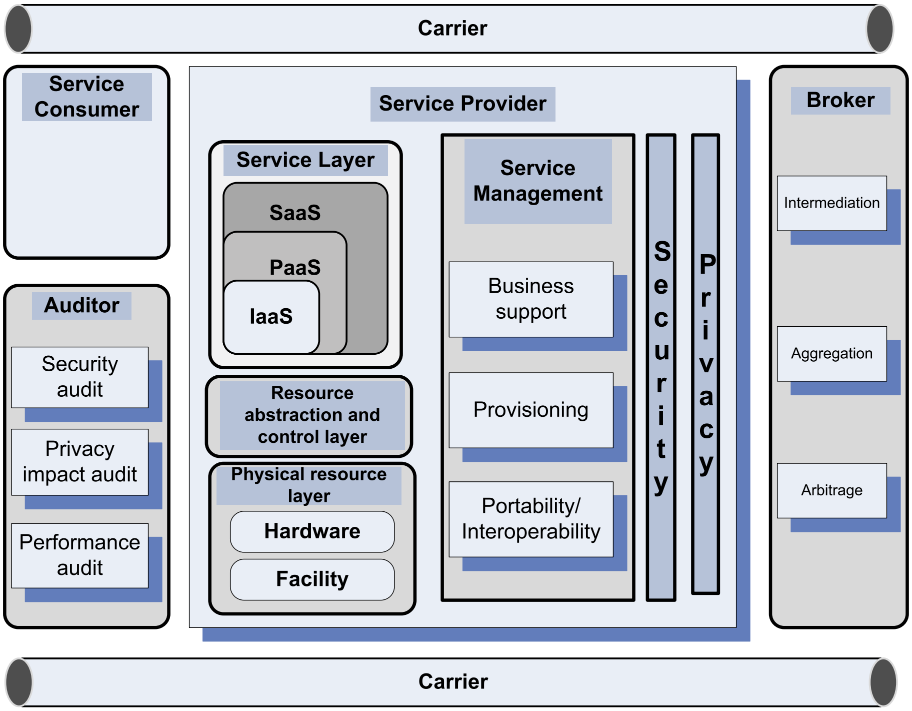

FIGURE 2.1

Entities involved in service-oriented computing and, in particular, in cloud computing according to NIST. The carrier provides connectivity between service providers, service consumers, brokers, and auditors.

Section 2.9. User experience is analyzed in Section 2.10, while Section 2.11 covers software licensing. Major challenges faced by cloud computing are discussed in Section 2.12. The cloud is changing the enterprise ecosystem at a stunning speed as presented in Section 2.13.

2.1 Cloud computing delivery models and services

According to NIST reference model in Fig. 2.1 [366], the entities involved in cloud computing are: *service consumer*—maintains a business relationship with, and uses service from, service providers; *service provider*—responsible for making a service available to service consumers; *carrier*—intermediary providing connectivity and transport of cloud services between providers and consumers; *broker*—manages the use, performance, and delivery of cloud services, and negotiates relationships between providers and consumers; *auditor*—a party that can conduct independent assessment of cloud services, information system operations, performance, and security of the cloud implementation.

An *audit* is a systematic evaluation of a cloud system that measures how well it conforms to a set of established criteria. For example, a security audit evaluates cloud security, and a privacy-impact audit evaluates cloud privacy assurance, while a performance audit evaluates cloud performance. Fig. 2.1 shows activities supporting cloud delivery models: (i) service management and provisioning including: virtualization, service provisioning, the call center, operations management, systems management, QoS management, billing, accounting, asset management, SLA management, technical support, and backups; (ii) security management including: ID and authentication, certification and accreditation, intrusion prevention, intrusion detection, virus protection, cryptography, physical security, incident response, access control, audit and trails, and firewalls; (iii) customer services such as: customer assistance and on-line help, subscriptions, business intelligence, reporting, customer preferences, and personalization; and (iv) integration services including data management and development.

Cloud service providers (CSPs) support one or more cloud delivery models: SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service), and DBaaS (Database as a Service). SaaS is a centrally hosted service based on a subscription model for licensed software. PaaS provides a computing platform and applications, without the need to manage the infrastructure required to build and run the applications. IaaS is the model where the CSP provides a menu of resource bundles and the users can choose the operating system and manage the resources in the bundle. DBaaS enables users to setup and operate several databases using a common set of abstractions, without the need to know the exact implementations of those abstractions for individual databases. Amazon is a pioneer in IaaS, Google's efforts are focused on SaaS and PaaS delivery models, and Microsoft is mostly involved in PaaS. Amazon, Oracle, and many other CSPs offer DBaaS services. Oracle Cloud is based on Java, SQL standards, and software systems such as Exadata, Exalogic, WebLogic, and Oracle Database.

Software as a Service. A wide range of stationary and mobile devices enable a large population of clients to access services provided by the applications using a thin client interface such as a web browser (e.g., web-based email). The users of services do not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. Examples of SaaS include: enterprise services, such as workflow management, group-ware and collaborative, supply chain, communications, digital signature, customer relationship management (CR), desktop software, financial management, geo-spatial, and search.

The SaaS delivery model is not suitable for applications requiring real-time response or those in which data is not allowed to be hosted externally. The most likely candidates for *SaaS* are applications with these characteristics: (a) Many competitors use the same product, such as Email; (b) there is a significant peak in demand periodically, such as in billing and payroll applications; (c) there is a need for the web or mobile access, such as mobile sales-management software; (d) there is only a short-term need, such as collaborative software for a project.

Platform as a Service. PaaS offers the capability to deploy consumer-created or acquired applications using programming languages and tools supported by the provider. The user does not manage or control the underlying cloud infrastructure including the network, servers, operating systems, or storage. The user has control over the deployed applications and, possibly, applications hosting environment configurations. Such services include: session management, device integration, sandboxes, instrumentation and testing, contents management, knowledge management, and Universal Description, Discovery, and Integration (UDDI), a platform-independent, Extensible Markup Language (XML)-based registry providing a mechanism to register and locate web-service applications.

PaaS is not particularly useful when the application must be portable, when proprietary programming languages are used, or when the underlaying hardware and software must be customized to improve the performance of the application. Its major application areas are in software development when multiple developers and users collaborate and the deployment and testing services should be automated.

Infrastructure as a Service. IaaS has the capability to provide processing, storage, networks, and other fundamental computing resources; the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of some networking components, e.g., host firewalls. Services offered by this delivery model include: server hosting, web servers, storage, computing hardware, operating systems, virtual instances, load balancing, Internet access, and bandwidth provisioning.

The IaaS cloud computing delivery model has a number of characteristics: The resources are distributed and support dynamic scaling; it is based on a utility pricing model and variable cost; and the hardware is shared among multiple users. This cloud delivery model is particularly useful when the demand is volatile, a new business needs computing resources, and it does not want to invest in a computing infrastructure, or when an organization is expanding rapidly.

Fig. 2.2 shows the degrees of freedom of cloud users and the interaction with cloud infrastructure, which vary from extremely limited for SaaS, to modest for PaaS, and significant for IaaS [122]. Table 2.1 shows that SaaS cloud service providers control not only the cloud infrastructure and the hypervisor, but also the network traffic, the operating system, and the applications. Iaas CSPs only provide the cloud infrastructure and the hypervisor.

Database as a Service. DBaaS is a cloud service where the database runs on the service provider physical infrastructure. Compared with on-site physical server and storage architecture, a cloud database service offers distinct advantages: instantaneous scalability, performance guarantees, specialized expertise, latest technology, failover support, and declining pricing. Most relevant features of the DBaaS model are:

1. Self-service—service provisioning without major deployment or configuration and without performance and cost penalties.
2. Device and location-independent abstract database resources without concern for hardware utilization.
3. Elasticity and scalability—automated and dynamic scaling.
4. Pay-as-you-go model—metered use of resources and cost reflecting the resources used.
5. Agility—the applications adapt seamlessly to new technology or additional requirements.

Cloud DBaaS uses a layered architecture. The *user interface layer* supports access to the service via the Internet. The *application layer* is used to access software services and storage space. The *database layer* provides efficient and reliable database service; it saves time for querying and loading data by reusing the query statements residing in the storage. *Data storage layer* encrypts the data when stored without user involvement; backup management and disk monitoring are also provided by this layer. Multi-tenancy is an integral part of the DBaaS model. In spite of its advantages, multi-tenancy poses resource management and security challenges, as discussed in Section 8.7.

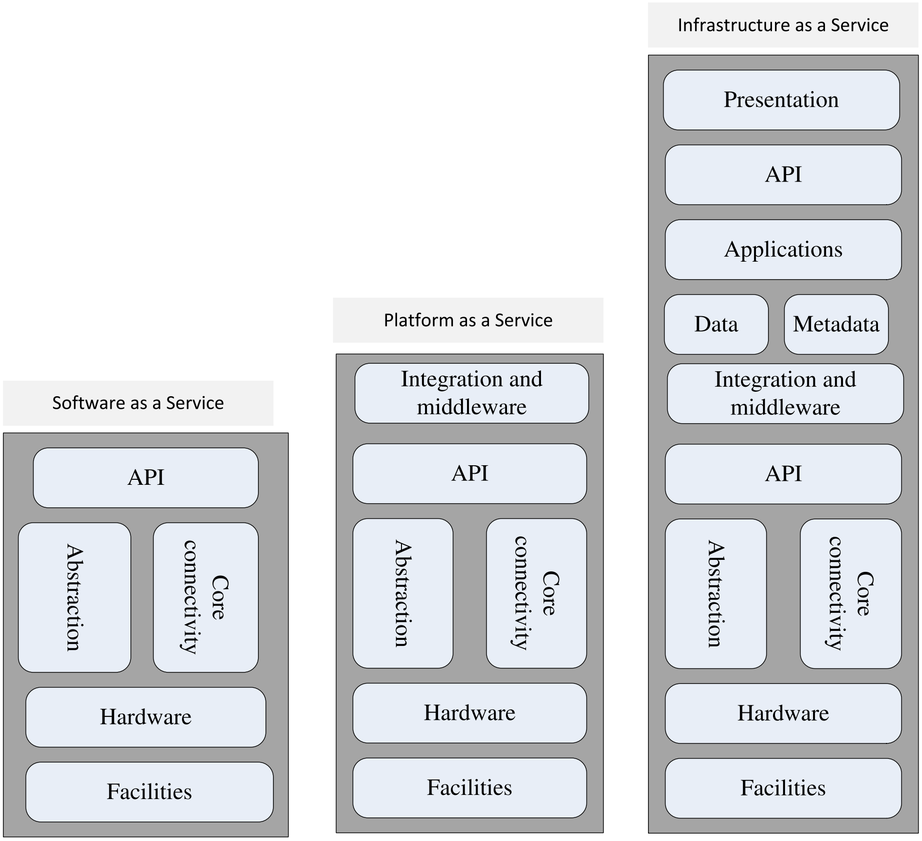

FIGURE 2.2

Software and cloud infrastructure control versus ease of use of the service. SaaS allows no infrastructure control and does not require any sophistication to use applications supplied by CSP. PaaS supports consumer-created or acquired applications using CSP-provided tools. IaaS empowers advanced developers and uses to chose the operating system, run arbitrary software, choose the type of instances, and manage instance resources.

Table 2.1 The shared security responsibility (SSR) model between cloud service providers (in bold) and cloud users for the three cloud delivery models.

SaaS	PaaS	IaaS
User Access	User Access	User Access
Data	Data	Data
Applications	Applications	Applications
Operating Systems	**Operating Systems**	Operating Systems
Network Traffic	**Network Traffic**	Network Traffic
Hypervisor	**Hypervisor**	**Hypervisor**
Infrastructure	**Infrastructure**	**Infrastructure**

This discussion shows that a service-oriented architecture involves multiple subsystems and complex interactions among these subsystems. Individual subsystems can be layered; for example, we see that the service layer sits on top of a resource abstraction layer that controls the physical resource layer.

Cloud delivery models will continue to coexist for the foreseeable future. Services based on SaaS will probably be increasingly more popular because they are more accessible to lay people, while services based on the IaaS will be the domain of computer-savvy individuals, large organizations, and the government. If the standardization effort succeeds, then we may see IaaS designed to migrate from one infrastructure to another and overcome the concerns related to vendor lock-in. The popularity of DBaaS services is likely to grow.

Infrastructure as Code (IaC). It is always beneficial to automate provisioning of cloud infrastructure, i.e., manage servers, operating systems, database connections, storage, and other infrastructure elements using a high-level descriptive language. This is precisely the function of IaC that takes advantage of virtualization and cloud native development allowing developers to provision their own virtual servers or containers on demand. A cloud native application consists of discrete, reusable components, *microservices*, acting as building blocks, often packaged in containers, and designed to integrate into any cloud environment. Microservices can be independently scaled, continuously improved, and quickly iterated through automation and orchestration processes.

IaC advantages are obvious, namely, improved consistency, more efficient development, lower costs, shorter time to production, and a more secure environment. For the functional or declarative IaC approach, a skilled administrator specifies the desired final state of the infrastructure, and the IaC software handles the rest—spinning up the virtual machine (VM) or container, installing and configuring the necessary software, resolving system and software interdependencies, and managing versioning. The procedural IaC approach automates provisioning the infrastructure one step at a time.

The most used IaC tools are *Ansible* and *Terraform*. Ansible is an open-source IT automation engine that is used to improve the scalability, consistency, and reliability of the IT environment. Ansible supports: (i) provisioning, i.e., setting up the servers; (ii) configuration management, i.e., changing the configuration of an application, OS, or device, start and stop services, installing or updating applications, and implementing a security policy, of other configuration tasks; and (iii) application deployment automates the deployment of applications using DevOps, a philosophy of efficient software development, deployment, and operation.

Terraform automates resource management across multiple providers, regardless of where physical servers, DNS servers, or databases reside and provisions applications written in any language. The first step in *Terraform* is to create an execution plan; then, the system generates a graph of all resources and parallelizes creation and modification of any non-dependent resources. A number of use cases are described on https://www.terraform.io/intro/use-cases.html including multi-tier applications, self-service clusters, SDN (Software Defined Networks), resource schedulers, and multi-cloud environments.

2.2 Amazon Web Services

Amazon has changed the face of computing over recent decades. First, it installed a powerful computing infrastructure to sustain its core business, selling online a variety of goods ranging from books and CDs to gourmet foods and home appliances. Then, the company discovered that this infrastructure can be further extended to provide affordable and easy-to-use resources for enterprise computing, as well as computing for the masses.

Amazon Web Services (AWS), based on the IaaS delivery model, offers an infrastructure consisting of computation and storage servers interconnected by high-speed networks and supports a set of services to access these resources. Businesses in more than 200 countries use AWS. A significant number of corporations and start-ups take advantage of Amazon cloud computing infrastructure.

AWS computing, storage, and communication services. Amazon announced a limited public beta release of its Elastic Computing platform called EC2 in 2006. AWS offered 24 services in 2008, 48 in 2009, 61 in 2010, 82 in 2011, 159 in 2012, and 280 in 2013; 449 new services and major features were released in 2014. AWS services reflect leading-edge technological developments: For example, machine learning and quantum simulation services are now supported, see Sections 13.2 and 13.3.

Elastic Compute Cloud (EC2) is a web service with a simple interface for launching instances of an application under several operating systems, such as several Linux distributions, OpenSolaris, FreeBSD, NetBSD, and Microsoft Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, and 2019. EC2 is based on the virtualization strategy discussed in detail in Chapter 5. For more than ten years AWS used the Xen hypervisor discussed in Section 5.8 and in 2017 replaced it with KVM discussed in Section 5.7.

An example of EC2 is a virtual server; the user chooses the region and the availability zone where this virtual server should be placed and also selects from a menu of instance types the one that provides the resources, CPU cycles, main memory, secondary storage, communication, and I/O bandwidth needed by the application.

When launched, an instance is provided with a *DNS name*; this name maps to a *private IP address* for internal communication within the internal EC2 communication network and a *public IP address* for communication outside the internal Amazon network, e.g., for communication with the user that launched the instance. Network Address Translation (NAT) maps external IP addresses to internal ones.

The public IP address is assigned for the lifetime of an instance and is returned to the pool of available public IP addresses when the instance is either stopped or terminated. An instance can request an *elastic IP address*, rather than a public IP address. The elastic IP address is a static public IP address allocated to an instance from the available pool of the availability zone; an elastic IP address is not released when the instance is stopped or terminated and must be released when no longer needed.

An instance is created either from a predefined Amazon Machine Image (AMI) digitally signed and stored in S3, or from a user-defined image. The image includes the operating system, the run-time environment, the libraries, and the application desired by the user. AMI images create an exact copy of the original image but without configuration-dependent information such as *hostname* or MAC address. A user can: (i) launch an instance from an existing AMI and terminate an instance; (ii) start and stop an instance; (iii) create a new image; (iv) add tags to identify an image; and (v) reboot an instance.

An instance specifies the maximum amount of resources available to an application and the interface for that instance, as well as the cost per hour. A server may run multiple virtual machines or instances started by one or more users; an instance may use storage services, S3, EBS, and Simple DB, as well as other services provided by AWS, see Fig. 2.3.

A user can interact with EC2 using a set of SOAP messages, see Section 11.1, and can list available AMI images, boot an instance from an image, terminate an image, display the running instances of a user, display console output, etc. The user has root access to each instance in the elastic and secure computing environment of EC2. The instances can be placed in multiple locations in different regions and availability zones.

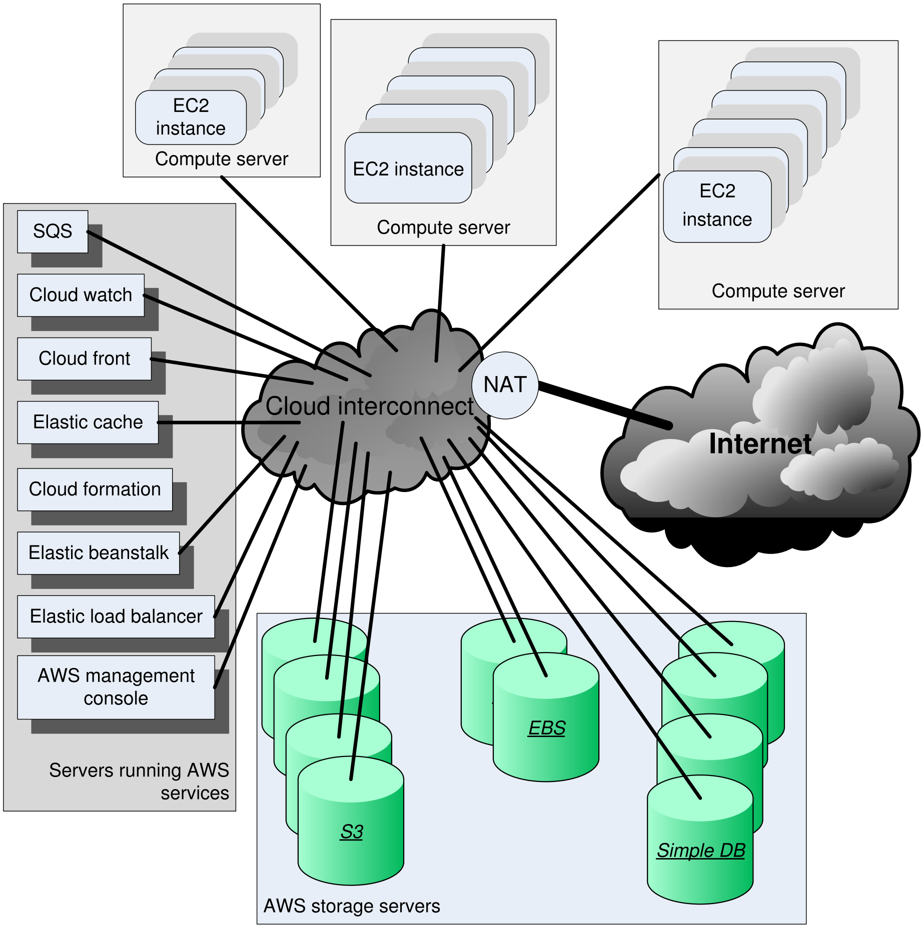

FIGURE 2.3

Availability zone—a cloud interconnect supports high-speed communication among compute and storage servers in the zone. Communication with servers in other availably zones and with cloud users is supported via a Network Address Translation (NAT), which maps external IP addresses to internal ones.

EC2 allows the import of Virtual Machine (VM) images from the user environment to an instance through a facility called *VM import*. It also distributes automatically the incoming application traffic among multiple instances using the *elastic load balancing* facility. EC2 associates an *elastic IP address* with an account; this mechanism allows a user to mask the failure of an instance and re-map a public IP address to any instance of the account, without the need to interact with the software support team.

Simple Storage System (S3) is a storage service designed to store large objects. It supports a minimal set of functions: write, read, and delete. S3 allows an application to handle an unlimited number of objects ranging in size from one byte to five terabytes. An object is stored in a *bucket* and retrieved via a unique developer-assigned key; a bucket can be stored in a region selected by the user.

S3 maintains for each object: the name, modification time, an access control list, and up to four kilobytes of user-defined metadata; object names are global. Authentication mechanisms ensure that data is kept secure; objects can be made public, and rights can be granted to other users. S3 supports `PUT`, `GET`, and `DELETE` primitives to manipulate objects, but it does not support primitives to copy, rename, or move an object from one bucket to another. Appending to an object requires a read followed by a write of the entire object.

S3 computes MD5[2] for every object written and returns it in a field called *ETag*. A user is expected to compute the MD5 of an object stored or written and compare this with the *ETag*; if the two values do not match, then the object was corrupted during transmission or storage. The S3 SLA guarantees reliability. S3 uses standards-based REST and SOAP interfaces, see Section 11.1; the default download protocol is HTTP, but BitTorrent[3] protocol interface is also provided at lower cost for large-scale distributions.

Elastic Block Store (EBS) provides persistent block-level storage volumes for use with EC2 instances. A volume appears to an application as a raw, unformatted, and reliable physical disk; the size of the storage volumes ranges from one gigabyte to one terabyte. The volumes are grouped together in availability zones and are automatically replicated in each zone. An EC2 instance may mount multiple volumes, but a volume cannot be shared among multiple instances. EBS supports the creation of snapshots of the volumes attached to an instance and then uses them to restart an instance. The storage strategy provided by EBS is suitable for database applications, file systems, and applications using raw data devices.

Simple DB is a non-relational data store that allows developers to store and query data items via web services requests; it supports store and query functions traditionally provided only by relational databases. Simple DB creates multiple geographically distributed copies of each data item and supports high-performance web applications; at the same time, it manages automatically the infrastructure provisioning, hardware and software maintenance, replication and indexing of data items, and performance tuning.

Simple Queue Service (SQS) is a hosted message queue. SQS is a system for supporting automated workflows; it allows multiple EC2 instances to coordinate their activities by sending and receiving SQS messages. Any computer connected to the Internet can add or read messages without any installed software or special firewall configurations. Applications using SQS can run independently and asynchronously and do not need to be developed with the same technologies. A received message is "locked" during processing; if processing fails, the lock expires and the message is available again. The timeout for locking can be changed dynamically via the *ChangeMessageVisibility* operation. Developers can access SQS through standards-based SOAP and Query interfaces. Queues can be shared with other AWS accounts and anonymously; queue sharing can also be restricted by IP address and time of day. An example showing the use of SQS is presented in Section 11.6.

CloudWatch is a monitoring infrastructure used by application developers, users, and system administrators to collect and track metrics important for optimizing the performance of applications and for

[2] MD5 (Message-Digest Algorithm) is a widely used cryptographic hash function; it produces a 128-bit hash value and is used for checksums. SHA-i (Secure Hash Algorithm, $0 \leq i \leq 3$) is a family of cryptographic hash functions; SHA-1 is a 160-bit hash function resembling MD5.

[3] BitTorrent is a peer-to-peer (P2P) communications protocol for file sharing.

increasing the efficiency of resource utilization. Without installing any software, a user can monitor approximately a dozen pre-selected metrics and then view graphs and statistics for these metrics. When launching an Amazon Machine Image (AMI), the user can start the CloudWatch and specify the type of monitoring. The basic monitoring is free of charge and collects data at five-minute intervals for up to ten metrics, while the detailed monitoring is subject to a charge and collects data at one minute interval. This service can also be used to monitor access latency to EBS volumes, available storage space for RDS DB instances, the number of messages in SQS, and other parameters of interest.

Virtual Private Cloud (VPC) provides a bridge between the existing IT infrastructure of an organization and the AWS cloud; the existing infrastructure is connected via a Virtual Private Network (VPN) to a set of isolated AWS computing resources. VPC allows existing management capabilities, such as security services, firewalls, and intrusion-detection systems, to operate seamlessly within the cloud.

Auto Scaling exploits cloud elasticity and provides automatic scaling of EC2 instances. The service supports: grouping of instances, monitoring of the instances in a group, and defining *triggers*, pairs of CloudWatch alarms and policies, which allow the size of the group to be scaled up or down. Typically, a maximum, a minimum, and the regular size of the group are specified. An Auto Scaling group consists of a set of instances described in a static fashion by launch configurations. When the group scales up, new instances are started using the parameters for the *runInstances* EC2 call provided by the launch configuration; when the group scales down, the instances with older launch configurations are terminated first. The monitoring function of Auto Scaling carries out health checks to enforce specified policies; for example, a user may specify a health check for elastic load balancing, and then Auto Scaling will terminate an instance exhibiting a low performance and start a new one. Triggers use CloudWatch alarms to detect events and then initiate specific actions; for example, a trigger could detect when the CPU utilization of the instances in the group goes above 90% and then scale up the group by starting new instances. Typically, triggers to scale up and down are specified for a group.

AWS services introduced in 2012 include: *Route 53*—a low-latency DNS service managing user DNS public records; *Elastic MapReduce (EMR)*—service supporting a hosted Hadoop running on EC2 and based on the MapReduce paradigm discussed in Section 11.5; *Simple Workflow Service (SWS)*—supports workflow management and allows scheduling, management of dependencies, and coordination of multiple EC2 instances; *ElastiCache*—a service enabling web applications to retrieve data from a managed in-memory caching system rather than a much slower disk-based database; *DynamoDB*—a scalable and low-latency fully managed NoSQL database service; *CloudFront*—a web service for content delivery; *Elastic Load Balancer*—a cloud service to automatically distribute the incoming requests across multiple instances of the application. Two of these services, the CloudFormation and the Elastic Beanstalk, are discussed next.

CloudFormation allows the creation of a stack describing the infrastructure for an application. The user creates a template, a text file formatted as in Javascript Object Notation (JSON), describing the resources, the configuration values, and the interconnection among these resources. The template can be parameterized to allow customization at run time e.g., to specify the types of instances, database port numbers, or RDS size.

Elastic Beanstalk interacts with other AWS services including EC2, S3, SNS, Elastic Load Balance, and AutoScaling. Elastic Beanstalk handles automatically the deployment, capacity provisioning, load balancing, auto-scaling, and application monitoring functions [488]. The service automatically scales the resources as required by the application, either up or down based on default Auto Scaling settings.

Some of the management functions provided by the service are: (i) deploying a new application version or rollback to a previous version; (ii) accessing results reported by CloudWatch monitoring service; (iii) providing email notifications when application status changes or application servers are added or removed; and (iv) accessing server log files without needing to login to the application servers.

Elastic Beanstalk service is available to developers using either a Java platform, the PHP server-side description language, or *.NET* framework. For example, a Java developer can create an application using an Integrated Development Environment, such as Eclipse, and package the code into Java Web Application Archive file of type ".war". The ".war" file should then be uploaded to the Elastic Beanstalk using the Management Console and then deployed, and in a short time the application will be accessible via an URL.

Lambda service: cloud computing is associated for many with Big Data applications and long-lasting computations rather than with real-time applications and short bursts of computing triggered by some external events. A few years ago, most likely anticipating services relevant to the Internet of Things, AWS introduced a *servless* computer service. In this case, applications are triggered by conditions and/or events specified by the end user. For example, the application may run for a brief period of time at midnight to check the daily energy consumption of an enterprise or may be activated weekly to check the sales of a chain or to turn on the alarm system of a home triggered by an event generated by the smartphone of the owner. In stark contrast to EC2, when customers are billed on an hourly basis, e.g., if a C4 instance is used for one hour and ten minutes the billing is for two hours, the *Lambda* service is billed for the actual time with a resolution of milliseconds. The service seems relatively easy to use. "First you create your function by uploading your code (or building it right in the Lambda console) and choosing the memory, timeout period, and AWS Identity and Access Management (IAM) role. Then, you specify the AWS resource to trigger the function, either a particular Amazon S3 bucket, Amazon DynamoDB table, or Amazon Kinesis stream. When the resource changes, Lambda will run a function, launch and manage the compute resources as needed in order to keep up with incoming requests." Amazon Kinesis is a data-streaming platform.

Regions and availability zones. Amazon has 28+ data centers on several continents. An *availability zone* (AZ) is a data center with 50 000–80 000 servers using 25–30 MW of power. All regions have at least two availability zones interconnected by high-speed networks. Regions do not share resources and communicate through the Internet. Storage is automatically replicated within a region. S3 buckets are replicated within an availability zone and between the availability zones of a region, while *EBS* volumes are replicated only within the same availability zone. Critical applications should replicate important information in multiple regions to be able to function when servers in one region are unavailable due to catastrophic events. Most AWS services are available in all regions, though some are not.

The billing rates differ from one region to another and can be roughly grouped into four categories: low, medium, high, and very high billing. These rates are determined by the components of the operating costs including energy, communication, and maintenance costs. Region selection is motivated by the desire to minimize costs, reduce the communication latency, and increase reliability and security.

AWS Networking. Each AWS region has redundant *transit centers* connecting private links to other AWS regions, private links to AWS Direct Connect customers, and to the Internet through peering and paid transit. Most major regions are interconnected by private fiber channels, and this avoids peering issues, buffering problems, and capacity limitations that may occur on public links.

Peak traffic between availability zones of up to 25 Tbps is supported. The communication latency between availability zones is in the one- to two-milliseconds range. Communication latency between two servers has three components: (i) application $\mapsto$ guest OS $\mapsto$ hypervisor $\mapsto$ Network Interface (NIC)—in the milliseconds range; (ii) through the NIC—in the microseconds range; and (iii) over the fiber—in the nanoseconds range. Single Root I/O Virtualization virtualizes the NICs, and each guest gets its own virtual NIC.

Users have several choices to interact and manage AWS resources: (i) the Web Management Console, but not all options may be available in this mode; (ii) command-line tools; (iii) AWS SDK libraries and toolkits provided for several programming languages including Java, PHP,[4] C#, and Obj C; (iv) raw REST requests as shown in Section 11.1.

Amazon Web Services Licensing Agreement (AWSLA) allows the cloud service provider to terminate service to any customer at any time for any reason and contains a covenant not to sue Amazon or its affiliates for any damages that might arise out of the use of AWS. AWSLA prohibits the use of "other information obtained through AWS for the purpose of direct marketing, spamming, contacting sellers or customers." It prohibits AWS from being used to store any content that is "obscene, libelous, defamatory or otherwise malicious or harmful to any person or entity;" it also prohibits S3 from being used "in any way that is otherwise illegal or promotes illegal activities, including without limitation in any manner that might be discriminatory based on race, sex, religion, nationality, disability, sexual orientation or age."

AWS Continuing Evolution. Amazon is one of the most important forces driving the spectacular evolution of cloud computing in recent years. AWS infrastructure has benefited from a wealth of new technologies. At this point in time, AWS is probably the most attractive and cost-effective cloud computing environment, not only for enterprise applications but also for computational science and engineering applications.

The massive effort to continually expand the hardware and the software of the AWS cloud infrastructure is astounding. Amazon has designed its own storage racks; such a rack holds 864 disk drives and weighs over a ton. The company has designed and built its own power substations. Three of its regions, US West (Oregon), AWS GovCloud (US), and EU (Frankfurt), are 100% carbon neutral.

AWS Nitro System, the platform for next generation EC2 instances, is a combination of dedicated hardware and a lightweight supervisor. While traditional hypervisors virtualize all system resources including, CPU, storage, and networking, the Nitro System allows AWS to "break apart those functions, offload them to dedicated hardware and software, and reduce costs by delivering practically all resources of a server to your instances." AWS offers several classes of EC2 instances powered by different processors. Representative instances in each class, with the most performant one listed first, are:

1. General purpose—provide a balance of computing, memory, and networking resource.
   A1—AWS Graviton with 64-bit ARM Neoverse cores and custom silicon.
   T3, **T3a**—AWS Nitro System; burstable general-purpose instance type.
   M6g—ARM-based AWS Graviton2.

[4] PHP evolved from a set of Perl scripts designed to produce dynamic web pages in a general-purpose server-side scripting language. The code embedded into an HTML source document is interpreted by a web server with a PHP processor module, which generates the resulting web page.

2. Compute optimized—for computing bound applications; high-performance processors.
   C6g—ARM-based AWS Graviton2.
   C5n—3.0-GHz Intel Xeon Platinum with AVX-512 instruction set.
3. Memory optimized—deliver fast performance for large memory footprint workloads.
   R6g—Arm-based AWS Graviton2.
   X1e—Intel Xeon E7-8880 v3; up to 3 904 GiB of DRAM memory.
   R5a—AMD EPYC 7000; all core turbo clock speed of 2.5-GHz AWS Nitro System.
4. Accelerated computing—instances use hardware accelerators, or co-processors, to perform functions, such as floating-point number calculations, graphics processing, or data pattern matching, efficiently.
   P3—High-frequency Intel Xeon E5-2686 v4 (Broadwell) or 2.5 GHz (base) Intel Xeon P-8175M and up to 8 NVIDIA Tesla V100 GPUs, each pairing 5 120 CUDA Cores and 640 Tensor Cores.
   P2—Intel Xeon E5-2686 v4 (Broadwell); NVIDIA K80 GPUs, each with 2 496 parallel processing cores and 12 GiB of GPU memory.
   G4—Intel Xeon Scalable (Cascade Lake); NVIDIA T4 Tensor Core GPUs.
   G3—Intel Xeon E5-2686 v4 (Broadwell); NVIDIA Tesla M60 GPUs, each with 2 048 parallel processing cores and 8 GiB of video memory.
5. Storage optimized—for workloads that require high, sequential read-and-write access to very large data sets on local storage.
   I3—Intel Xeon E5-2686 v4 (Broadwell); non-volatile memory express (NVMe) SSD-backed instance storage.
   I3en—3.1 GHz Intel Xeon Scalable (Skylake) processors with AVX-512 instruction set; up to 60 TB of NVMe SSD.
   H1—2.3 GHz Intel Xeon E5 2686 v4; up to 16 TB of HDD storage.

Each instance packages a different combination of processors, memory, storage, and network bandwidth. The number of vCPUs, as well as the type of processor, its architecture, and clock speed, are different for different instance types. A vCPU is a virtual processor assigned to one virtual machine. Memory is given in Gibibytes, 1 GiB = 2^{30} bytes or 1 073 741 824 bytes, while 1 GB = 10^9 bytes. There are no recent benchmarks comparing the performance of supercomputers in the top 500 list with AWS instances.

2.3 Google Clouds

Google is a major player in cloud computing; it has pioneered applications of Artificial Intelligence (AI) and Machine Learning (ML), has developed TPU (Tensor Processing Units), and populated some of its instances with TPUs. Moreover, unlike Amazon, which is a very secretive organization, Google has disseminated information about the new developments in hardware and software of its cloud infrastructure. The large number of papers published by Google Research Division contributes significantly to advancements in cloud computing.

Google Cloud infrastructure consists of a large number of clusters in multiple geographical locations. A typical cluster has around 10 000 servers, and its workload is a mix of CPU-intensive batch computations and in-memory databases for latency-sensitive applications. Google's effort is concentrated in several areas of Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS) [204]. Services such as Gmail, Google Drive, Google Calendar, Picasa, and Google Groups are free of charge for individual users and available for a fee for organizations. These services are running on Google clouds and can be invoked from a broad spectrum of devices, including smartphones, tablets, and laptops. The data for these services is stored in data centers on the cloud.

Google App Engine. App Engine (AE) is an infrastructure for building web and mobile applications and running these application on Google servers. Initially, it supported only Python, but support for Java was added later. The database for code development can be accessed with GQL (Google Query Language) with a SQL-like syntax.

App Engine is an ensemble of computer, storage, search, and networking services. The *Compute Engine* (CE) supports the creation of VMs with resources tailored to the application needs. The CE configurations range from micro instances to the ones with 32 vCPUs or 208 GB of memory. Up to 64 TB of network storage can be attached to a VM. Always-encrypted local solid-state drive (SSD) block storage and automatic scaling are also supported. Several operating systems including Debian, CentOS, CoreOS, SUSE, Ubuntu, Red Hat, FreeBSD, or Windows 2008 R2 and 2012 R2 are supported. One can create and manage CE instances in several ways, including: (i) Google Cloud Console, a web UI comparable to management tools like vCenter or XenCenter; (ii) Google Cloud SDK; and (iii) Compute Engine API.

Container Engine (CntE) is a cluster manager and orchestration system for Docker containers built on the Kubernetes system. The *Container Registry* stores private Docker images. CntE schedules and manages containers automatically according to user specifications. JSON config files are used to specify the amount of CPU/memory, the number of replicas, and other relevant information. The Cloud Container Engine SLA commitment is a monthly uptime of at least 99.5%. *Cloud Functions* (CF) is a lightweight, event-based, asynchronous system to create single-purpose functions that respond to cloud events. CFs are written in Javascript and executed in a Node.js runtime environment. *Cloud Load Balancing* supports scalable load balancing on the Google Cloud Platform.

Cloud Storage is a unified object storage enabling a multi-region operation for applications including video streaming and frequently accessed web and images sites. *Cloud SQL* is a fully-managed database service, *Cloud Bigtable* is a high performance NoSQL database service for large analytical and operational workloads, and *Cloud Datastore* is a highly-scalable NoSQL database for web and mobile applications.

Big Data applications are supported by several services. *Bigquery* is a fully-managed enterprise data warehouse for large-scale data analytics. *Cloud Dataflow* supports stream and batch execution of pipelines. *Cloud Dataproc* manages Spark and Hadoop service. *Cloud Datalab* is an interactive tool for large-scale data exploration, analysis, and visualization. *Cloud Pub/Sub* is a global service for real-time reliable messaging and data streaming.

Network functionality is managed by *Cloud Virtual Network* (CVN). AppEngine users can connect resources to each other and isolate them from one another in a Virtual Private Cloud using the CVN. Cloud routers maintain virtual routers enabling Border Gateway Protocol (BGP) to route updates between a user Compute Engine network and user non-Google network. CVN supports Virtual Private Networks (VPNs) and distributed firewalls. *Cloud CDN* is a low-latency, low-cost content delivery network. *Cloud DNS* is a resilient, low-latency DNS for Google's worldwide network.

Several App Engine services support security. *Cloud Identity and Access Management* (IAM) provides tools to manage resource permissions and to map job functions within a company to groups and roles. *Cloud KMS* is a key management service enabling users to manage encryption and generate, use, rotate, and destroy AES256 key encryption keys. *Cloud Security Scanner* is used to scan for common vulnerabilities in Google App Engine applications. App Engine provides cloud development tools. *Cloud SDK* is a set of tools including *gcloud, gsutil*, and *bq*, to access the Compute Engine, the Cloud Storage and other services. *Cloud Source Repositories* provides Git version control to support collaborative development for applications or services. *Android Studio*, *PowerShell*, *IntelliJ*, *Eclipse,* and *Visual Studio* tools are also available.

The array of management tools includes *Stackdriver* which supports monitoring, logging, and diagnostics tools along with *Stackdriver Monitoring* tool, which provides performance, uptime, and overall health information on cloud applications. *Stackdriver Debugger* lets users inspect the state of an application without logging statements and without stopping or slowing down the application. *Stackdriver Error Reporting* counts, analyzes, aggregates crashes, and provides a cleaned exception stack trace.

Virtually all CSPs require users to specify in one form of another the type and amount of resources used; jobs that exceed these limits are either throttled or killed. Google uses a system called Autopilot, evolved from the previous job management sub-system of Borg, discussed in Section 4.7, to configure resources automatically. Autopilot adjusts the number of concurrent tasks in a job and the CPU/memory limits for individual tasks and uses vertical and horizontal scaling to reduce the difference between user-specified CPU and RAM resource limits and the actual usage, the so-called *slack*, while making sure that tasks do not run out of resources [426]. The slack of Autopiloted jobs is half of the slack of manually managed jobs, and the number of jobs severely impacted by out-of-memory is also significantly reduced.

A range of services supporting machine learning are also provided. *Cloud Machine Learning* is a managed service based on the TensorFlow model to build machine learning models, which work on any type of data. *Cloud Natural Language API* is a text analysis tool that can be used to extract information about people, places, events, etc., while *Cloud Speech API* enables developers to convert audio to text by applying powerful neural network models and *Cloud Vision API* is used to understand the content of an image by encapsulating powerful machine learning models.

Other widely used Google services

Gmail. The service hosts emails on Google servers and provides a web interface to access them and tools for migrating from Lotus Notes and Microsoft Exchange.

Google Docs is a web-based software for building text documents, spreadsheets, and presentations. It supports features such as tables, bullet points, basic fonts, and text size; it allows multiple users to edit and update the same document, to view the history of document changes, and it provides a spell checker. The service allows users to import and export files in several formats including Microsoft Office, PDF, text, and OpenOffice extensions.

Google Calendar is a browser-based scheduler; it supports multiple calendars for a user, the ability to share a calendar with other users, the display of daily/weekly/monthly views, search for events, and synchronization with the Outlook Calendar. The calendar is accessible from mobile devices; event reminders can be received via SMS, desktop pop-ups, or emails. It is also possible to share your calendar with other Google calendar users. *Picasa* is a tool to upload, share, and edit images; it provides 1 GB of disk space per user free of charge. Users can add tags to images and attach locations to photos using *Google Maps*. *Google Groups* enables users to host discussion forums to create messages online or via email.

Google Co-op enables users to create customized search engines based on a set of categories.

Google Base enables users to load structured data from different sources to a central repository, a very large, self-describing, semi-structured, heterogeneous database where each item follows a simple schema: item type, attribute names. Since few users are aware of this service, *Google Base* is accessed in response to keyword queries posed on *Google.com*, provided that there is relevant data in the database. To fully integrate Google Base, the results should be ranked across properties. Also, the service needs to propose appropriate refinements with candidate values in select menus; this is done by computing histograms on attributes and their values during query time.

Google Drive is an online service for data storage available for PCs, MacBooks, iPhones, iPads, and Android devices and allows organizations to purchase up to 16 TB of storage.

Specialized structure-aware search engines for several areas, including travel, weather, and local services, have already been implemented. However, the data available on the web covers a wealth of human knowledge; it is not feasible to define all the possible domains, and it is nearly impossible to discern where one domain ends and another begins.

Google adheres to a bottom-up, engineer-driven, liberal licensing, and user-application development philosophy, while Apple, a recent entry in cloud computing, tightly controls the technology stack, builds its own hardware, and requires the applications developed to follow strict rules. Apple products, including the iPhone, the iOS, the iTunes Store, Mac OS X, and iCloud, offer unparalleled polished and effortless interoperability, while the flexibility of Google results in more cumbersome user interfaces for the broad spectrum of devices running the Android OS.

Google manages vast amounts of data. In a world where users would most likely desire to use multiple cloud services from independent providers, the question of whether the traditional Data Base Management Services (DBMS) are sufficient to ensure interoperability comes to mind. A DBMS efficiently supports data manipulations and query processing but operates in a single administrative domain and uses a well-defined schema. The interoperability of data-management services requires *semantic integration* of services based on various schemas. An answer to the limitations of traditional DBMS are *dataspaces* that do not aim at data integration but at data co-existence [183].

2.4 Microsoft Windows Azure and online services

Azure and Online Services are PaaS and, respectively, SaaS cloud platforms from Microsoft. Windows Azure is an operating system, SQL Azure is a cloud-based version of the SQL Server, and Azure AppFabric is a collection of services for cloud applications.

Windows Azure has three core components (see Fig. 2.4): *Compute*, which provides a computation environment, *Storage* for scalable storage, and *Fabric Controller*, which deploys, manages, and monitors applications; it interconnects nodes consisting of servers, high-speed connections, and switches. The Content Delivery Network (CDN) maintains cache copies of data to speed up computations. The Connect subsystem supports IP connections between the users and their applications running on Windows Azure. The API interface to Windows Azure is built on REST, HTTP, and XML. The platform

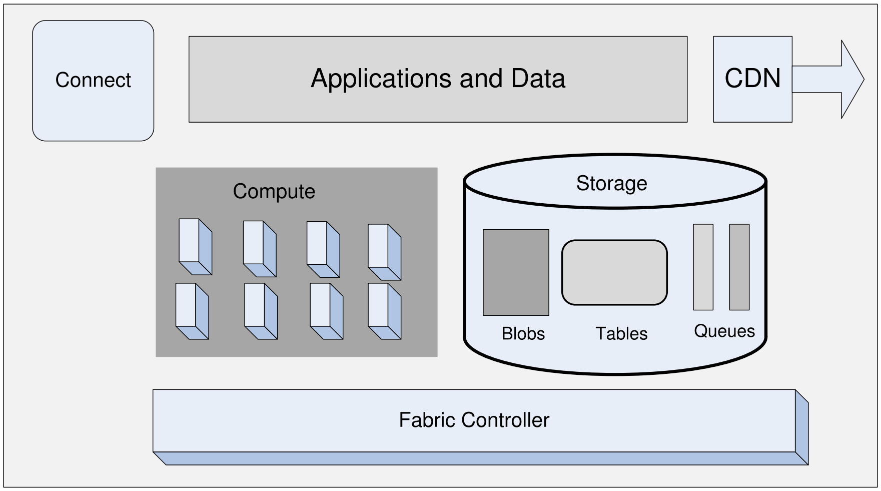

FIGURE 2.4

Azure components: Compute—runs cloud applications; Storage—uses blobs, tables, and queues to store data; Fabric Controller—deploys, manages, and monitors applications; CDN—maintains cache copies of data; and Connect—allows IP connections between the user systems and applications running on Windows Azure.

includes five services: Live Services, SQL Azure, AppFabric, SharePoint, and Dynamics CR. A client library and tools are also provided for developing cloud applications in Visual Studio.

Computations carried out by an application are implemented as one or more *roles*; an application typically runs multiple *instances of a role*. One distinguishes: (i) web role instances—to create web applications; (ii) worker role instances—run Windows-based codes; and (iii) VM role instances—run a user-provided Windows Server 2008 R2 image.

Scaling, load balancing, memory management, and reliability are ensured by a *fabric controller*, a distributed application replicated across a group of machines that owns all of the resources in its environment (computers, switches, and load balancers), and it is aware of every Windows Azure application. The fabric controller decides where new applications should run; it chooses the physical servers to optimize utilization using configuration information uploaded with each Windows Azure application. The configuration information is an XML-based description of how many web role instances, how many worker role instances, and what other resources the application needs; the fabric controller uses this configuration file to determine how many VMs to create.

Blobs, tables, queue, and drives are used as scalable storage. A blob contains binary data, and a container consists of one or more blobs. Blobs can be up to a terabyte, and they may have associated metadata, e.g., the information about where a JPEG photograph was taken. Blobs allow a Windows Azure role instance to interact with persistent storage as if it were a local NTFS[5] file system. Queues

[5] NTFS (New Technology File System) is the standard file system of the Microsoft Windows operating system starting with Windows NT 3.1, Windows 2000, and Windows XP.

enable web role instances to communicate asynchronously with worker role instances. Microsoft Azure platform currently does not provide or support any distributed parallel-computing frameworks, such as *MapReduce, Dryad* or *MPI*, other than the support for implementing basic queue-based job scheduling [213].

2.5 IBM clouds

In recent years IBM has accelerated its effort to eventually overtake Amazon, Google, Microsoft, and other CSP competitors. This effort was boosted by the 2018 acquisition of open-source software provider Red Hat and the 2020 decision to make a $1 billion investment in cloud computing. IBM is also a leader in quantum computing and quantum information theory and provides access to several quantum computing services as discussed in Section 13.3.

IBM is making a big play in hybrid clouds and gives companies tools to more easily navigate between public and private environments. Some of the reason for focusing on hybrid clouds are: (i) flexibility—users are able to run applications wherever they perform best; (ii) increased security—based on container built-in security; and (iii) performance—use resources of the public cloud for larger workloads after developing, testing, and running smaller workloads on a private cloud.

IBM is also operating in edge computing. The motivation for this effort is that 5G technology brings computation and data storage closer to where data is generated, enabling better data control, reduced costs, faster insights and actions, and continuous operations. It is estimated that, 75% of enterprise data will be processed at the edge in 2025, instead of only 10% today.

IBM offers a cloud platform with over 170 products and services. High-performance cloud servers can be configured in hourly and monthly options, including up to 20 TB of bandwidth. Red Hat helps to build and deploy container-based applications on a fully managed, integrated, and reliable platform. The emphasis on cloud security is winning over large organizations, and IBM AI services aim to modernize, collect, organize, analyze, and infuse user data in new ways to accelerate user's journey to AI.

2.6 Cloud storage diversity and vendor lock-in

The short history of cloud computing shows that cloud services may be unavailable for short, or even for extended, periods of time. Such an interruption of service is likely to impact negatively an organization using the cloud and possibly diminish, or cancel completely, the benefits of utility computing for that organization. The potential for permanent data loss in case of a catastrophic system failure poses an even greater danger.

Last, but not least, a vendor may decide to increase the prices for service and charge more for computing cycles, memory, storage space, and network bandwidth than other cloud service providers. The alternative, switching to another cloud service provider, could be very costly due to the large volume of data to be transferred from the old to the new provider. Transferring petabytes of data over the network takes a substantial amount time and incurs substantial charges for the network bandwidth.

A solution to problems posed by vendor lock-up is to replicate data to multiple cloud service providers. Straightforward replication is very costly and, at the same time, poses technical challenges.

The overhead to maintain data consistency could drastically affect the performance of the virtual storage system consisting of multiple full replicas of organization's data spread over multiple vendors.

Another solution could be based on an extension of the design principle of a RAID-5 system used for reliable data storage. This elegant idea immediately raises several questions: How does the response time of such a scheme compare with the one of a single storage system? How much overhead is introduced by a proxy? How could this scheme avoid a single point of failure, namely, the proxy? Are there standards for data access implemented by all vendors? An experiment to answer some of these question has been reported [8]; the RACS system uses the same data model and mimics the interface to AWS S3. The prototype implementation in [8] led the authors to conclude that the cost increases and the performance penalties of the RACS systems are relatively minor. An implementation avoiding the single point of failure uses several proxies; clients connected to several proxies can access data stored on multiple clouds.

In practice, it remains to be seen if such a solution is feasible for organizations with a very large volume of data, given the limited number of cloud storage providers and the lack of standards for data storage. A basic question is if it makes sense to trade basic tenets of cloud computing, such as simplicity and homogeneous resources controlled by a single administrative authority, for increased reliability and for freedom from vendor lock-in.

This brief discussion hints at the need for standardization and for scalable solutions, two of the many challenges faced by cloud computing in the near future. The pervasive nature of scalability dominates all aspects of cloud management and cloud applications. Solutions performing well on small systems no longer do when the system scale increases by one or more orders of magnitude. Experiments with small test-bed systems produce inconclusive results. The only alternative is to conduct intensive simulations to prove, or disprove, the advantages of a particular algorithm for resource management or the feasibility of a particular data-intensive application.

2.7 Cloud interoperability

Cloud interoperability could alleviate the concerns that users become hopelessly dependent on a single cloud service provider, the so called vendor lock-in, discussed in Section 2.6. It seems natural to ask the question if a "cloud of clouds," a federation of clouds that cooperate to provide a better user experience, is technically and economically feasible. Since the Internet is a network of networks, a federation of clouds seems plausible [58–60].

Closer scrutiny shows that the extension of the concept of interoperability from networks to clouds is far from trivial. A network offers one high-level service, the transport of digital information from a source, a host outside a network to a destination, another host, or another network that can deliver the information to its final destination. This transport of information through a network of networks is feasible because agreements on basic questions were reached before the Internet was born. It was then decided on how to: (a) uniquely identify the source and the destination of the information; (b) navigate through a maze of networks; and (c) transport the data between a source and a destination. The three elements on which agreements were reached are, respectively, the IP address, the IP protocol, and transport protocols such as TCP and UDP.

The situation is quite different in cloud computing. First, there are no standards for either storage or processing; second, the clouds we have seen so far are based on different delivery models. Moreover, the set of services supported by each of these delivery models is not only large, but it is open; new services are offered every few months. The question if Cloud Service Providers are willing to cooperate to build up a federation of clouds is an open one. Some CSPs may think that they have a competitive advantage due to the uniqueness of the added value of their services. Thus, exposing how they store and process information may adversely affect their business. Moreover, no CSP will be willing to change its internal operation, so a first question is if an Intercloud could be built under these conditions.

Following the concepts borrowed from the Internet, a federation of clouds that does not dictate the internal organization or the structure of a cloud, but only the means to achieve cloud interoperability, is feasible. Nevertheless, building such an infrastructure seems to be a formidable task. First, we need a set of standards for interoperability covering items such as: naming, addressing, identity, trust, presence, messaging, multicast, and time. We need common standards for identifying all objects involved, the means to transfer, store, and process information, and we also need a common clock to measure the time between two events.

An Intercloud would then require the development of an *ontology*[6] for cloud computing. Then each cloud service provider would have to create a description of all resources and services using this ontology. Due to the very large number of systems and services, the volume of information provided by individual cloud service providers would be so large that a distributed database, not unlike the Domain Name Service (DNS), would have to be created and maintained. According to [58], this vast amount of information would be stored in Intercloud *root* nodes, analogous to the root nodes of the DNS.

Each cloud would then require an interface, a so-called Intercloud *exchange*, to translate the common language describing all objects and actions included in a request originating from another cloud in terms of its internal objects and actions. To be more precise, a request originating in one cloud would have to be translated from the internal representation in that cloud to a common representation based on the shared ontology, and then, at the destination, it should be translated into an internal representation that can be acted upon by the destination cloud. This raises immediately the question of efficiency and performance. This question cannot be fully answered now, as an Intercloud exists only on paper, but there is little doubt that the performance will be greatly affected.

Security is a major concern for cloud users, and an Intercloud could create new threats. The primary concern is that tasks will cross from one administrative domain to another and sensitive information about tasks and user could be disclosed during this migration. A seamless migration of tasks in an Intercloud requires a well-thought-out trust model.

The Public-Key Infrastructure (PKI), an all-or-nothing trust model, is not adequate for an Intercloud where the trust must be nuanced. PKI is a model to create, distribute, revoke, use, and store digital certificates. It involves several components: (1) The Certificate Authority (CA) binds public keys to user identities in a given domain; (2) the third-party Validation Authority (VA) guarantees the uniqueness of the user identity; and (3) the Registration Authority (RA) guarantees that the binding of the public key to an individual cannot be challenged, the so-called *non-repudiation*. A nuanced model for handling digital certificates means that one cloud acting on behalf of a user may grant access to another cloud to read data in storage, but not to start new instances.

A solution for trust management is based on dynamic *trust indexes* that can change over time [59]. The Intercloud roots play the role of the CA, while the Intercloud exchanges determine the trust indexes

[6] An ontology provides the means for knowledge representation within a domain. It consists of a set of domain concepts and the relationships among the concepts.

between clouds. Encryption must be used to protect the data in storage and in transit in the Intercloud. OASIS[7] Key Management Interoperability Protocol (KMIP) is proposed for key management. In summary, an Intercloud opens up a wide range of interesting research topics. The practicality of the concepts can only be discussed if cloud standardization efforts under way at NIST bear fruit.

2.8 Service-level Agreements and Compliance-level Agreements

A Service Level Agreement (SLA) is a negotiated contract between two parties, the customer and the service provider. The agreement can be legally binding or informal and specifies customer services rather than how the service provider delivers services. The objectives of the agreement are: (i) identify and define the customer's needs and constraints, including the level of resources, security, timing, and quality of service; (ii) provide a framework for understanding including a clear definition of classes of service and costs; (iii) simplify complex issues, for example, clarify the boundaries between the responsibilities of clients and those of service provider in case of failures; (iv) reduce areas of conflict and eliminate unrealistic expectations; and (v) encourage dialog in the event of disputes.

An SLA records a common understanding in several areas: (i) services, (ii) priorities, (iii) responsibilities, (iv) guarantees, and (v) warranties. An agreement usually covers: services to be delivered, performance, tracking and reporting, problem management, legal compliance and resolution of disputes, customer duties and responsibilities, security, handling of confidential information, and termination.

Each area of service in cloud computing should define a "target level of service" or a "minimum level of service" and specify the levels of availability, serviceability, performance, operation, or other attributes of the service, such as billing; penalties may also be specified in the case of non-compliance with the SLA. It is expected that any Service-Oriented Architecture (SOA) will eventually include middleware supporting SLA management; the Framework 7 project supported by the European Union is researching this area, see http://sla-at-soi.eu/.

There are two well-differentiated phases in SLA management: the negotiation of the contract and the monitoring of its fulfillment in real-time. In turn, automated negotiation has three main components: (i) the *object of negotiation* that defines the attributes and constraints under negotiation; (ii) the *negotiation protocols* that describe the interaction between negotiating parties; and (iii) the *decision models* that are responsible for processing proposals and generating counter proposals.

It is critical for a cloud user to carefully read the service-level agreement and to understand the limitations of the liability a cloud provider is willing to accept. In many instances, the liabilities do not apply to damages caused by a third party or to failures attributed either to customer's hardware and software or to hardware and software from a third party. For example, if a distributed denial of service attack (DDoS) causes the entire IaaS infrastructure to fail, the cloud service provider is responsible for the consequences of the attack. The user is responsible if the DDoS affects only several instances including the ones running the user application.

The concept of compliance in cloud computing is discussed in [70] in the context of the user ability to select a provider of service; the selection process is subject to customizable compliance with

[7] OASIS stands for Organization for the Advancement of Structured Information Standards.

user requirements such as security, deadlines, and costs. The authors propose an infrastructure called *Compliant Cloud Computing* (C3) consisting of: (i) a language to express user requirements and the Compliance Level Agreements (CLA), and (ii) the middleware for managing CLAs. A policy-based framework for automated SLA negotiation for a virtual computing environment is described in [520].

2.9 Responsibility sharing between user and service provider

After reviewing cloud services provided by Amazon, Google, and Microsoft, we are in a better position to understand the differences between SaaS, IaaS, and PaaS. There is no confusion about SaaS, the service provider supplies both the hardware and the application software; the user has direct access to these services through a web interface and has no control on cloud resources. Typical examples are Google with Gmail, Google Docs, Google Calendar, Google Groups, and Picasa and Microsoft Online Services.

Fig. 2.5 describes the limits of responsibility between a user and CSP. An SaaS user is partially responsible for the interface; the user responsibility increases in the case of PaaS and includes the interface and the application. An IaaS user is responsible for all events in the virtual machine running the application; the service provider supplies the hardware (servers, storage, networks), and system software (operating systems, databases) and ensures system attributes such as security, fault-tolerance, and load balancing. PaaS provides only a platform including the hardware and system software such as operating systems and databases; the service provider is responsible for system updates, patches, and the software maintenance. PaaS does not allow any user control of the operating system, security features, or the ability to install applications. Typical examples are Google App Engine, Microsoft Azure, and Force.com, provided by Salesforce.com.

2.10 User challenges and experience

Computer clouds pose a fair number of challenges to software developers and cloud users, including security, compliance, managing costs, and lack of expertise. Cloud users attempt to reduce their cloud costs through a range of measures including: (i) careful resource utilization monitoring; (ii) avoiding peak hours and shutting down temporary workloads; (iii) purchasing AWS reserved instances along with effective use of spot instances; and (iv) moving workloads to regions with lower costs.

There are a few studies of user experiences based on a large population of cloud computing users. An empirical study of the experience of a small group of users of the Finish Cloud Computing Consortium is reported in [383]. The main user concerns are: security threats; the dependence on fast Internet connections; forced version updates; data ownership; and user behavior monitoring. While all users reported that trust in the cloud services is important, two-thirds raised the issue point of fuzzy boundaries of liability between cloud user and the provider, about half did not fully comprehend the cloud's functions and its behavior, and about one-third were concerned about security threats.

The security threats perceived by this group of users are: (i) abuse and villainous use of the cloud; (ii) APIs that are not fully secure; (iii) malicious insiders; (iv) account hijacking; (v) data leaks; and (iv) issues related to shared resources. Identity theft and privacy were a major concern for about half of

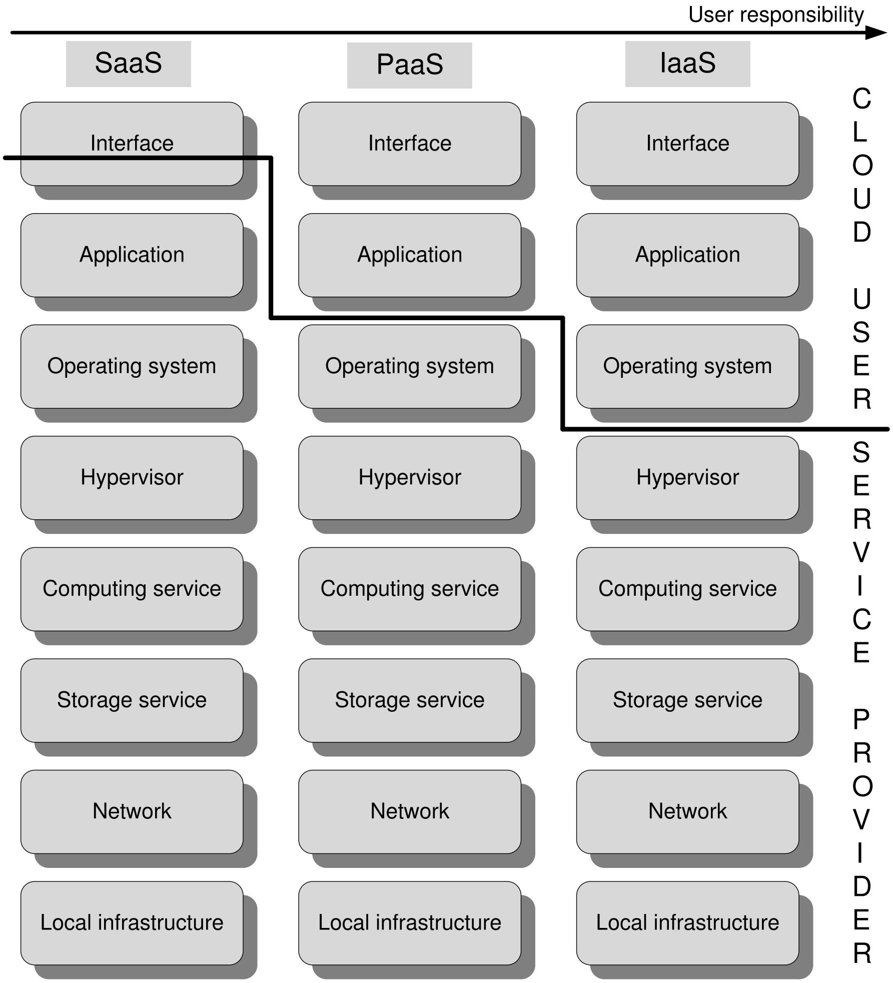

FIGURE 2.5

The limits of responsibility between a cloud user and the cloud service provider.

the users questioned; availability, liability, and data ownership and copyright was raised by a third of respondents.

The suggested solutions to these problems are: Service Level Agreements and tools to monitor usage should be deployed to prevent the abuse of the cloud; data encryption and security testing should enhance the API security; an independent security layer should be added to prevent threats caused by malicious insiders; strong authentication and authorization should be enforced to prevent account hijacking; data decryption in a secure environment should be implemented to prevent data leakage; and compartmentalization of components and firewalls should be deployed to limit the negative effect of resource sharing.

A broad set of concerns identified by the NIST working group on cloud security includes: (i) potential loss of control/ownership of data; (ii) data integration, privacy enforcement, data encryption; (iii) data remanence after de-provisioning; (iv) multi-tenant data isolation; (v) data location requirements within national borders; (vi) hypervisor security; (vii) auditing of data-integrity protection; (viii) verification of subscriber policies through provider controls; and (ix) certification/accreditation requirements for a given cloud service.

A study conducted by IBM [249] identified barriers to public and private cloud adoption. The top workloads mentioned by users involved in this study are: data mining and other analytics (83%), application streaming (83%), help-desk services (80%), industry specific applications (80%), and development environments (80%). The study also identified workloads that are not good candidates for migration to a public cloud environment, such as: (a) sensitive data such as health care records; (b) multiple co-dependent services, e.g., online transaction processing; (c) third-party software without cloud licensing; (d) workloads requiring auditability and accountability; and (e) workloads requiring customization.

2.11 Software licensing

Software licensing for cloud computing is an enduring problem without a universally accepted solution. License-management technology is based on the old model of computing centers with licenses granted on the basis of named users or as site licenses. This technology developed for a centrally managed environment cannot accommodate the distributed service infrastructure of cloud computing. IBM reached an agreement allowing some of its software products to be used on EC2. Also, MathWorks developed a business model for MATLAB® use in Grid environments [84]. SaaS deployment model is gaining acceptance because it allows users to pay for only the services they use.

There is significant pressure to change traditional software licensing and find non-hardware-based solutions for cloud computing; the increased negotiating power of the users coupled with the increased software piracy has renewed interest in alternative schemes such as those proposed by the SmartLM research project (http://www.smartlm.eu). SmartLM license management requires a complex software infrastructure involving Service Level Agreement, negotiation protocols, authentication, and other management functions.

A commercial product based on the ideas developed by this research project is *elasticLM,* which provides license and billing Web-based services [84]. The architecture of the elasticLM license service has several layers: co-allocation, authentication, administration, management, business, and persistency. The authentication layer authenticates communications between the license service and the billing service, as well as the individual applications; the persistence layer stores the usage records; the main responsibility of the business layer is to provide the licensing service with the licensing prices; the management coordinates various components of the automated billing service.

When a user requests a license from the license service, the terms of the license usage are negotiated, and they are part of a Service Level Agreement (SLA) document; the negotiation is based on application-specific templates and the license cost becomes part of the SLA. SLA describes all aspects of resource usage, including application ID, duration, number of processors, and guarantees, such as the maximum cost and deadlines. When multiple negotiation steps are necessary, the WS-Agreement Negotiation protocol is used.

To verify the authorization to use a license, an application must have the certificate of an authority. This certificate must be available locally to the application because the application may be executed in an environment with restricted network access; this opens the possibility for an administrator to hijack the license mechanism by exchanging the local certificate.

2.12 Challenges faced by cloud computing

Cloud computing inherits some of the challenges of parallel and distributed computing discussed in Chapter 3 and, at the same time, faces major challenges of its own. The complexity of hardware and software infrastructure supporting cloud service is astounding. Two billion lines of code are maintained by Google and drive applications such as Google Search, Google Maps, Google Docs, Google+, Google Calendar, Gmail, YouTube, and every other Google Internet service. By comparison, Windows operating system developed by Microsoft since the 1980s has some 50 million lines of code, a factor of 40 less than what Google has developed during two decades of existence.

The software stack for cloud computing has dramatically evolved in its quest to provide a unified higher-level view of the system, rather than a large collection of individual machines. Virtualization and containerization are ubiquitous abstractions that allow easier access to the increasingly larger and diverse population of cloud users. Distributed and semi-structured storage systems such as Google's BigTable [94] or Amazon's Dynamo [133] are widely used. The list of systems supporting higher-level abstractions includes Pig [190], FiumeJava [90], and Spark [532]. The effort to build one layer of abstraction removed from the underlaying hardware, a sort of operating system for Internet-scale jobs is underway. Systems such as Dryad [255], DryadLinq [530], Mesos [240], Borg [495], Omega [442], and Kubernets [80] attempt to bridge the gap between a clustered infrastructure and the assumptions made by applications about their environments. These systems manage a virtual computer aggregating the resources of a physical cluster with a very large number of independent servers.

In the early days of network-centric computing, it was postulated that Web searching is the "killer application" that will drive the software and hardware of large-scale systems for the next decades [52]. It turns out that the applications running on computer clouds are very diverse. *The broad spectrum of cloud applications adds to the challenges faced by cloud infrastructure. For example, controlling the tail latency of workloads consisting of a mix of time-critical and batch jobs is far from trivial [130]. Some critical system requirements are contradictory, e.g., multiplexing resources to increase efficiency and lowering the response time, while supporting performance and security isolation.*

The specific challenges differ for the cloud delivery models, but in all cases, the difficulties are created by the very nature of utility computing based on resource sharing and resource virtualization. Moreover, cloud computing requires a different trust model than the ubiquitous user-centric model we have been accustomed to for a very long time. *The most significant challenge is security; gaining the trust of a large user base is critical for the future of cloud computing. It is unrealistic to expect that a public cloud will provide a suitable environment for all applications.*

The SaaS model faces similar challenges because other online services are required to protect private information, such as financial or healthcare services. Since in this case, a user interacts with cloud services through a well-defined interface, in principle, it is less challenging for the provider of service to close some of the attack channels. Still, such services are vulnerable to denial-of-service attacks, and the users are fearful of malicious insiders.

Data in storage is most vulnerable to attacks, so special attention should be devoted to the protection of storage servers. Data replication necessary to ensure continuity of service in the case of storage-system failure increases vulnerability. Data encryption may protect data in storage, but eventually, data must be decrypted for processing, and then it is exposed to attacks. Homomorphic encryption enables logic operations to be performed on encrypted data, but it is impractical at this time.

The IaaS is by far the most challenging model to defend against attacks; indeed, an IaaS user has considerably more degrees of freedom than allowed by the other two cloud delivery models. An additional source of concern is that the considerable resources of a cloud could serve as the host to initiate attacks against the networking and the computing infrastructure.

Virtualization is a critical design option for this model, but it exposes the system to additional sources of attacks. The trusted computing base (TCB) of a virtual environment includes not only the hardware and the hypervisor, but also the management operating system. As we shall see in Section 8.10, the entire state of a VM can be saved to a file to allow migration and recovery, both highly desirable operations; yet, this possibility challenges the strategies to bring the servers belonging to an organization to a desirable and stable state. Indeed, an infected VM can be inactive when the systems are cleaned up and an infected VM can wake up later and infect other systems. This is another example of the deep intertwining of desirable and undesirable effects of basic cloud computing technologies.

The next major challenge is related to cloud resource management. A systematic, rather than ad hoc, resource-management strategy requires the existence of controllers tasked to implement several classes of policies: admission control, capacity allocation, load balancing, energy optimization, and, last but not least, to provide Quality of Service (QoS) guarantees.

It seems reasonable to expect that such a complex system can only function based on self-management principles. But self-management and self-organization raise the bar for the implementation of logging and auditing procedures critical for the security and trust in a provider of cloud computing services. Under self-management, it becomes next to impossible to identify the reasons why a certain action that resulted in a security breach was taken.

The last major challenge we address is related to interoperability and standardization. Vendor lock-in, the fact that a user is tied to a particular cloud service provider is a major concern for cloud users. Standardization would support interoperability and, thus, alleviate some of the fears that a service critical for a large organization may not be available for an extended period of time. But imposing standards at a time when a technology is still evolving is not only challenging but can be counterproductive because it may stiffen innovation.

Further research in various areas of cloud computing is an important objective of the National Science Foundation (NSF). The NSF supports research-community access to two cloud facilities, CloudLab and Chameleon. *CloudLab* is a testbed allowing researchers to experiment with cloud architectures and new applications. Some 15 000 cores, at three sites in Utah, Wisconsin, and South Carolina, are available for such experiments. *Chameleon* is an OpenStack KVM experimental environment for large-scale cloud research.

2.13 Cloud computing as a disruptive technology

We present now a few statistics supporting the view that computer clouds are changing the enterprise computing landscape at a stunning speed. Gartner[8] predicts that the public cloud service revenues will increase from \$196.7 billion in 2018 to \$354.6 billion in 2022. More than \$1.3 trillion in IT spending will be affected by the shift to the cloud. PaaS, SaaS, and IaaS cloud service providers will almost double their revenues, from \$26.4 to \$58.0 billion, from \$85.7 to \$151.1 billion, and from \$32.4 to \$74.1 billion, respectively.

Eighty-two percent of the enterprise workload resided on clouds and 40 zettabytes (40×10^{21} bytes) of data flowed through cloud servers and networks by the end of 2020. Ninety-four percent of enterprises use the cloud, and 66% of enterprises have dedicated cloud teams working on cloud cost optimization, selecting applications to be run on clouds and setting up policies for cloud users. Manufacturing, professional services, and banking plan to spend the most on cloud computing services, \$19.7 billion, \$18.1 billion, and \$16.7 billion, respectively.

Cost-cutting is the top reason why companies migrate to the cloud. Eighty percent of companies report operation improvements within the first few months of adopting cloud computing. Organizations with more than 1 000 employees are mainly looking for flexibility and reduced costs of operation, while smaller companies want to ensure their business continuity.

A 2021 study reports that: 78% of organizations participating in the study use hybrid clouds, i.e., public and private; 19% use only public and 2% use one private clouds; and 31% of respondents spend more than \$12 million annually on cloud computing. Companies with less than 1 000 employees (SMBs) are the driving force of the economy; 41% of SMBs favor the public cloud, and 94% of SMBs adopt the cloud due to upgraded security. Fifty percent of US government organizations are now using the cloud.

AWS has the largest cloud computing market share of 32%. Dropbox, Google Drive, and Microsoft OneDrive are the leading cloud storage providers with 47.3%, 26.9%, and 15.3% of the market, respectively. The multicloud is an aspirational goal for enterprises concerned about vendor lock-in. Many organizations are working hard to abstract their applications and allow them to be moved across clouds. Legacy vendors have created platforms that can plug into multiple clouds using VMware or Red Hat.

It is expected that in the immediate future artificial intelligence, analytics, the IoT, edge computing, and serverless and managed services will differentiate the top cloud service providers. All these areas require a highly skilled workforce. Ergo, cloud computing experts are in high demand; site reliability engineers, enterprise account executives, customer success managers, and solution architects are sought after, according to LinkedIn.

While enterprises migrate in large numbers to cloud computing to cut costs and reduce risk, the latter rationale is increasingly being challenged. On one hand, the number of potentially vulnerable and attractive targets increases, and, on the other hand, hackers are becoming more adept at exploiting vulnerabilities in cloud systems, see Section 8.15. Some of these vulnerabilities are exposed by the increasingly larger number of cloud users, some with little or no training and understanding of potential risks associated with cloud computing.

[8] The sources for data presented in this section: (i) https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020; (ii) https://resources.flexera.com/web/pdf/report-cm-state-of-the-cloud-2021.pdf.

2.14 Exercises and problems

Problem 1. The list of desirable properties of a large-scale distributed system includes transparency of access, location, concurrency, replication, failure, migration, performance, and scaling. Analyze how each one of these properties applies to AWS.

Problem 2. Compare the three cloud computing delivery models, SaaS, PaaS, and IaaS, from the point of view of application developers and users. Discuss the security and the reliability of each one of them. Analyze the differences between the PaaS and the IaaS.

Problem 3. Compare the Oracle Cloud offerings (see https://cloud.oracle.com) with the cloud services provided by Amazon, Google, and Microsoft.

Problem 4. Read the IBM report [249] and discuss the workload preferences for private and public clouds and the reasons for the preferences.

Problem 5. Many organizations operate one or more computer clusters and contemplate the migration to private clouds. What are the arguments for and against such an effort?

Problem 6. Evaluate the SLA toolkit at http://www.service-level-agreement.net/. Is the interactive guide useful, and what does it miss? Does the SLA template include all clauses that are important in your view, and what is missing? Are the examples helpful?

Problem 7. Software licensing is a major problem in cloud computing. Discuss several ideas to prevent an administrator from hijacking the authorization to use a software license.

Problem 8. Annotation schemes are widely used by popular services such as Flickr photo-sharing service, which support the annotation of photos. Sketch the organization of a cloud service used for sharing medical x-ray, tomography, CAT-scans, and other medical images and discuss the main challenges for its implementation.

Problem 9. An organization debating whether to install a private cloud or to use a public cloud, e.g., the AWS, for its computational and storage needs, asks your advice. What information will you require to base your recommendation on, and how will you use each one of the following items: (a) the description of the algorithms and the type of the applications the organization will run; (b) the system software used by these applications; (c) the resources needed by each application; (d) the size of the user population; (e) the relative experience of the user population; and (f) the costs involved?

Problem 10. A university is debating the question in Problem 9. What will be your advice and why? Should software licensing be an important element of the decision?