1

PRINCIPLES AND CONCEPTS OF CLOUD COMPUTING

1.1 KINDS OF MODERN SOFTWARE ARCHITECTURES

Before diving into cloud computing itself, let us consider some important concepts and kinds of modern software architectures and analyze the place of cloud computing in this scheme.

Here are some typical kinds of modern software:

- Client–server systems
- Web services and Web applications
- Integrated distributed software solutions
- Built-in systems
- Real-time systems
- Software for mobile devices
- Software for wearable computers
- Middleware (midlevel software)
- Software for cloud computing and datacenters
- Software for computer clusters
- Software for virtualization
- Software for information management
- Software for knowledge management
- Software for scientific computing.

In general, modern software architectures tend to get more and more complicated.

Trustworthy Cloud Computing, First Edition. Vladimir O. Safonov.

Client–server system paradigm and architecture have become widely spread for decades. A client–server system consists of a server or set of servers and a set of clients, connected to a local area network. The following kinds of servers are used in most local networks: *application* server, *Web* server, *email* server, *database* server, *file* server, and so on [13].

Internet (Web) applications are intended for use on the net. Currently the majority of them are developed on .NET [14] or Java [15] platforms, though some software developers still prefer to write Internet applications in older languages such as C. In modern Web programming, languages with *dynamic types* are widely used – JavaScript, Python, and Ruby. Their characteristic features are the dynamic change and construction of new types at runtime, which is comfortable, since it reflects the dynamic nature of Web applications and Web sites.

Internet applications are classified into *client side* applications (e.g., Web browsers) and *server side* applications (e.g., Web services).

Integrated software solutions are distributed software systems for information processing and supporting the business and functioning of enterprises, companies, banks, or universities. The characteristic features of integrated software solutions are modules for authentication and authorization of users, modules for accessing databases, modules for networking, and modules for implementing the business logic of the company. Integrated solutions can be developed using several programming languages.

Built-in systems are software for specialized microprocessors controlling various kinds of specific devices, from nuclear reactors to freezers, cardiostimulators, electric power transmission systems, and cars. The characteristic requirement of such software is fixed *response time* interval with some critical upper limit that is dramatic to satisfy for usability, reliability, and security of the whole system and the controlled object in general, or even for continuing the life of some living organism to be controlled. The typical requirement to the basic working cycle of such a system is the absence of interrupts, which could cause critically undesirable time delays.

Software for mobile devices is one of the most modern and widely used kinds of software. Its specific characteristics are the use of limited amount of resources (for the first turn, limited memory size) and the need to take into account a variety of models of mobile devices (differences of their screens and control keys) when implementing the graphical user interface of mobile applications. Currently, most of the software for mobile devices is developed on the Java platform; an especially popular mobile platform now is Google Android.

As an exotic but real-life example, consider the *software for wearable computers*. Such specialized computers are built into specific kinds of wear or uniform (e.g., space suits) that they monitor the state, health, and behavior of a human, and give expert recommendations to him or her. This class of software also has strict limitations on the computing resources used.

Middleware (or *mid-level software*) is a kind of communication software present in the software architectural scheme between the client and the server and supporting

their networking communication protocols. A typical modern example of middleware is communication software for sending and receiving *instant messages* between mobile devices, and laptop and desktop computers.

Software for datacenters is yet another modern kind of software. Its most important components are powerful server-side operating system, middleware to support networking communications, and database management systems (e.g., Microsoft SQL Server).

Software for virtualization is a modern software intended for the installation and use of *virtual machines* on real computer hardware, with the purpose of extending computing features, using other kinds of operating systems, or using software developed for other hardware platforms. Examples of such software are *Microsoft Virtual PC* and a new toolkit *Microsoft Hyper-V hypervisor*.

Software for cloud computing consists of various kinds of server-side operating systems (e.g., Windows Server or Linux) and other software that supports the use of cloud resources (applications and data) by the cloud clients. It is considered in more detail subsequently in this chapter.

Software for knowledge management plays a more and more important role now, in relation to Web intellectualization and popularity of *intelligent software solutions* that contain modules of logical assessment of the computation results, in addition to computational modules. Examples of knowledge management software are the *Protégé* knowledge management system developed at Stanford University [16] implementing the ontology management language OWL and our own software product developed by our team under my supervision at St. Petersburg University – *Knowledge.NET* [17], an extension of the C# language by knowledge management features, implemented as a plug-in to Visual Studio.

Software for information management is a set of office applications for document processing (e.g., Microsoft Office) and database management systems (e.g., Oracle DMBS, MySQL, Microsoft SQL Server).

Software for scientific computing is a set of software tools and packages that support the solving of scientific tasks, for example, MATHLAB.

1.2 CHARACTERISTIC FEATURES OF MODERN SOFTWARE

Now let us consider the most characteristic features inherent to most of the software systems, regardless of their problem domain and their implementation platform.

The most important feature of modern software systems is their *Web awareness*. The most popular kinds of modern platforms supporting this feature are .NET and Java.

A characteristic feature of modern software is *unification of models of programs and data*, which actually follows from their Web and net awareness. The *Unified Modeling Language* (*UML*) has been used for more than 30 years as a de facto standard for modeling software and the processes of its development. As for data representation, the de facto standard in this area is *XML,* which enables unified structured textual representation of data, used especially when transferring them via the network.

Trustworthy computing is a modern approach to software development proposed by Microsoft. Its main idea is to take into account the requirements and considerations of security and reliability of the software product under development from the early specification and design stages, to implement those requirements in the software product, and to apply specific kinds of software testing and verification.

An important principle of modern software is a *unified infrastructure* that integrates the tools, data, programs, and knowledge used to solve various application tasks.

Reusability of software code is very important for successful development of software, since it allows the developers to save resources and efforts in software development by using the same software modules in several software solutions.

Service-oriented architecture (*SOA*) of software reflects the trend toward explicit formulation of the concept of *software service* (preferably, Web service). Please see more details on SOA in Section 1.4.

Virtualization is widely used in software for modeling new hardware architectures, extending the features of data access, memory size, and so on.

Cloud computing is currently one of the most popular approaches to development and use of software, implementing the metaphor of the cloud – part of the Internet or intranet network through which the users have access to computing resources – applications, data, and knowledge. This approach is overviewed in the Introduction and covered in detail in the book.

Knowledge management plays an important part in modern software since, to solve many kinds of real world tasks, the use of purely algorithmic methods is not enough; what is required is *integration of methods of software engineering and knowledge* engineering. This important idea is implemented in our knowledge management toolkit referred to as Knowledge.NET [17].

1.3 BASIC CONCEPTS OF MODERN SOFTWARE ARCHITECTURE

Now let us consider the basic concepts of modern software architecture important for understanding cloud computing.

A *client* is the user and/or computer consuming some software service on the network. There are several categories of clients, depending on the kind of services they consume: *Web* clients, *email* clients, *database* clients, and so on. In relation to cloud computing, we mostly consider *Web* clients, since the cloud is controlled via Web interface enabled by a Web browser.

A *server* is a computer or datacenter, a related set of computers providing some software services. From the viewpoint of the above classification, a datacenter can be regarded as a big, structured, and complicated *Web server*.

A *thin client* is a client of a Web service (i.e., a *Web client*) with minimal user interface, *stateless* (without storing information on its state), unable to keep information on its *session*, without a full-fledged GUI comparable in its features to a typical non-Web client application. So a thin client is a Web client communicating to the Web service via basic features of the browser and via the HTTP protocol to send HTTP

requests and to get HTML pages as response to them. From cloud computing viewpoint, the thin client scheme is surely non-suitable for cloud clients, although it is the simplest to implement. The cloud clients would expect from the cloud a comfortable user interface as they are accustomed to from most client applications, and without it they would consider the cloud non-suitable and non-trustworthy.

A *rich client* is a Web client having a rich user interface (windows, menus, scrollbars, buttons, images, etc.), communicating to the Web service via the layer of intermediate software (currently referred to as *middleware*) that enables GUI functionality and network communications. This middleware is usually implemented as a *plug-in* to the Web browser that should be installed on the client computer above the browser before using the cloud. Examples of such plug-ins are *Microsoft Silverlight, Oracle JavaFX, and Macromedia Flash*.

A *layer* is a major independently implemented component of software (group of software modules) that communicates with the other layers that constitute a software product. To use simplified geometric analogies to represent the architecture of the software, there can be *horizontal layers* and *vertical layers (cuts).*

Abstraction layer (also known as *horizontal layer*) which, in classical scheme, has *number N* as a related set of software modules whose implementation can only use (call) modules of the previous layer $N-1$ only ($N>0$). Abstraction layer is the implementation of a related set of intermediary-level concepts (modules). Abstraction layer number 0 is implemented by the target platform hardware or by core software libraries (APIs) usually predefined in the implementation programming language, for example, by the package *java.lang* in Java language. The concept of abstraction layers was formulated by Professor E. Dijkstra in the late 1960s when developing the "*THE*" operating system at the Technical University of Eindhoven, Holland [13].

Vertical layer (cut), referred to in modern software paradigms as *aspect* [1], is a collection of scattered fragments of software code implementing some *cross-cutting* functionality, for example, a set of security checks, in some application. The authors of this concept are Professor A.L. Fouxman (USSR, Rostov University, 1979, in his monograph titled "Technological aspects of software systems development"; his approach was called *technology of vertical cuts*) and G. Kiczales, the father of aspect-oriented programming (Xerox PARC, now University of British Columbia; the scientific advisor of the AspectJ project) [1].

Middleware is a collection of software layers that lie between the clients and the server and enable their interaction via communication protocols.

A *tier* is part of a software solution implementing some independent functionality of the software solution architecture. For example, a *business tier* is the implementation of the business logic of the solution; a *Web tier* is the implementation of the communication of the solution with the Web. A tier is a more complicated software concept than an abstraction layer. Abstraction layers can be represented, using geometric analogy, as vertically located segments of software relying bottom-up above each other, the layer N above the layer $N-1$. A tier, on the contrary, can be any part of the architecture of any software solution (or represent different parts of different software solutions), without any definite number, so the concept of the number for the tiers does not make sense. So a two-dimensional geometric analogy ("horizontal

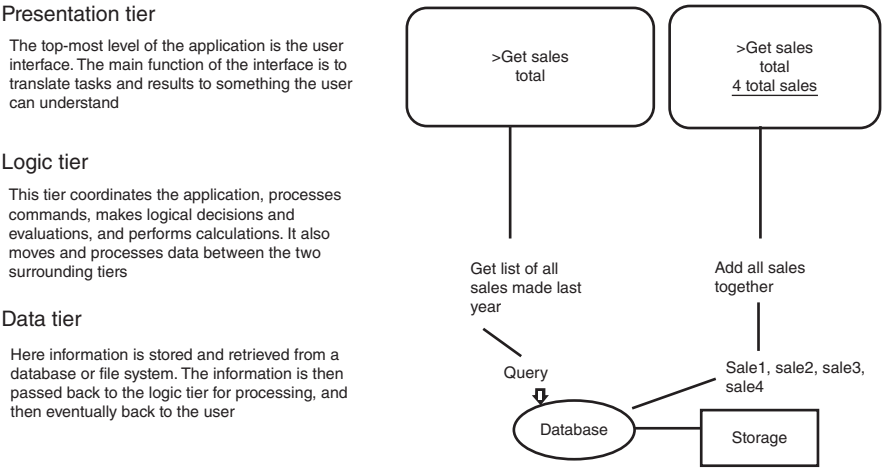

Figure 1.1 Multitiered architecture.

layer/vertical layer") does not work for modern software architectures whose more adequate representation could be imagined as a semantic net or a labeled graph.

Multitier architecture is a kind of software architecture based on the idea of implementing presentation of the results, data processing, and data control as separate processes. *Example:* using middleware to communicate with the server and using a database management system for communication with data.

An example of a multitiered architecture is depicted in Figure 1.1.

Multitenant architecture is a kind of architecture of client–server software based on the principle of the use of the same instance of a server solution, running on the server, by many *tenants* (clients). An example of multitenant software is a Web service.

From the viewpoint of the above approaches, cloud computing can be characterized as satisfying the principles of *multitiered* and *multitenant architecture*.

As for abstraction layers, for specifying modern software architectures this concept looks obsolete, since most of the modules in a software solution are reusable; so in different solutions the same tiers could have different numbers. Two-dimensional geometric models cannot adequately specify modern software architectures.

1.4 SERVICE-ORIENTED ARCHITECTURE (SOA)

SOA is one of the most up-to-date approaches to software development, based on the idea of representing software as an extensible set of *services* (typically, *Web* services).

Service is a software component available for the user for consuming, adding to the current workspace, and monitoring.

Simple example of a service: Getting the weather forecast via the Internet.

The *basic principle of SOA* is as follows. From the client viewpoint, a set of the software products to be consumed by the user is represented as a set of simple-to-use

Web services with comfortable graphical user interface. This working set of services is often referred to as *mash-up*.

A service-oriented model should be extensible: the client should be able to add new services or change the working set of available services.

The clients should be also able to call and consume services from different kinds of computing devices – desktop, laptop or tablet computers, or mobile devices.

The interface of a service is referred to as its *contract*. It is a set of *Web methods*. Each of the Web methods is specified by its name and types of its arguments. The contracts of each Web method should be explicitly specified and should be available for requests of SOA clients who are interested in the full set of the available Web methods.

The platform of implementation of services should be insignificant for the user. A service can be implemented on the .NET, or in Java, or any other suitable platform. The only important rule for the implementer of the service is to follow the *standards* of service development. Currently, there are two commonly used standards for developing services.

One of the service development standards is ***WSDL* (*Web Service Description Language*).** It is based on representing the contract of the service (its Web interface) in XML format in a specification language referred to as *WSDL*. A WSDL service works synchronously, so the client waits for one Web method call to finish before calling the other Web method. So this standard of consuming services works slower but is more reliable. In this standard, a service is *stateful* (remembers its state) and keeps information on the *session* of its consuming. The arguments of such service Web methods, objects of some types, are transferred via the network in *serialized* form – represented as a typed stream of bytes. One of the commonly used methods of serialization is XML; the other ones are offered by Java and .NET (on the .NET platform, the term *marshaling* is used as a synonym for the term *serialization*).

The other service development standard is ***REST* (*Representational State Transfer*)**. A common IT slang is to refer to such services as *RESTful*. With this standard, a Web method of the service is called *asynchronously* which is, generally speaking, faster, as compared to using a WSDL-based service. In addition, a RESTful service is *stateless*, so the information on its state, as well as the arguments of the Web method, is passed as parameters to the Web method call.

The developer of the services should have an opportunity to *publish* his or her Web services somewhere on the Web where the clients can find (*discover*) it.

Support of the SOA model is provided by a number of modern software tools and Web portals; for example,

- by *Microsoft SharePoint,* which is a simple-to-use toolkit to create extensible Web pages and Web services;
- by *UDDI* (*Universal Discovery, Description and Integration*) technology supported by Microsoft, available via the Web portal http://uddi.xml.org/ intended for publishing and discovering Web services.

From the viewpoint of the service-oriented model, cloud computing is the most up-to-date SOA model implementation. The cloud provides access to a set of its Web

services via the client browser. Publication of the newly developed cloud services is quite possible via the cloud. For example, on the Microsoft Azure cloud platform, the cloud Web interface provides a simple way to create a new empty Web service and then to implement it using some integrated development environment (IDE, e.g., *Visual Studio*) and to *publish* the newly implemented Web service in the cloud.

1.5 SOFTWARE AS A SERVICE (SaaS)

Software as a Service (*SaaS*) is a model of software development based on the use of *licenses software services on demand* by the tenants who purchase their licenses from service providers. Sometimes the SaaS model is referred to as *software-on-demand*.

The term *SaaS* originated in the late 1990s.

The main idea of SaaS is to use software on demand at low price, instead of purchasing full license to the software for all platforms.

The main characteristics of the SaaS model are as follows:

- Access to commercial software via the network
- Remote control of the software by the tenants via the central Web site
- The use of the "one-to-many" (*multitenant application*) model, that is, the use of one instance of software service by many tenants
- Centralized control over new versions and patches of the software services (the tenants can download new versions via the network)
- Continuing integration of software services into the hybrid set of software consumed by the tenant, as *mash-up*, that is, hybrid Web applications.

From this viewpoint, cloud computing corresponds to the principles of SaaS, in the following respects.

On the one hand, the principle of SaaS was used by cloud providers as the main principle of consuming the cloud services. The cloud client should *subscribe* to a set of cloud services for some period of time (e.g., for a year) at a reasonable price. This is exactly the principle of SaaS, as stated in the late 1990s, before the cloud era. Surely, each cloud provider also presents to any cloud client a trial or a learning period of completely free use of the cloud – typically, 1 month. At the end of this trial period the tenant can make a justified decision on cloud subscription.

On the other hand, the term SaaS is used as one of the models for organizing cloud services, when the full capability of the cloud is not necessary and what is needed is to integrate into the current mash-up of the tenant some new cloud features, for example, a new kind of the Web search engine.

1.6 KEY IDEAS AND PRINCIPLES OF CLOUD COMPUTING

Cloud computing is one of the most popular, hottest, fashionable directions of information technology in progress. The concept of cloud has already been associated long

ago with the metaphoric picture of the Internet that provides availability to a number of Web services. *Cloud computing* is a practical implementation of this idea, based on a structured collection of scalable and virtualized computing resources (software and data) available to the users via the Internet and implemented on the basis of powerful *data (processing) centers*. A cloud client entirely uses Web interface to the cloud enabled by a Web browser, and does not need any extra software to be installed on the client computing device for using the cloud.

Informally speaking, a cloud provided by some company is a good characteristic of this company. The cloud accumulates and expresses not only the technologies of the company but its spirit, trustworthiness, and its attitude also to the users "in one cloud cover."

The general structure of the cloud is depicted in Figure 1.2.

From the viewpoint of the users, there exist various kinds of clouds, as considered below.

Public cloud is a cloud model in which the cloud applications, cloud storage, and other cloud resources are available to any registered cloud user who pays for the cloud services. This model is the most prospective and the most convenient for users

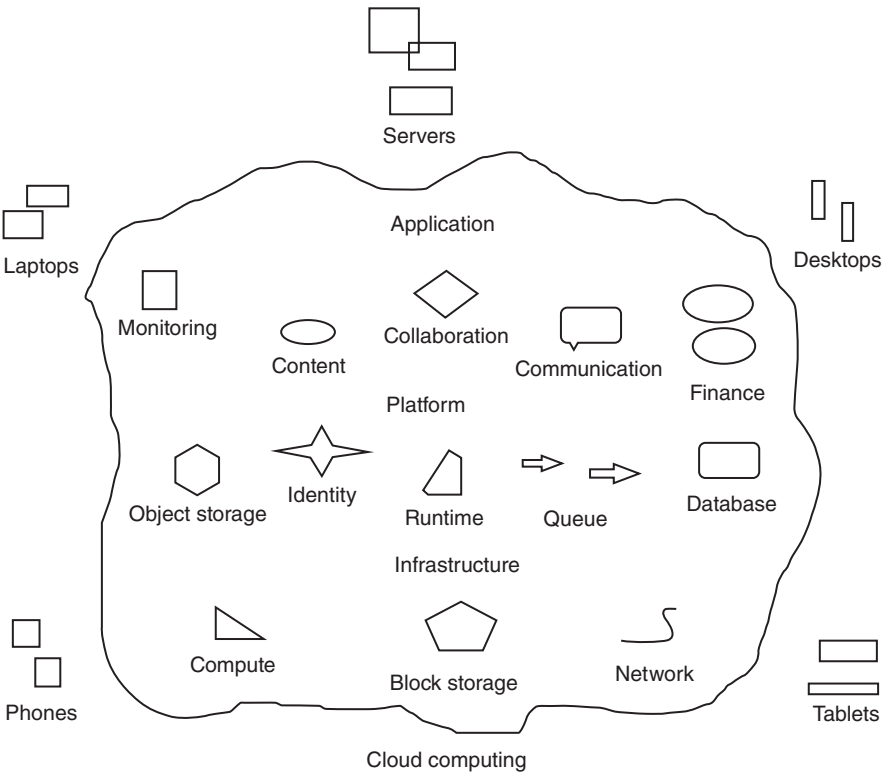

Figure 1.2 The general structure of the cloud.

but the most expensive to implement and the most resource consuming. Only a very large company can allow itself to develop and support such cloud model. The functioning of a public cloud is based on several big datacenters, each of which occupies a large building and consumes a lot of electric energy. An example of a public cloud is Microsoft Azure. Other examples are the Amazon Web Services cloud, Oracle cloud, and IBM Bluemix cloud.

Community cloud is a smaller cloud model in which the cloud infrastructure and services are available to some professional community. An example is the *Institute of Electrical and Electronics Engineers* (*IEEE*) community cloud. To use this cloud, the minimal requirement is to become an IEEE member.

Private cloud is a cloud model in which the cloud services are available only to the employees of some company. The development and maintenance of such a cloud is quite realistic for any company – even for a small one. I recommend the readers to start their own cloud development from creating a private cloud. Moreover, the providers of public clouds, such as Oracle, IBM, and Microsoft, provide support for the fast development of private clouds.

Hybrid cloud is a cloud model implementing a hybrid of several related public, community, or private clouds with the purpose of their joint use to solve some concrete tasks.

Clouds are offered by several companies (e.g., IBM, Oracle, Google, Microsoft, etc.) that serve as the *cloud (service) providers*. They provide, in the form of their clouds, structured collections of powerful computing resources, which the individual users typically do not have.

As a rule, the users must pay the cloud provider for the services of the cloud for a certain period of time (e.g., 1 year). There are also complimentary public cloud services, for example, those available on the Windows Live (http://www.live.com) portal.

Kinds of clouds in cloud computing (public, private and hybrid clouds) are illustrated in Figure 1.3.

No matter how prospective cloud computing is, there are some limitations and shortcomings of the cloud approach.

The first one is as follows. The user appears to be fully dependent on the cloud where the software and data consumed by the user are available and cannot directly control either the cloud computers in the datacenter, or even back up his or her data stored in the cloud. In this relation, there arise a lot of issues: security of cloud computing, keeping the privacy of the users' data, and so on. Some of those issues are far from their solution as yet.

The second group of serious problems of organizing cloud computing is related to managing the datacenters: their power consumption and their load balancing, since cloud computing with a public cloud inevitable leads to the need of serving many million users at each moment of time. For reasons of heavy power consumption, currently some companies, in spite of all the cloud computing perspectives, have even had to close their cloud datacenters, each occupying one or several big buildings of several thousand square feet.

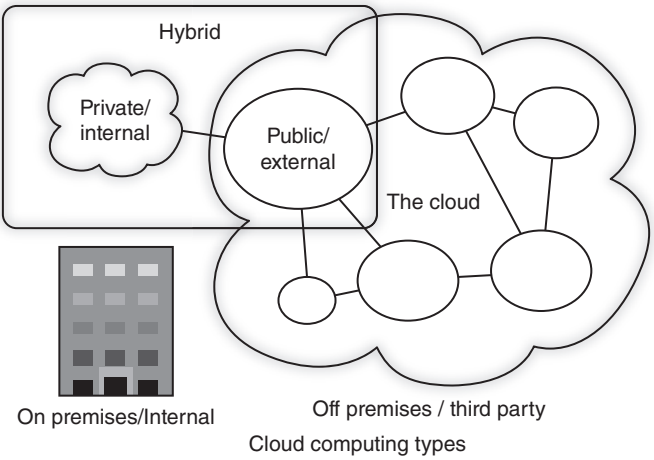

Figure 1.3 Kinds of clouds in cloud computing: public, private, and hybrid clouds.

1.7 COMPONENTS OF CLOUD PLATFORMS AND KINDS OF CLOUD SERVICING

Any cloud platform consists of the following main components:

- *applications* – the cloud software services available in the cloud;
- *runtimes* – the virtual and real machines available in the cloud that enable executing applications on some *platforms* (e.g., in Java or .NET runtime environments);
- *security and integration mechanisms*, including authentication, authorization, and encryption/decryption modules;
- *databases* – the databases available in the cloud that provide similar features as their non-cloud prototypes; for example, SQL Azure database management system available on the Microsoft Azure cloud platform, a cloud analog for Microsoft SQL Server;
- *servers* – computers with large volume of resources (memory, CPUs, and cores), controlled by server-side operating systems, for example, Windows Server or Linux; server hardware is usually clustered into tightly coupled groups;
- *virtualization tools* – software toolkits to support mechanisms of creating and using virtual machines, virtual networks, and other kinds of virtual resources; for example, Microsoft Hyper-V hypervisor and VMWare vSphere hypervisor; virtualization tools can be implemented in software and partly in hardware;
- *cloud user interface support* tools, in the form of *cloud management portal*; for example, Microsoft Silverlight plug-in for the Internet Explorer browser;

– *storage* – memory racks, networked hard disks, and other mass storage devices;
– *networking tools* – routers, hubs, and networking software.

Access to the cloud is enabled via the central *cloud management portal,* which first performs the cloud user authentication via login and password, and after the authentication, the user is redirected to the main cloud Web page displaying the cloud features available. The Web pages constituting the user interface of the cloud enable interactive help features for the users to better navigate and learn faster the architecture and features of the cloud.

For example, the cloud management portal on Microsoft Azure platform is http://manage.windowsazure.com. There is yet another Microsoft Azure portal, http://portal.azure.com, quite new at the moment of writing the book, at a preview stage. Both portals are linked together, so that it is possible to travel from the preview portal to the management portal.

This is just a general scheme of the cloud architecture; more details are given below. Different clouds may vary in their interface. User interfaces of the cloud portals tend to evolve and to change their appearance. For example, the cloud most familiar to me, Microsoft Azure (see Chapter 5), has changed the architecture and the appearance of its portal three times during the period 2011–2015.

What kinds of services are provided in the clouds? Let us consider their classification, which tends to evolve and to add the new "-as-a-Service" terms to the cloud terminology.

Based on the above scheme of cloud structure, from architectural viewpoint, clouds and the kinds of services can be classified as follows.

Private (On-Premises) cloud – a model of cloud servicing in which the cloud client (software developer) controls all of the cloud components outlined above.

Infrastructure-as-a-Service (IaaS) – in a cloud of this kind, the client (software developer) controls the applications, the runtimes, mechanisms of security and integration, and the databases. The rest of the components are controlled by the cloud provider. This is a model of cloud client servicing in which the cloud provider offers to the clients virtual machines and their resources: images of the disks, virtual networks, and so on. *Virtual cloud infrastructure* (at a small subscription payment, or just free) – this is what is the most valuable in cloud computing, one of the main reasons why so many clients started using the cloud.

Platform-as-a-Service (PaaS) – in a cloud of this kind, the client (software developer) controls only his or her own applications (cloud services of their own development); the rest of the cloud components are controlled by the cloud provider. This is a model of cloud client servicing in which the cloud provider offers to the clients the whole *computing platform*: an operating system, environment for running applications written in various programming languages, a database, and a Web server.

Software-as-a-Service (SaaS) – in a cloud of this kind, the client does not control any cloud components; everything in the cloud is controlled automatically by

the cloud provider; the software components in the cloud are ready to use and there is no need to develop the client's own cloud services. This is a model of cloud client servicing in which the cloud provider publishes in the cloud a number of useful applications used by the cloud clients. A good example is the cloud approach by Google – *Google Cloud Apps*, a set of useful applications that can be easily integrated into the browsers of the cloud clients.

Network-as-a-Service (NaaS) is a relatively new kind of cloud servicing model. In this model, the cloud provider offers to the cloud clients some kinds of *network services*: transport of the network; virtual private networks (VPNs); cloud solutions to unite computing devices into a secure network. An example of such a solution is *Windows Intune*, a cloud solution in Microsoft Azure cloud to create a network of personal computers and smartphones. Another example of NaaS is cloud-based email service, such as hotmail.com.

Resource-as-a-Service (RaaS) is a new kind of cloud servicing and selling cloud computing resources. In this model, instead of offering whole virtual machines for long periods of time as in IaaS clouds, the cloud provider offers to the cloud clients individual resources (such as CPU, memory, and I/O resources) for brief periods of time. The idea of this approach is to help cloud users save their financial resources. In IaaS cloud, a full virtual machine is provided for the cloud user, but the user has to fully pay for it, although he actually uses only some working cycles of that machine. The RaaS approach helps calculate the cloud rental payment more exactly.

Recovery-as-a-Service (also abbreviated as ***RaaS*****,** or ***DRaaS***, for ***Disaster Recovery as a Service***) is a new kind of cloud computing service to enable recovery of some application and data from disaster. The application or data suffering from disaster may run in some private datacenter. The cloud services in this case enable full recovery of the disrupted service or data in the cloud. An example of such a service is offered by VMWare as *vCloud* (http://vmware.com). The recovery service is available in a hybrid cloud. Generally speaking, there are three kinds of RaaS models:

- *To Cloud RaaS* – when the source to be recovered is in a private datacenter and the recovery services (backup or recovery target) are provided in the cloud;
- *In Cloud RaaS* – when both the resource to recover and the recovery services are in the cloud;
- *From Cloud RaaS* – when the source is in the cloud and the backup or recovery site is in the private datacenter.

Data-as-a-Service (DaaS) is a kind of cloud service in which the data files (texts, images, videos, sounds, etc.) are provided to the clients of the cloud on demand, regardless of their geographic locations. Examples of such cloud services are demonstrated by Oracle Cloud: http://cloud.oracle.com. Oracle provides the following kind of DaaS services:

– *DaaS for Marketing*
– *DaaS for Sales*
– *DaaS for Social*.

In addition, Oracle cloud provides access to Oracle Cloud Database, a cloud analog of the widely used Oracle DBMS.

1.8 LAYERS OF THE CLOUD ARCHITECTURE

In the architecture of the cloud, the following layers exist.

The client layer is the client software used for accessing the cloud services, typically, *a Web browser*, for example, Internet Explorer or Google Chrome.

The services layer is formed of the cloud Web services used via the cloud model, that is, via a structured collection of Web sites with some kind of specific URL addresses. For example, in Microsoft Azure cloud platform, a typical structure of the URL address of any cloud service is http://username.cloudapp.net where *username* is the service name given by the cloud user, the developer of the service.

The applications layer is composed of the programs available via the cloud that does not require installation on the client computer (as already emphasized above, this layer is one of the main advantages of the cloud model). An example is the publicly available portal http://www.live.com implementing cloud mail (please try it to feel the advantages of the cloud).

The platform layer is a software platform that provides a full set of tools for deployment and the use of cloud computing on a client computer without any extra installations or purchase of new hardware. The platform layer consists of *software development platforms* used in the cloud platform implementation (e.g., *.NET* and *Node.js* for the Microsoft Azure cloud platform), *IDEs* (e.g., Visual Studio) that enable *development of cloud services* and their *publication* in the cloud, and *plug-ins for cloud client browsers* implementing rich client user interface for cloud users (e.g., *Microsoft Silverlight*).

The storage layer supports storing the cloud user's data in the cloud and accessing them via the cloud. In fact, cloud data, as well as cloud services, are available via the specific cloud Web sites they are stored on, with the specific URL addresses characteristic of the cloud used. For example, cloud storage objects on Microsoft Azure platform are stored on Web sites with URL addresses of the kind http://storageObjectName.core.windows.net where *storageObjectName* is the specific name of an object in cloud storage.

The infrastructure layer is the layer that provides full virtualized infrastructure via the cloud. An example is the cloud portal http://manage.windowsazure.com of Microsoft Azure cloud platform. A user that logs in to the cloud is provided by a full-fledged cloud platform infrastructure controlled by rich-client style cloud user interface. The cloud infrastructure supports ways of creating and using Web sites, virtual machines, cloud services, cloud databases, cloud mobile services, cloud multimedia services, and many other kinds of interesting cloud objects. This infrastructure

is rapidly developing. Other cloud platforms, for example, Amazon Web Services cloud, provide similar opportunities.

1.9 SCHEME OF ARCHITECTURE OF THE CLOUD

A scheme of architecture of the cloud is illustrated in Figure 1.4.

The following components are depicted in the scheme:

- *Services* available via the cloud
- *Infrastructure* for their deployment and use
- *Platform* – a set of tools for using the cloud
- *Storage* – support of storing the users' data in the datacenter(s) implementing the cloud.

The components of the cloud are typically represented by Web services with their URL addresses. To continue the Microsoft Azure example, irrespective of whether the components of the cloud are cloud applications (services) or cloud data referenced by URL addresses of the kinds http://username.cloudapp.net (cloud application) or http://storageObjectName.core.windows.net (cloud storage), they all are implemented as specialized Web services providing access to cloud applications or cloud data, according to the Web services standards overviewed above. There are many other kinds of URL addresses for specific cloud objects in this scheme.

An example of cloud architecture is depicted in Figure 1.5.

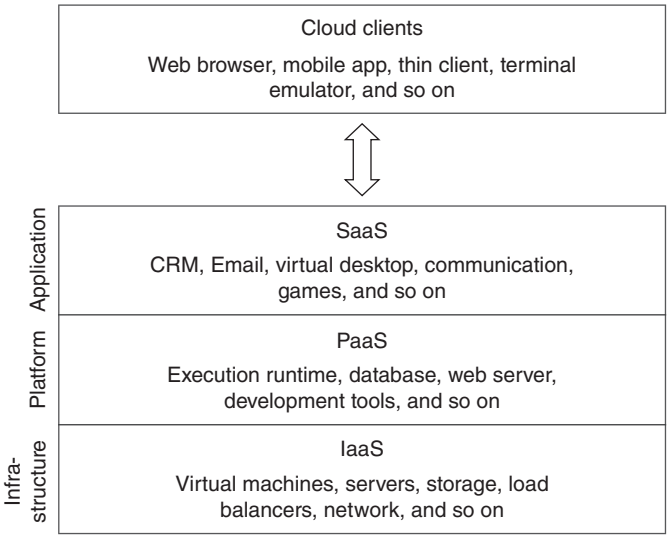

Figure 1.4 Scheme of architecture of the cloud.

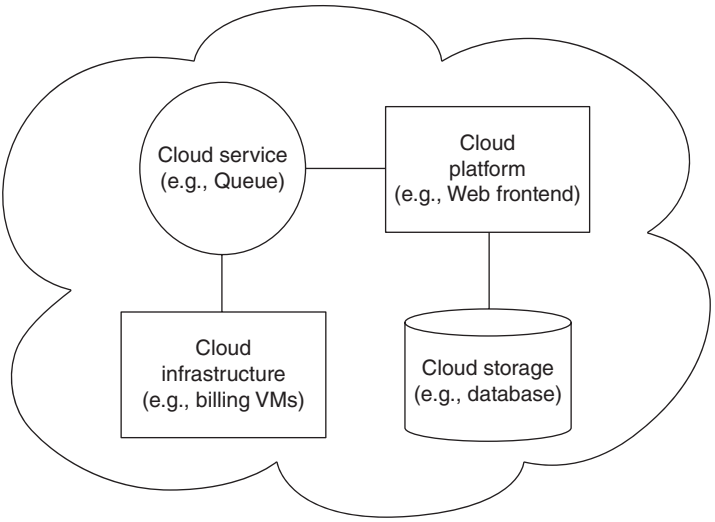

Figure 1.5 Example of cloud architecture.

In this example, the user may call some *cloud service*, for example, the one implementing the *Queue* concept. Tools to access this service are parts of the *cloud platform* that provides the *Web interface* (*Web frontend*) comfortable for accessing the service. Up-to-date cloud user interface, such as in Microsoft Azure or in other cloud platforms, is as comfortable as a typical user interface of local applications, so that the user will not feel any difference between cloud GUI and local application GUI. Via the cloud platform, the *cloud storage* is available (in Microsoft Azure, it is implemented by the *Storage* component), and also a *cloud database* (in Microsoft Azure represented by the *SQL Azure* component). Via cloud services, the whole cloud infrastructure is available, for example, *virtual machines*.

1.10 ROLES OF PEOPLE IN CLOUD COMPUTING

With the advent of cloud computing, the *roles* of the specialists participating in its development and use have changed, as compared to traditional "developer/user" paradigm.

Cloud providers are the companies that own the *datacenters* supporting their clouds. According to NIST [18], the major activities of the cloud provider are *service deployment*, *service orchestration* (arrangement, coordination, and management of cloud services to enable their optimal use)*, cloud service management, security,* and *privacy.* So the cloud provider is responsible to the cloud users for trustworthiness of the cloud services. The ISO/IEC Standards of cloud terminology [19] use the term *cloud service provider*.

Cloud users (clients) can be any users of the Internet who pay for the cloud services or use trial free subscription to the cloud, or use the private cloud of their company.

Hardware and software vendors for the cloud are the companies who develop hardware and basic software for datacenters. From this viewpoint, for example, Microsoft is the cloud provider for the Microsoft Azure cloud platform and is the major software vendor for this cloud platform. However, there are different hardware vendors and even different software vendors, since one of the server operating systems used in Microsoft Azure is *Linux,* which is not developed by Microsoft.

A *cloud architect* is the major developer of the cloud architecture.

A *cloud integrator* is a system administrator responsible for adding components to the cloud and updating them.

A *cloud auditor* [20] is a person or company who audits the cloud to examine the cloud *operations*, performance, and security against some criteria.

A *cloud service broker* [20] is an intermediary between cloud providers and cloud users, who negotiates the relationship between them.

A *cloud carrier* [18] is the intermediary that provides connectivity and transport of cloud services from cloud providers to cloud consumers.

Recently, a new role has emerged in cloud computing – *participant of a cloud community* of specialists who are interested in cloud computing, for example, *IEEE Cloud Computing Community*.

1.11 STANDARDS OF CLOUD COMPUTING

The cloud computing model and cloud implementation are based on the principles of following a whole set of standards.

Standards of cloud terminology, organization, and architecture are defined in [18–20]. I am trying to use the cloud terminology standards throughout the book; however, it does not appear to be stabilized yet.

There is a draft standard [21] for *InterCloud*, a "cloud of clouds," intended for the future integration of clouds provided by different companies. In this prospective role, the InterCloud standard is very important. However, currently the situation in cloud computing is far from the cloud integration stage. It can be characterized as active and aggressive development of many competitive cloud platforms.

For software components interaction in the cloud, the following standards are used, as explained subsequently.

HTTP – the hypertext transfer protocol, the basic networking protocol of the Web. The methods of HTTP have the format of *HTTP Method_Name URI_Address* and are as follows:

- *GET* – gets and displays in the client Web browser the requested HTML Web page (referenced by a URI address) from the Web server to the Web client
- *POST* – sends the filled out Web form from the Web client to the Web server
- *HEAD* – gets and displays in the client browser only the header of the requested Web page (without the body of the HTML page)
- *PUT* – uploads the content of the request to the URI address of the resource; if the resource with such URI is missing, it is created

- *PATCH* – updates a fragment of the resource, similar to *PUT*
- *DELETE* – deletes the given resource
- *TRACE* – traces the changes or additions of the intermediate servers (if any)
- *OPTIONS* – checks the functionality of the Web server by returning the HTTP methods the server supports
- *CONNECT* – converts the request connection to a transparent TCP/IP tunnel, to support SSL connection (*https*) via an unencrypted proxy.

At the end of any HTTP request, there can be a part starting with "?". After the "?" sign the Web client can provide parameters for the HTTP method performed by the Web server. For example, the request http://my/cloud/service?wsdl asks the given cloud service to return its Web methods if the service is implemented according to the WSDL standard.

The HTTP protocol is well suited for traditional non-grouped actions on the cloud, for example, for visualizing a cloud Web page. As for operations on *big data* (which are especially important now), the HTTP protocol is not fast enough and special; more efficient protocols should be used instead.

XML (extensible markup language) is used in cloud computing to represent Web configuration files (such as *Web.config* in .NET applications [14]), and to serialize data for their transfer via the network. One of the forms of networking data serialization in XML format is referred to as *SOAP* (*Simple Object Access Protocol*). In SOAP standard, objects are represented and transferred via the network in the form of specific XML files referred to as *envelopes* (the *soap:envelope* XML tag is used to distinguish between SOAP and other files). XML is used also to specify the interface of Web services in the format referred to as *WSDL* (see Section 1.2). However, SOAP and WSDL are slower to use, as compared to REST [22]; so cloud providers prefer to use REST in their cloud access APIs. The calls of RESTful APIs do not require XML format and are almost as simple as the basic methods of the HTTP protocol.

XMPP (Jabber) is one of the widely used standards to send and receive *instant messages* from one computing device (typically a laptop, tablet or smartphone) to another. The standard is based on using the XML format to represent *instant messages*, which corresponds well to the spirit and standards of the Web. The XMPP protocol plays a central role in the InterCloud standard [21]: according to this draft standard, the clouds in the near future should communicate and configure their interaction using XMPP. However, to my mind XMPP-based style of cloud interaction may work slow (especially when processing big data) because of its verbosity caused by using XML.

SSL (Secure Socket Layer) is a standard of the secure use of sockets, which is especially important to securely transfer confidential information (such as people names, credit card numbers, mobile phone numbers, and so on) via the Web in encrypted form. The SSL standard is used in the *https* protocol.

AJAX (Asynchronous JavaScript and XML) is a standard to efficiently use Web browsers when the number of the Web pages and possible redirections from one to the other may be big. Cloud computing is based on very intensive use of Web browsers,

so using AJAX on the client side can dramatically improve the Web connection performance. The AJAX standard and technology are based on the use of JavaScript and XML to reduce the amount of redirections between Web pages. AJAX implementation uses the idea for preliminary grouping of related sets of Web pages that are likely to be used together, and to transfer such a group via the Web by one GET command, instead of spending dozens of GETs for each individual page. The usability of AJAX for large Web applications can be confirmed by my own experience of developing commercial Web software products.

HTML 5 is the latest version of the hypertext markup language used on the Web, finally standardized in 2014. This new version is especially important for cloud computing, due to a number of new features of HTML 5: *offline clients* (including *offline databases*) used in cloud sessions; support of the use from *smartphones*; extended support to represent *multimedia information* (e.g., the new <video> and <audio> tags). These new features of HTML allow the users to refer to it as a special new version of HTML for cloud computing. HTML 5 is also considered as a potential candidate for cross-platform mobile applications. HTML 5 features were designed with considerations of being able to run on mobile devices (smartphones or PDAs) and on tablets. Also, the *Document Object Model* (*DOM*) that allows to represent documents as objects became an inherent part of HTML 5 specification (with previous versions of HTML, it was used just as some extension). So HTML 5 is suitable for use in cloud computing with mobile devices.

OMF (Open Media Framework) is a standard used in cloud computing for representing and transferring multimedia files, for example, video and audio.

OVF (Open Virtualization Format, or *Open Virtual Machine Format)* is an open standard for organization of *virtual machines* that play such a key role in cloud computing. The first version of the standard was developed by Microsoft, HP, Dell, IBM, and VMware in 2007, before the cloud era. The current version of the standard, OVF 2.0, accepted by the *Distributed Management Task Force* (*DMTF*), is targeted at cloud computing. The standard defines the structure of an *OVF package* that contains information on the packaged virtual machines. The OVF package contains an *OVF descriptor* – an XML file that describes the packaged virtual machine. The OVF package also contains *disk images* of the virtual machine, and may contain *certificate files* and other auxiliary files. The OVF format is approved by many software companies. For example, it is used in *Microsoft System Center Virtual Machine Manager*, in *IBM Smart Cloud*, and in *Oracle VM*.

Virtual Hard Disk (VHD) is a file format and standard used by Microsoft for representing virtual hard disks. The format was used since 2003 in *Microsoft Virtual PC* – the virtualization software product, a predecessor of *Microsoft Hyper-V (hypervisor)*. Currently the VHD format is used in Microsoft Azure cloud for representing virtual hard disks in Azure virtual machines, and for representing large multimedia files in Azure multimedia services.

REST (Representational State Transfer, see also Section 1.3 above) is a standard used in cloud computing to efficiently organize Web cloud services. As mentioned before, with such standard, information on the state of the Web service is passed via

the arguments and results of a Web method. Also, Web methods are called asynchronously. So the REST standard is a good basis for efficient use of cloud services. The advantages of REST for cloud computing are as follows [22]:

- REST uses a standard HTTP protocol without additional messaging layers that would make it more "heavyweight."
- REST uses URI addresses to access Web (cloud) resources.
- REST uses the standard set of HTTP operations on Web resources: *GET, POST, PUT, DELETE, and HEAD.*
- RESTful Web services are fully *stateless*, that is, they do not explicitly keep information on their state. This can be tested by restarting the cloud server and checking its availability using REST APIs.
- RESTful Web services support *caching* infrastructure using HTTP GET method (for most servers). This can improve the performance if the data the RESTful Web service returns is not changed frequently and is not dynamic in nature. For example, RESTful services are well suitable to communicate to a cloud database, which is not too big and which is not at generation stage. For example, using RESTful services to get from the cloud or transfer to the cloud some *big data* may appear not so efficient. The issues of using the cloud with big data are discussed in Ref [23] and will be considered later in Chapter 4.
- The RESTful service producer and service consumer need to keep a common understanding of the context as well as the content being passed along, since there is no standard set of rules to describe the RESTful Web services interface.
- REST is especially useful for restricted-profile devices such as smartphones and PDAs, for which the overhead of additional parameters such as headers and other SOAP elements are less.
- RESTful services are easy to integrate with the existing Web sites and are exposed with XML so that the HTML pages can consume the same with ease. There is no need to refactor the existing Web site architecture. This increases the productivity of software developers, since they will not have to rewrite everything from scratch and just need to add the existing functionality.

REST-based implementation is simpler, as compared to SOAP and WSDL.

1.12 HOW THE CLOUDS COME TRUE: ORGANIZATION OF DATACENTERS AND CLOUD HARDWARE

Now let us try to understand the specifics of cloud datacenters organization – how the clouds come true and how a cloud datacenter is organized.

The concept of *datacenter*, as a special facility providing computing, storage, and networking services, appeared long ago, in the 1980s, since the needs of IT industry required aggregation of big computing resources. For example, Microsoft founded

its first datacenter in 1989 in Redmond, Washington [24]. Very large datacenters are organized by many other big companies, for example, by Google and Facebook.

The question is how the *cloud* datacenters are different from the others, for example, from an IT enterprise-level datacenter belonging to some company? The distinction is in their *scale*. Cloud computing required a scalability that could hardly be imagined before. According to [24], an IT enterprise-level datacenter provides 10,000 seats (workplaces), whereas a cloud-level datacenter should provide 1,000,000 (a million) seats (workplaces) that can be geographically distributed worldwide. When you use, for example, the Microsoft Azure cloud, you can make your choice of placing your cloud service at any datacenter from the Washington state of the United States to Western Europe or Middle Asia. The order of magnitude of the number of computer servers in a typical public cloud datacenter can be one million. Similar is the situation with the public clouds provided by the other companies.

With such a giant number of computers, a different approach to availability and reliability of servers in a cloud datacenter should be used [24].

A classical approach to measuring reliability is the *Mean Time between Failures* (*MTBF*) – an average period of time between the subsequent failures of a hardware system (e.g., a server). However, with cloud-scale datacenters, this approach is impractical, since the sheer amount of hardware in cloud datacenters inevitably leads to possible hardware failures at each moment. So, the typical classical requirement of 99.9–100% availability of any server (with million servers running at a time) is not realistic. With the traditional approach of datacenter organization, it is not easy to quickly switch a software service to another, not faulty, hardware server.

So, yet another strategy is taken in cloud datacenters, based on another quantitative measure – *Mean Time to Recover* (*MTTR*). This quantity characterizes the average time for a cloud datacenter to recover from a hardware failure. The responsibility of choosing another suitable server to switch a software service from a faulty server to an up-and-running one is taken by the specialized software. For example, in Microsoft Azure cloud, such operations are performed by *Fabric Controller*, a stateful software application that manages software services and distributes them between the hardware servers.

So, the principle of cloud datacenter organization, instead of enabling high *reliability*, is to enable high *resilience* – the ability to quickly recover from hardware failures.

Another common issue in cloud datacenters is their big power consumption. In this relation, the following quantitative measure is used to estimate the efficiency of power use – *Power Usage Effectiveness* (*PUE*). It is calculated as the ratio of the *total facility power* to *IT equipment power*. The ideal value is 1.0, which is actually never reached. A typical datacenter in the United States has a PUE of 2.0 [24]. State-of-the-art datacenters now have PUEs of 1.12–1.2, with the industry average being 1.8 [25].

The following basic concepts are used to describe hardware in modern datacenters.

Rack is a group of related equipment (e.g., servers, telecommunication hardware) mounted in a single shelf. Rack is a typical way of organizing servers in large datacenters. A unit of rack height measurement is referred to as *U* (acronym for *Unit*). A 1 U rack is typically 19-inch (482.6 mm) or 23-inch (584.2 mm) high.

Blade server "is a stripped down server computer with a modular design optimized to minimize the use of physical space and energy. Whereas a standard rack-mount server can function with (at least) a power cord and network cable, blade servers have many components removed to minimize power consumption and other considerations, while still having all the functional components to be considered a computer" [26].

To mount racks together and to enable power supply and cooling for their hardware components, *chassis* are used. For example, in Microsoft cloud datacenters, 12 U-sized chassis are used, of original Microsoft design. Each of the 12 U chassis can accommodate up to 24 server blades – either *compute* or *storage*. The chassis enable efficient sharing of resources by multiple server nodes.

From higher level viewpoint, datacenters can be regarded as collections of hardware *clusters*. A cluster, generally speaking, is a group of tightly coupled computers working together as a single system. In a cloud datacenter, a cluster is a primary unit of hardware organization. In a Microsoft cloud datacenter, for example, each cluster consists of 20 racks, with a total of approximately 1000 rack-mounted blade servers (also called *nodes*). A cluster is a unit of fault isolation.

Each cluster is controlled by special software – *Fabric Controller*. For better availability, the Fabric Controller runs in five instances for each cluster. The role of the Fabric Controller is twofold: managing datacenter hardware and managing software cloud services. Actually, Fabric Controller is a kernel of the cloud operating system.

More details on Fabric Controller and its functions in Microsoft Azure cloud are provided in Chapter 5.

Detailed description of datacenter architectures is outside the scope of this book. Please see [24] for more details. The very good academic book [27] covers cloud datacenter hardware architecture in its Chapter 7.

1.13 SPECIFICS AND COMPONENTS OF SOFTWARE FOR CLOUD COMPUTING

As we have seen before, cloud computing is a new kind of client–server system. So, both client-side software and server-side software are used to implement cloud models. The client-side software components participating in cloud computing are as follows.

Cloud client operating systems. It is the operating system that controls the client hardware – a desktop, a tablet, a laptop computer, or a mobile device, such as a smartphone. There are some requirements and limitations for cloud client operating systems, although purely advertising texts on cloud computing are trying to persuade the users that "just an operating system is enough." Informally speaking, a client OS should be "novel enough." For example, it is not possible to use Microsoft Azure

cloud with old client operating systems such as Windows XP. For accessing the cloud, it is necessary to use Windows 10, Windows 8/8.1, or Windows 7 with Service Pack 1 on the client computer.

Cloud client Web browsers and rich client support plug-ins. As well as the client operating systems, the client Web browsers used by cloud clients should be new enough. For example, to access Microsoft Azure cloud, the version of Internet Explorer should be 11 or newer (at the moment of writing the book, March 2015). Older versions of the browser do not support cloud communication. Moreover, the client browser should have a special plug-in installed to enable rich client Web interface. This requirement is suggested to the cloud client during the first attempt at accessing the cloud. For the Internet Explorer browser, the kind of plug-in to be installed is *Microsoft Silverlight*.

***Developer clients' software tools and platforms for developing cloud services*.** Most cloud clients act purely as "end users" since they consume ready-to-use cloud services developed by cloud providers or third-party developer companies, already published in the cloud. But there are many active cloud users, especially students and researchers, who would like to extend the cloud by their own cloud services. Such users should take into account that the requirements of client computers, and the software to be used for cloud services *development* are much higher than just for ordinary cloud clients. The cloud stimulates creative activity of the users, due to the fact that it frees the users from mundane routine work. But to develop cloud services is a much more complicated work than to consume them. First, the client operating system must be novel enough to support cloud services. For example, Solaris 11, the new OS by Oracle, satisfies this requirement, since it is the first version of the Oracle OS to contain cloud computing support. As for Microsoft operating systems, only the latest OS – Windows 10, Windows 8 or Windows 8.1, or the enhanced version of the previous OS – Windows 7 with Service Pack 1 – are suitable for developing for the cloud. Second, to develop cloud services, there should be a suitable state-of-the-art *IDE* [7] based on a trustworthy underlying software platform. This IDE with the underlying software platform should support development of cloud Web services satisfying modern standards (such as REST or WSDL) and the principles of trustworthy computing [1]. A cloud service, as a server-side code, should be secure (resilient to attacks) and reliable (in particular, support type-safe data processing – the type of each object should be known and recognizable at runtime; there should be no memory leaks, etc.), and should guarantee the privacy of information processed. Not every software platform and programming language satisfies these requirements. For example, the C language is not suitable for cloud service development, since many of its inherent features, such as arbitrary casting from one type to another, contradict the principles of trustworthy computing and may cause hardly recognizable bugs and even potential security vulnerabilities. Actually, the most suitable software platforms for developing cloud services are Java and .NET. For example, .NET is the basic platform used for implementation of Microsoft Azure cloud. This platform should also be used for developing new cloud services for Microsoft Azure. The IDE most suitable for this purpose is Microsoft Visual Studio, more exactly, its latest version at the

moment of writing the book – *Visual Studio 2013, update 3* [7]. This version supports special kinds of projects (solutions) for cloud services development – *Windows Azure Projects* – and for developing mobile services and mobile applications that can communicate to mobile services published in the cloud. Also, the *Windows Azure SDK* should be installed on your computer. Windows Azure SDK is not included into Visual Studio by default; it should be installed separately. This is just an example illustrating the amount of preliminary work that should be done to make your client computer a suitable tool not only for using the cloud but also for extending it by new cloud services. Examples of working cloud services for Microsoft Azure developed with Visual Studio are given in the Appendix.

Similarly, to develop cloud services for another company's cloud – Oracle Java Cloud – the Java language and platform should be used, with the corresponding IDE from Oracle – *NetBeans* [28]. A plug-in to NetBeans should be installed that supports special kinds of projects for cloud services development in Java. Similarly to Azure, the *Oracle Java Cloud Service SDK* should be installed and used together with NetBeans to develop cloud services for Java cloud.

The server-side software for cloud computing is even more complicated, as shown below.

Cloud server-side operating system(s). The operating systems used on the computers in a cloud datacenter. The server-side operating systems, as well as the server-side hardware, are provisioned by the cloud provider company. For example, in Microsoft cloud datacenters that support Microsoft Azure cloud, the latest server-side operating system used is *Windows Server 2012 Release 2* that supports large-scale computing and virtualization. Also, Microsoft uses some dialects of Linux as server-side OS, such as Ubuntu and SUSE. In Oracle datacenters supporting Oracle cloud, *Solaris 11* is used as server-side OS, positioned as the first Oracle operating system supporting the cloud. In IBM datacenters supporting IBM cloud, *z* operating system is used as the server-side OS.

***Host and guest operating systems*.** One of the most attractive features of cloud computing, in accordance to the IaaS cloud model, is an opportunity to create and use a *virtual machine* or a set of virtual machines. The virtual machine available in the cloud can have an operating system different from the server-side OS used in the datacenter. The following terms are used to refer to those operating systems. *Host operating system* is the operating system used on real machines of the datacenter. *Guest operating system* is the operating system running on the virtual machine. For example, in Microsoft Azure cloud, it is possible to create and use a virtual machine whose guest operating system can be Windows 2012 R2 (with some slight changes, as compared to Windows 2012 R2 host OS), Windows 8.x, Windows 7.x, Linux Ubuntu, Linux SUSE, Oracle Solaris 11, and a number of others.

Hypervisors. To control virtual machines created and used in the cloud, a special software component is used, referred to as *hypervisor.* The term hypervisor (as hardware virtualization component) is being used from the 1970s when the first hypervisors were implemented in hardware. Now, different companies use different hypervisors in their cloud datacenters. For example, Microsoft uses *Hyper-V* hypervisor as part of their Windows Server 2012 R2 operating system. This hypervisor can

also be installed on a machine running, for example, Windows 8. The predecessor of Hyper-V developed in 2007 was called *Microsoft Virtual* PC. Another widely used hypervisor is *vSphere* by VMWare. IBM uses in its cloud datacenters the *WebSphere* infrastructure solutions to support virtualization.

Cloud management middleware. Software of this special kind is used in cloud datacenters to fulfill the following functions:

- Provisioning and de-provisioning hardware and software components; replacing the faulty components in the total cloud datacenter configuration
- Monitoring hardware components and software services
- Load balancing, that is, distributing cloud services between the machines in the cloud datacenter or in different datacenters supporting the same cloud
- Metering cloud usage, tracking costs, and billing.

For example, in IBM cloud datacenters, the *Tivoli* system software [29] is used as cloud management middleware. Tivoli Systems was a company acquired by IBM who developed this software product. It includes a storage manager, workload scheduler, network endpoint manager, automation manager, and a configuration and change management database.

In Microsoft practice, several kinds of software products are used as cloud management middleware. The above-mentioned Fabric Controller is an internal cloud software component that controls hardware clusters and blades, and also manages software cloud services, enabling basic hardware and software availability, reliability, and resilience. Also, *Microsoft System Center* is used in various control functions, with the following components: *Configuration Manager, Virtual Machine Manager, Mobile Device Manager, Operations Manager, Capacity Planner, Data Protection Manager*.

1.14 CLOUD COMPUTING-RELATED TRENDS, ACTIVITIES, AND RESOURCES

There is a great, rapidly emerging interest among IT specialists in cloud computing. To meet this growing interest, lots of cloud computing communities and conferences are organized, and more and more cloud computing journals are founded each year.

The most popular cloud computing journal is the *IEEE Transactions on Cloud Computing* [30]. Its advantage over many other cloud publications is the deep scientific analysis of cloud issues in the papers of this journal. To understand the specifics of this journal and its scope, let us consider the current call for papers for the oncoming special issues of the journal:

- Many-task computing in the cloud
- Cloud security engineering
- Mobile clouds.

Another IEEE publication – *IEEE Computer Magazine* [31] – pays much attention to cloud computing in the section referred to as *Cloud Cover,* although it covers the more general scope of themes. It contains small papers on challenges and news of cloud computing. For example, the paper [23] poses a serious issue of efficiency of using big data with the current clouds.

Yet another interesting new journal on cloud computing is the *Journal of Cloud Computing: Advances, Systems and Applications (JOCCASA).* This is a *Springer Open Journal*. The site of the journal is available at [32]. The journal publishes research articles on cloud computing accessible at no cost.

One of the leading journals in the field is the *International Journal on Cloud Computing* [33]. The journal publishes extended versions of papers from prestigious conferences on cloud computing, for example, from IEEE CLOUD.

What also deserves the readers' attention is the SYS-CON company's *Cloud Computing Journal* [34]. This company already publishes a number of the other software developers' style journals – such as *Java Developer Journal* and *.NET Developer Journal* (the latter journal has published five of my articles on .NET development and teaching and on aspect-oriented programming). Now they have founded their own cloud computing journal, which looks very promising.

Cloud computing communities. The largest of them is probably the *IEEE Cloud Computing Community* [35]. Membership in this community is recommended to any IT professional with interest in cloud computing. The community provides a lot of information by email (cloud news). Membership in this community is free. The community has established the IEEE portal on cloud computing [36]. The portal is a rich collection of cloud computing information on the following topics:

- Cloud computing conferences
- Cloud computing education (courses) and careers
- Cloud computing publications (especially IEEE journals)
- Cloud computing standards, including two new IEEE standards related to *Intercloud* and *Cloud Profiles* (now existing in the form of working drafts)
- Special section on the *InterCloud TestBed* international project aimed at cloud interoperability.

There are also cloud computing communities organized by big industrial companies – cloud providers.

For example, the *IBM Cloud Computing Community* portal is available at [37]. It offers considerable information on IBM cloud – webinars, demonstrations, news feeds, and publications.

The *Oracle Cloud Computing Community* portal [38] offers interesting cloud forums, podcasts, and blogs.

The *Microsoft Azure in Education* [39] portal publishes many educational resources on cloud computing and Microsoft Azure. They offer a number of educational courses on Microsoft Azure on its features and parts, for example, virtual machines, mobile services, and cloud services. Also, they have launched the

Microsoft Azure Educator Grant program. As a result, any university educator who teaches Azure in his or her course can apply for free access to Microsoft Azure cloud for the duration of the course, typically, one semester, for him (her) in person and for the students taking the course. This is a great opportunity to try the Azure cloud in action, to organize hands-on labs on it, and to develop a term or graduate student project based on Microsoft Azure. I have been one of the grateful participants of this program since 2011. Thanks a lot to Microsoft Research for this opportunity!

Conferences on cloud computing. As for the conferences and exhibitions on cloud computing, hundreds of them are held worldwide every month, almost every week, in any region of the world. Most famous cloud conferences in scientific research style are organized by IEEE. The leading one is the annual *IEEE CLOUD* conference. The portal of the IEEE CLOUD 2014 is available at [40]. The 2014 IEEE seventh International Conference on Cloud Computing (CLOUD 2014) is the leading theme conference for modeling, developing, publishing, monitoring, managing, and delivering *XaaS* (everything as a service) in the context of various types of cloud environments. This is a traditional scientific research multitrack conference. Extended versions of the accepted papers are published in leading cloud computing journals such as *IEEE Transactions on Cloud Computing*.

The Microsoft TechEd (*technical education*) conferences attract special interest. They are organized annually in each major region, for example, TechEd Europe, TechEd America, and TechEd Asia. These conferences include many deep technical talks on various subjects related to Microsoft technologies, and other Microsoft partner companies – Intel, HP, and so on. Hundreds of talks at TechEd conferences are devoted to the Microsoft Azure cloud platform and the related software products – Windows System Center, Windows Intune, and Visual Studio with its support of developing cloud services. The materials of the latest of TechEd Europe 2014 are published in [41]. I participated in Microsoft TechEd Europe 2013 in Madrid, which was very impressive, because of its wide scope, number of talks, and number of participants. The number of participants was about 3,600 and there were several hundred talks. Among the lecturers at the conference were such famous cloud experts as Mark Russinovich and Mark Minasi from Microsoft. I consider the latest TechEd conferences, for the first turn, a great school of cloud computing, though many other subjects have been covered at those conferences. In general, the materials of TechEd conferences are well suited as a teaching resource at universities, for courses, seminars, and hand-on labs on cloud computing.

A very helpful series of conferences similar to TechEd is organized by Microsoft Russia [42]. Those conferences are held on the same high technical level as the TechEd conferences. Technical talks are given by young Microsoft evangelists, both from Russia and from other countries. The latest such event was called “The Cloud in Russia” [43] and was organized by Microsoft Russia with very good quality presentations made by Microsoft Russia evangelists. I was watching this conference online for several hours with great pleasure, and there was much information helpful to me as a cloud expert. The site [43] contains video materials of the conference. I highly recommend them not only for my students but also for every IT student and specialist interested in Microsoft Azure (English translation is available).

Educational resources on cloud computing. There are a lot of educational resources on cloud computing. Besides books such as [8–12] discussed in the introduction, there are hundreds of other cloud computing books, white papers, help and support Web pages, e-books, and other educational resources issued by academics and by public cloud provider companies.

Surely each of the companies teaches its own cloud approach, so it is hardly possible to use such material as "independent" on cloud computing. But they contain short general introductions about cloud computing, what it is, and what the basic SaaS, PaaS, and IaaS approaches to cloud services model are. One of the very good examples of such free e-books is the book [44] on Microsoft Azure Fundamentals published online in 2014 by Microsoft Press. Another good source of educational and reference information on Microsoft Azure is *Microsoft Developer's Network* (*MSDN*) Web pages. They are numerous and the user can find information on any aspect of the Azure cloud. MSDN pages are greatly suitable for quick reference but not so comfortable for learning, because of the scattering of pieces of the valuable material into hundreds of small pages, which is typical of the hypertext approach.

To compare Microsoft's approach on cloud educational resources to those by other companies, let us consider the approach by Oracle. The Oracle cloud e-books portal [45] contains dozens of small e-books on various aspects of Oracle cloud, for example, architecture of the cloud portal, running Java Enterprise applications in the cloud, and so on. Each of the e-books is about 12–15 pages, so they are more suitable to give general information and impression on the corresponding Oracle cloud aspect than for deep diving into details. Some of the e-books are written in purely advertising style. In comparison, Microsoft's Azure fundamentals book [44] has 246 pages, and its style is quite suitable both for self-education and for organizing hand-on labs.

However, there are very good introductory cloud books by Oracle [46–48] written by M. Wessler and published by John Wiley & Sons in "Oracle special edition" series. Those books cover the basics of Oracle cloud [46], Oracle enterprise cloud architectures [47], and Oracle approach to virtualization [48]. The books can be used as generals tutorial on cloud computing, and as overview of principles of Oracle cloud architecture in particular. Especially valuable are the section on cloud management in [46], and the description of Oracle virtualization tools in [48].

A good example of the right approach to technical education in the cloud area is demonstrated by IBM. The free e-book [49] has 146 pages, is published online in July 2014, and is a good introduction to general concepts of cloud computing and to IBM cloud approach.

Also freely downloadable are educational e-books on HP cloud. HP offers a rich collection of commercial books [50] on HP cloud published by HP press. One such HP cloud book is *HP ATP – Data Center and Cloud and HP ASE – Data Center and Cloud Architect Official Certification Study Guide*, HP Press, 2014, by Pluta Ch., 532 pp. This book provides material for certification exam on cloud computing and HP cloud. To my knowledge, such cloud computing books are unique.

As for educational resources on Google cloud, there is a good new book [51] on Google Compute Engine, the Google cloud platform. It is a tutorial and can be used for self-education on cloud computing and Google cloud platform.

Cloud computing universities. A helpful site on cloud computing is *Cloud Computing Wire* [52]. It publishes various kinds of information on cloud computing and surely deserves subscribing to their weekly newsletter. Especially interesting for IT students is the list of *cloud computing universities* [52]. It currently contains 44 American universities offering courses on cloud computing. Among them are Harvard, Stanford, Berkeley, Carnegie-Mellon, and some other leading universities in the United States. Hopefully there will appear similar European portals on "cloud computing universities." As for our St. Petersburg University, I think we deserve to be added to this list of cloud computing universities, due to my above-mentioned publications – Internet courses and books [3–7] on cloud computing, and due to our 5-year experience of teaching students cloud computing. I hope this book will increase the popularity of our school of cloud computing, both teaching and research in this area.

How to organize cloud computing education. Surely books, courses, and hand-on labs on cloud computing are very helpful and just necessary for any university. Cloud computing should be an important part of university education, both at bachelor and masters levels. However, I was a bit surprised when I found information on some European universities offering the MS degree in cloud computing. As an experienced university professor in IT area, I consider cloud computing to be a very important part of IT but not suitable for issuing MS diplomas in that area. Actually, cloud computing is just one of the possible approaches to organizing computing resources, quite popular at the current moment. But cloud computing-oriented education and diplomas is to my mind too narrow an approach. A university graduate in IT area should be more universally educated, and should also know many important other modern parts of IT, such as trustworthy computing, scientific computing, mobile computing, and so on. So I think MS in Informatics, Computer Science, Software Engineering, or Information Technologies are more suitable MS university degrees than, for example, an MS degree on cloud computing, as well as on compilers, operating systems, and databases.

I hope Chapter 2 will be helpful to get more educational and practical information on a number of popular cloud platforms.

EXERCISES TO CHAPTER 1

E1.1 Please list the modern kinds of software you know, and explain the meaning and role of each of them.

E1.2 What is the essence of the client–server paradigm and what kind of servers in local networks do you know?

E1.3 What software platforms are used for developing Web applications?

E1.4 What are the specific issues and features of mobile applications?

E1.5 What kind of software is middleware and for which purposes is it used?

E1.6 What are the main goals of software for virtualization?

E1.7 Why are software tools for knowledge management so popular now, and which knowledge management toolkits do you know?

E1.8 Please formulate the most characteristic features of modern software.

E1.9 Please provide a definition of client in the client–server paradigm.

E1.10 Please provide a definition of server in the client–server paradigm.

E1.11 What is a thin client and what kind of user interface does it consume?

E1.12 What is a rich client and what are its advantages over a thin client?

E1.13 What is a layer in software architecture?

E1.14 What is an abstraction layer in software architecture?

E1.15 What is a vertical cut (or aspect) in software architecture?

E1.16 What is a tier in software architecture? Please give some examples of typical tiers.

E1.17 What is multitier architecture?

E1.18 What is multitenant architecture?

E1.19 What are the main principles of SOAs?

E1.20 What is the contract of a service?

E1.21 What is a Web method of a service?

E1.22 Is the implementation platform of a service significant to its consumer in the SOA model?

E1.23 Please list the most popular standards for implementation of a software service.

E1.24 Please define the main principles of the WSDL standard.

E1.25 Please define the main principles of the REST standard. Why do you think cloud providers prefer to provision REST APIs for their cloud services, rather than WSDL APIs?

E1.26 Which popular Web site development tools and Web sites provide support for SOA?

E1.26 What is SaaS? What was the original meaning of this term and which meaning has it acquired now, due to cloud computing?

E1.27 Please give a definition of cloud computing and your own understanding of this term.

E1.28 What kind of user interface does a cloud client use?

E1.29 Are any software installations on a cloud client computer necessary to use the cloud? What are the minimal requirements to cloud client computers?

E1.30 What is a cloud datacenter?

E1.31 What kinds of clouds do you know?

E1.32 What is a public cloud?

E1.33 What is a private cloud?

E1.34 What is a hybrid cloud?

E1.35 What is a community cloud?

E1.36 Please give examples of free public cloud services and sites. Practice in their use (e.g., Windows Live).

E1.37 Please describe the major shortcomings and issues of the cloud model, in your own opinion.

E1.38 Please list the main components of the cloud.

E1.39 What is the cloud management portal? Please give examples of the URL addresses of cloud management portals you know and use.

E1.40 Please list all kinds of "-as-a-Service" cloud servicing models you remember, and explain their meanings and functionalities.

E1.41 What are architectural layers of the cloud? What are their meanings and roles?

E1.42 What kind of reference is used in cloud computing to address a cloud service?

E1.43 What are the main roles of people participating in cloud computing?

E1.44 Which organizations developed standards for cloud computing?

E1.45 What is InterCloud?

E1.46 Please define the main methods of the HTTP protocol used in cloud computing.

E1.47 Please explain how XML is used in cloud computing.

E1.48 What is XMPP protocol and how is it used (or planned to be used) in cloud computing?

E1.49 What is SSL and what is its meaning for Web in general and for cloud computing in particular?

E1.50 What is AJAX and how does it help optimize access to the cloud?

E1.51 What is HTML 5 and which of its features are well suitable for cloud computing?

E1.52 What is OMF and how is it used in cloud computing?

E1.53 What is Open OVF and how is it used in cloud computing?

E1.54 What is VHD format and how does Microsoft use it in its Azure cloud?

E1.55 What is REST and what are its advantages over the other standards (SOAP and WSDL) for use in cloud computing?

E1.56 What is a cloud datacenter and in which respects is it different from ordinary IT enterprise level datacenter?

E1.57 What is the classical approach (MTBF) to reliability of technical systems and its quantitative measurement?

E1.58 What kind of approach to reliability and resilience (MTTR) is used for cloud datacenters to take into account their specifics?

E1.59 What kind of measure of the efficiency of electric power consuming is used for datacenters?

E1.60 What is rack in datacenters and what units are used for its size measurement?

E1.61 What is blade server in datacenters and what is its difference from ordinary server?

E1.62 What is chassis and how is it used in datacenters?

E1.63 What is cluster and which role does it play in cloud datacenters?

E1.64 What is Fabric Controller and what kind of functions does it perform in Microsoft Azure cloud datacenters?

E1.65 What kind of requirements apply to cloud client operating systems?

E1.66 How should the cloud client browsers be enhanced to use rich client Web interface? Please give concrete examples.

E1.67 What kinds of software tools, including IDEs, should be used for developing cloud services?

E1.68 Which IDE and in which version is recommended to use for Microsoft Azure cloud services development? Which software platform is recommended for such kind of development?

E1.69 Is the Visual Studio itself enough for cloud services development or is any extra software tools installation needed for that purpose?

E1.70 Which language, software platform, and IDE are recommended for Oracle cloud services development?

E1.71 Is the C programming language recommended for cloud services development? Why not?

E1.72 Which server-side operating systems are used in cloud datacenters by major public cloud providers?

E1.73 Please provide definitions for host operating system and guest operating system.

E1.74 What is hypervisor, which role does it play in cloud datacenter, and which hypervisors are used by major cloud providers?

E1.75 What kinds of cloud management middleware are used in cloud datacenters and which functions do they perform?

E1.76 Please list the cloud computing journals you know.

E1.77 Please list the cloud computing communities you know.

E1.78 Please list the cloud computing conferences you know and describe your impressions of them if you have attended such conferences.

E1.79 Which educational resources on the cloud offer major cloud providers?

E1.80 Which universities are included in the list of cloud computing universities? Is your university on this list?