

Topics in

Virtualization and Cloud Computing

Dr Balwinder Singh Sodhi

July, 2017

Preface

Virtualization and Cloud Computing are vast areas of study. Though there are several excellent reference books on these topics, most of them tend to have a narrow focus. In this book, I have tried to compile those Virtualization and Cloud Computing topics which are necessary to gain an in-depth understanding of the internals of these two important technologies. A majority of the topics in this book have been taken from the content which is freely available on the Web under a permissive license.

Balwinder Singh Sodhi

Ropar PB India 140001

Licensing Notice

This book is released under CC BY-SA License: <https://creativecommons.org/licenses/by-sa/3.0/>

Contents

1	Virtualization	1
1.1	Hardware virtualization	1
1.1.1	Snapshots	2
1.1.2	Migration	2
1.1.3	Failover	2
1.1.4	Video game console emulation	3
1.1.5	Licensing	3
1.2	Desktop virtualization	3
1.3	Other types	3
1.4	Nested virtualization	4
1.5	See also	5
1.6	References	5
2	Hardware virtualization	7
2.1	Concept	7
2.2	Reasons for virtualization	7
2.3	Full virtualization	8
2.4	Hardware-assisted virtualization	8
2.5	Partial virtualization	8
2.6	Paravirtualization	10
2.7	Operating-system-level virtualization	10
2.8	Hardware virtualization disaster recovery	10
2.9	See also	11
2.10	References	11
2.11	External links	12
3	Full virtualization	13
3.1	See also	14
3.2	References	15
3.3	External links	15
4	Paravirtualization	16
4.1	History	16

4.2	Linux paravirtualization support	17
4.3	See also	17
4.4	References	17
4.5	External links	17
5	Hypervisor	18
5.1	Classification	18
5.2	Mainframe origins	19
5.3	Unix and Linux servers	20
5.4	x86 systems	21
5.5	Embedded systems	21
5.6	Security implications	21
5.7	References	22
5.8	External links	22
6	Hardware-assisted virtualization	23
6.1	History	23
6.2	Pros	24
6.3	Cons	24
6.4	See also	24
6.5	References	25
6.6	Further reading	25
7	Emulator	26
7.1	Emulators in computing	26
7.2	Emulation in preservation	27
7.2.1	Benefits	27
7.2.2	Obstacles	28
7.3	Emulators in new media art	28
7.4	Emulation in future systems design	29
7.5	Types of emulators	29
7.6	Structure of an emulator	31
7.6.1	Memory subsystem	31
7.6.2	CPU simulator	31
7.6.3	I/O	32
7.7	Emulation versus simulation	33
7.8	Logic simulators	33
7.9	Functional simulators	33
7.10	Video game console emulators	33
7.11	Terminal emulators	34
7.12	In literature	34
7.13	Legal controversy	34

7.14 See also	34
7.15 Notes	35
7.16 References	35
7.17 External links	36
8 Snapshot (computer storage)	37
8.1 Rationale	37
8.2 Implementations	37
8.2.1 Volume managers	37
8.2.2 File systems	38
8.2.3 In databases	38
8.2.4 In virtualization	39
8.2.5 Other applications	39
8.3 See also	39
8.4 Notes	39
8.5 References	39
8.6 External links	39
9 Migration (virtualization)	40
9.1 Subjective effects	40
9.2 Objective effects	40
9.3 Relation to failover	41
9.4 References	41
10 Operating-system-level virtualization	42
10.1 Uses	42
10.1.1 Overhead	42
10.1.2 Flexibility	42
10.1.3 Storage	43
10.2 Implementations	43
10.3 See also	43
10.4 References	43
10.5 External links	44
11 Application virtualization	45
11.1 Description	45
11.2 Related technologies	46
11.3 Benefits of application virtualization	46
11.4 Limitations of application virtualization	47
11.5 See also	47
11.6 References	47
12 Portable application	49

12.1 Portable Windows applications	49
12.2 Portability on Linux and UNIX-like systems	50
12.3 Portable cross-platform applications	51
12.4 See also	51
12.5 References	52
13 Memory virtualization	53
13.1 Description	53
13.2 Benefits	53
13.3 Products	53
13.4 Implementations	54
13.4.1 Application level integration	54
13.4.2 Operating System Level Integration	54
13.5 Background	54
13.6 See also	55
13.7 References	56
14 Storage virtualization	57
14.1 Block virtualization	57
14.1.1 Address space remapping	57
14.1.2 Meta-data	58
14.1.3 I/O redirection	58
14.1.4 Capabilities	58
14.1.5 Replication	58
14.1.6 Pooling	59
14.1.7 Disk management	59
14.1.8 Benefits	59
14.1.9 Risks	60
14.1.10 Implementation approaches	61
14.2 File based virtualization	64
14.3 See also	64
14.4 References	65
14.5 External links	65
15 Network virtualization	66
15.1 Components	66
15.2 External virtualization	66
15.3 Internal virtualization	67
15.3.1 Examples	67
15.4 Use in testing	67
15.5 Wireless network virtualization	67
15.6 External links	67

15.7 See also	67
15.8 Further reading	68
15.9 References	68
16 Software-defined networking	69
16.1 History	69
16.2 Concept	70
16.3 Limitations of other networking technologies	71
16.4 The need for a new network architecture	72
16.5 Architectural components	73
16.6 SDN deployment models	74
16.7 Applications	74
16.8 Access control	75
16.9 See also	76
16.10References	76
16.11External links	77
17 Network Functions Virtualization	78
17.1 Background	78
17.2 History	78
17.3 NFV Framework	78
17.4 Practical aspects	79
17.5 Distributed NFV	79
17.6 NFV modularity benefits	79
17.7 Relationship to SDN	80
17.8 Industry impact	80
17.9 MANO - Management and Orchestration	80
17.10See also	81
17.11References	81
18 Cloud computing	82
18.1 Overview	83
18.2 History	84
18.2.1 Origin of the term	84
18.2.2 The 1950s	84
18.2.3 The 1990s	85
18.2.4 Since 2000	85
18.3 Similar concepts	85
18.4 Characteristics	86
18.5 Service models	87
18.5.1 Infrastructure as a service (IaaS)	87
18.5.2 Platform as a service (PaaS)	88

18.5.3	Software as a service (SaaS)	89
18.5.4	Unified Communications as a Service	89
18.6	Cloud clients	89
18.7	Deployment models	89
18.7.1	Private cloud	89
18.7.2	Public cloud	90
18.7.3	Hybrid cloud	90
18.7.4	Others	91
18.8	Architecture	91
18.8.1	Cloud engineering	92
18.9	Security and privacy	92
18.10	Cloud Marketplaces	93
18.11	Other services that require it	93
18.12	The future	93
18.13	See also	93
18.14	References	94
18.15	External links	97
19	Elasticity (cloud computing)	98
19.1	Example	98
19.2	Purpose	98
19.3	Problems	98
19.3.1	Resources provisioning time	99
19.3.2	Monitoring elastic applications	99
19.3.3	Elasticity requirements	99
19.3.4	Multiple levels of control	99
19.4	See also	99
19.5	References	99
19.6	External links	100
20	Platform as a service	101
20.1	Development and uses	101
20.2	Advantages and disadvantages	101
20.3	Types	102
20.3.1	Public, private and hybrid	102
20.3.2	Mobile PaaS	102
20.3.3	Open PaaS	102
20.3.4	PaaS for Rapid Development	102
20.3.5	System types	102
20.4	Providers	103
20.5	See also	103
20.6	References	104

21 Software as a service	105
21.1 History	105
21.2 Distribution	106
21.3 Pricing	106
21.4 Architecture	106
21.5 Characteristics	107
21.5.1 Configuration and customization	107
21.5.2 Accelerated feature delivery	107
21.5.3 Open integration protocols	107
21.5.4 Collaborative (and “social”) functionality	107
21.6 Adoption drivers	108
21.7 Adoption challenges	108
21.8 Emerging trends	109
21.9 Data escrow	109
21.10 Criticism	109
21.11 See also	110
21.12 References	110
22 Cloud computing issues	112
22.1 Threats and opportunities of the cloud	112
22.2 Privacy	112
22.2.1 Sharing information without a warrant	113
22.3 Privacy solutions	113
22.4 Compliance	114
22.5 Legal	114
22.6 Vendor lock-in	115
22.7 Open source	115
22.8 Open standards	115
22.9 Security	116
22.10 Sustainability	116
22.11 Abuse	117
22.12 IT governance	117
22.13 Consumer end storage	117
22.14 Ambiguity of terminology	117
22.15 Performance interference and noisy neighbors	117
22.16 Monopolies and privatization of cyberspace	118
22.17 See also	118
22.18 References	118
23 Cloud computing comparison	121
23.1 General information	121
23.2 Supported Hosts	121

23.3 Supported Clients	121
23.4 Providers	121
23.5 Features	121
23.6 See also	121
23.7 References	122
23.8 Text and image sources, contributors, and licenses	123
23.8.1 Text	123
23.8.2 Images	128
23.8.3 Content license	130

Chapter 1

Virtualization

Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources.

Virtualization began in 1960s mainframe computers as a method of logically dividing the system resources provided by mainframes between different applications. Since then, the meaning of the term has broadened.*[1]

1.1 Hardware virtualization

Main article: [Hardware virtualization](#)

See also: [Mobile virtualization](#)

Hardware virtualization or *platform virtualization* refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.*[2]*[3]

In hardware virtualization, the *host machine* is the actual machine on which the virtualization takes place, and the *guest machine* is the virtual machine. The words *host* and *guest* are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. The software or *firmware* that creates a virtual machine on the host hardware is called a *hypervisor* or *Virtual Machine Manager*.

Different types of hardware virtualization include:

- **Full virtualization** – almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
- **Partial virtualization** – some but not all of the target environment is simulated. Some guest programs, therefore, may need modifications to run in this virtual environment.
- **Paravirtualization** – a hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.

Hardware-assisted virtualization is a way of improving overall efficiency of virtualization. It involves CPUs that provide support for virtualization in hardware, and other hardware components that help improve the performance of a guest environment.

Hardware virtualization can be viewed as part of an overall trend in enterprise IT that includes *autonomic computing*, a scenario in which the IT environment will be able to manage itself based on perceived activity, and *utility computing*, in which computer processing power is seen as a utility that clients can pay for only as needed. The usual goal of virtualization is to centralize administrative tasks while improving *scalability* and overall hardware-resource

utilization. With virtualization, several operating systems can be run in parallel on a single central processing unit (CPU). This parallelism tends to reduce overhead costs and differs from multitasking, which involves running several programs on the same OS. Using virtualization, an enterprise can better manage updates and rapid changes to the operating system and applications without disrupting the user. “Ultimately, virtualization dramatically improves the efficiency and availability of resources and applications in an organization. Instead of relying on the old model of “one server, one application” that leads to underutilized resources, virtual resources are dynamically applied to meet business needs without any excess fat” (ConsonusTech).

Hardware virtualization is not the same as **hardware emulation**. In hardware emulation, a piece of hardware imitates another, while in hardware virtualization, a hypervisor (a piece of software) imitates a particular piece of computer hardware or the entire computer. Furthermore, a **hypervisor** is not the same as an **emulator**; both are computer programs that imitate hardware, but their domain of use in language differs.

1.1.1 Snapshots

Main article: [Snapshot \(computer storage\)](#)

A *snapshot* is the state of a virtual machine, and generally its storage devices, at an exact point in time. A snapshot enables the virtual machine's state at the time of the snapshot to be restored later, effectively undoing any changes that occurred afterwards. This capability is useful as a **backup** technique, for example, prior to performing a risky operation.

Virtual machines frequently use **virtual disks** for their storage; in a very simple example, a 10-gigabyte **hard disk drive** is simulated with a 10-gigabyte **flat file**. Any requests by the VM for a location on its physical disk are transparently translated into an operation on the corresponding file. Once such a translation layer is present, however, it is possible to intercept the operations and send them to different files, depending on various criteria. Every time a snapshot is taken, a new file is created, and used as an overlay for its predecessors. New data are written to the topmost overlay; reading existing data, however, needs the overlay hierarchy to be scanned, resulting in accessing the most recent version. Thus, the entire stack of snapshots is virtually a single coherent disk; in that sense, creating snapshots works similarly to the **incremental backup** technique.

Other components of a virtual machine can also be included in a snapshot, such as the contents of its random-access memory (RAM), BIOS settings, or its configuration settings. "Save state" feature in video game console emulators is an example of such snapshots.

Restoring a snapshot consists of discarding or disregarding all overlay layers that are added after that snapshot, and directing all new changes to a new overlay.

1.1.2 Migration

Main article: [Migration \(virtualization\)](#)

The snapshots described above can be moved to another host machine with its own hypervisor; when the VM is temporarily stopped, snapshotted, moved, and then resumed on the new host, this is known as migration. If the older snapshots are kept in sync regularly, this operation can be quite fast, and allow the VM to provide uninterrupted service while its prior physical host is, for example, taken down for physical maintenance.

1.1.3 Failover

Main article: [Failover](#)

Similar to teleportation above, failover allows the VM to continue operations if the host fails. However, in this case, the VM continues operation from the *last-known* coherent state, rather than the *current* state, based on whatever materials the backup server was last provided with.

1.1.4 Video game console emulation

Main article: [Video game console emulator](#)

A video game console emulator is a program that allows a [personal computer](#) or video game console to emulate a different video game console's behavior. Video game console emulators and [hypervisors](#) both perform hardware virtualization; words like “virtualization”, “virtual machine”, “host” and “guest” are not used in conjunction with console emulators.

1.1.5 Licensing

Virtual machines running proprietary operating systems require licensing, regardless of the host machine's operating system. For example, installing [Microsoft Windows](#) into a VM guest requires its licensing requirements to be satisfied.

1.2 Desktop virtualization

Main article: [Desktop virtualization](#)

Desktop virtualization is the concept of separating the [logical desktop](#) from the physical machine.

One form of desktop virtualization, virtual desktop infrastructure (VDI), can be thought of as a more advanced form of hardware virtualization. Rather than interacting with a host computer directly via a keyboard, mouse, and monitor, the user interacts with the host computer using another desktop computer or a mobile device by means of a network connection, such as a [LAN](#), [Wireless LAN](#) or even the [Internet](#). In addition, the host computer in this scenario becomes a [server computer](#) capable of hosting multiple virtual machines at the same time for multiple users.*[4]

As organizations continue to virtualize and converge their data center environment, [client architectures](#) also continue to evolve in order to take advantage of the predictability, continuity, and quality of service delivered by their [converged infrastructure](#). For example, companies like [HP](#) and [IBM](#) provide a hybrid VDI model with a range of virtualization software and delivery models to improve upon the limitations of [distributed client computing](#).*[5] Selected client environments move workloads from PCs and other devices to data center servers, creating well-managed virtual clients, with applications and client operating environments hosted on servers and storage in the data center. For users, this means they can access their desktop from any location, without being tied to a single client device. Since the resources are centralized, users moving between work locations can still access the same client environment with their applications and data.*[5] For IT administrators, this means a more centralized, efficient client environment that is easier to maintain and able to more quickly respond to the changing needs of the user and business.*[6]*[7]

Another form, session virtualization, allows multiple users to connect and [log into](#) a shared but powerful computer over the network and use it simultaneously. Each is given a desktop and a personal folder in which they store their files.*[4] With [multiseat configuration](#), session virtualization can be accomplished using a single PC with multiple monitors keyboards and mice connected.

[Thin clients](#), which are seen in desktop virtualization, are simple and/or cheap computers that are primarily designed to connect to the network. They may lack significant [hard disk storage space](#), [RAM](#) or even [processing power](#), but many organizations are beginning to look at the cost benefits of eliminating “thick client” desktops that are packed with software (and require software licensing fees) and making more strategic investments.*[8] Desktop virtualization simplifies software versioning and patch management, where the new image is simply updated on the server, and the desktop gets the updated version when it reboots. It also enables centralized control over what applications the user is allowed to have access to on the workstation.

Moving virtualised desktops into the cloud creates hosted virtual desktops (HVD), where the desktop images are centrally managed and maintained by a specialist hosting firm. Benefits include scalability and the reduction of capital expenditure, which is replaced by a monthly operational cost.*[9]

1.3 Other types

Software

- Operating system-level virtualization, hosting of multiple virtualized environments within a single OS instance.
- Application virtualization and workspace virtualization, the hosting of individual applications in an environment separated from the underlying OS. Application virtualization is closely associated with the concept of portable applications.
- Service virtualization, emulating the behavior of dependent (e.g., third-party, evolving, or not implemented) system components that are needed to exercise an application under test (AUT) for development or testing purposes. Rather than virtualizing entire components, it virtualizes only specific slices of dependent behavior critical to the execution of development and testing tasks.

Memory

- Memory virtualization, aggregating random-access memory (RAM) resources from networked systems into a single memory pool
- Virtual memory, giving an application program the impression that it has contiguous working memory, isolating it from the underlying physical memory implementation

Storage

- Storage virtualization, the process of completely abstracting logical storage from physical storage
- Distributed file system, any file system that allows access to files from multiple hosts sharing via a computer network
- Virtual file system, an abstraction layer on top of a more concrete file system, allowing client applications to access different types of concrete file systems in a uniform way
- Storage hypervisor, the software that manages storage virtualization and combines physical storage resources into one or more flexible pools of logical storage* [10]
- Virtual disk drive, a computer program that emulates a disk drive such as a hard disk drive or optical disk drive (see comparison of disc image software)

Data

- Data virtualization, the presentation of data as an abstract layer, independent of underlying database systems, structures and storage.
- Database virtualization, the decoupling of the database layer, which lies between the storage and application layers within the application stack over all.

Network

- Network virtualization, creation of a virtualized network addressing space within or across network subnets
- Virtual private network (VPN), a network protocol that replaces the actual wire or other physical media in a network with an abstract layer, allowing a network to be created over the Internet

1.4 Nested virtualization

Nested virtualization refers to the ability of running a virtual machine within another, having this general concept extendable to an arbitrary depth. In other words, nested virtualization refers to running one or more hypervisors inside another hypervisor. Nature of a nested guest virtual machine does not need not be homogenous with its host virtual machine; for example, application virtualization can be deployed within a virtual machine created by using hardware virtualization.* [11]

Nested virtualization becomes more necessary as widespread operating systems gain built-in hypervisor functionality, which in a virtualized environment can be used only if the surrounding hypervisor supports nested virtualization; for example, **Windows 7** is capable of running **Windows XP** applications inside a built-in virtual machine. Furthermore, moving already existing virtualized environments into a cloud, following the **Infrastructure as a Service (IaaS)** approach, is much more complicated if the destination IaaS platform does not support nested virtualization.*[12]*[13]

The way nested virtualization can be implemented on a particular **computer architecture** depends on supported **hardware-assisted virtualization** capabilities. In case a particular architecture does not provide hardware support required for nested virtualization, various software techniques are employed to enable it.*[12] Over time, more architectures gain required hardware support; for example, since the **Haswell** microarchitecture (announced in 2013), Intel started to include **VMCS shadowing** as a technology that accelerates nested virtualization.*[14]

1.5 See also

- Timeline of virtualization development
- Network Functions Virtualization
- Smarter Computing
- Emulation (computing)
- Computer simulation
- Numeronym (explains that “V12N” is an abbreviation for “virtualization”)
- Consolidation ratio
- I/O virtualization
- Application checkpointing

1.6 References

- [1] Graziano, Charles. “A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project” . Retrieved 2013-01-29.
- [2] Turban, E; King, D; Lee, J; Viehland, D (2008). “Chapter 19: Building E-Commerce Applications and Infrastructure” . *Electronic Commerce A Managerial Perspective*. Prentice-Hall. p. 27.
- [3] “Virtualization in education” . IBM. October 2007. Retrieved 6 July 2010. A virtual computer is a logical representation of a computer in software. By decoupling the physical hardware from the operating system, virtualization provides more operational flexibility and increases the utilization rate of the underlying physical hardware.
- [4] “Strategies for Embracing Consumerization” . Microsoft Corporation. April 2011. p. 9. Retrieved 22 July 2011.
- [5] Chernicoff, David, “HP VDI Moves to Center Stage,” ZDNet, August 19, 2011.
- [6] Baburajan, Rajani, “The Rising Cloud Storage Market Opportunity Strengthens Vendors,” infoTECH, August 24, 2011. It.tmcnet.com. 2011-08-24.
- [7] Oestreich, Ken, “Converged Infrastructure,” CTO Forum, November 15, 2010. Thectoforum.com.
- [8] “Desktop Virtualization Tries to Find Its Place in the Enterprise” . Dell.com. Retrieved 2012-06-19.
- [9] “HVD: the cloud's silver lining” . Intrinsic Technology. Retrieved 30 August 2012.
- [10] “Enterprise Systems Group White paper, Page 5” . Enterprise Strategy Group White Paper written and published on August 20, 2011 by Mark Peters.
- [11] Orit Wasserman, Red Hat (2013). “Nested virtualization: Shadow turtles” (PDF). KVM forum. Retrieved 2014-04-07.
- [12] Muli Ben-Yehuda; Michael D. Day; Zvi Dubitzky; Michael Factor; Nadav Har' El; Abel Gordon; Anthony Liguori; Orit Wasserman; Ben-Ami Yassour (2010-09-23). “The Turtles Project: Design and Implementation of Nested Virtualization” (PDF). *usenix.org*. Retrieved 2014-12-16.

- [13] Alex Fishman; Mike Rapoport; Evgeny Buditov; Izik Eidus (2013-06-25). “HVX: Virtualizing the Cloud” (PDF). *rackcdn.com*. Retrieved 2014-12-16.
- [14] “4th-Gen Intel Core vPro Processors with Intel VMCS Shadowing” (PDF). Intel. 2013. Retrieved 2014-12-16.

Chapter 2

Hardware virtualization

Computer **hardware virtualization** is the virtualization of computers or operating systems. It hides the physical characteristics of a computing platform from users, instead showing another abstract computing platform. ^[1]^[2] At its origins, the software that controlled virtualization was called a “control program”, but the terms “hypervisor” or “virtual machine monitor” are now preferred. ^[3]

2.1 Concept

The term “virtualization” was coined in the 1960s to refer to a *virtual machine* (sometimes called “pseudo machine”), a term which itself dates from the experimental IBM M44/44X system. The creation and management of virtual machines has been called “platform virtualization”, or “server virtualization”, more recently.

Platform virtualization is performed on a given hardware platform by *host* software (a *control program*), which creates a simulated computer environment, a *virtual machine* (VM), for its *guest* software. The guest software is not limited to user applications; many hosts allow the execution of complete operating systems. The guest software executes as if it were running directly on the physical hardware, with several notable caveats. Access to physical system resources (such as the *network access*, display, keyboard, and *disk storage*) is generally managed at a more restrictive level than the *host* processor and system-memory. Guests are often restricted from accessing specific *peripheral devices*, or may be limited to a subset of the device's native capabilities, depending on the hardware access policy implemented by the virtualization host.

Virtualization often exacts performance penalties, both in resources required to run the hypervisor, and as well as in reduced performance on the virtual machine compared to running native on the physical machine.

2.2 Reasons for virtualization

- In the case of *server consolidation*, many small physical servers are replaced by one larger physical server to increase the utilization of costly hardware resources such as CPU. Although hardware is consolidated, typically OSs are not. Instead, each OS running on a physical server becomes converted to a distinct OS running inside a virtual machine. The large server can “host” many such “guest” virtual machines. This is known as *Physical-to-Virtual* (P2V) transformation.
- Consolidating servers can also have the added benefit of reducing energy consumption. A typical server runs at 425 W^[4] and VMware estimates an average server consolidation ratio of 10:1. ^[5]
- A virtual machine can be more easily controlled and inspected from outside than a physical one, and its configuration is more flexible. This is very useful in kernel development and for teaching operating system courses. ^[6]
- A new virtual machine can be provisioned as needed without the need for an up-front hardware purchase.
- A virtual machine can easily be relocated from one physical machine to another as needed. For example, a salesperson going to a customer can copy a virtual machine with the demonstration software to his laptop,

without the need to transport the physical computer. Likewise, an error inside a virtual machine does not harm the host system, so there is no risk of breaking down the OS on the laptop.

- Because of the easy relocation, virtual machines can be used in disaster recovery scenarios.

However, when multiple VMs are concurrently running on the same physical host, each VM may exhibit a varying and unstable performance, which highly depends on the workload imposed on the system by other VMs, unless proper techniques are used for temporal isolation among virtual machines.

There are several approaches to platform virtualization.

Examples of virtualization scenarios:

- Running one or more applications that are not supported by the host OS: A virtual machine running the required guest OS could allow the desired applications to be run, without altering the host OS.
- Evaluating an alternate operating system: The new OS could be run within a VM, without altering the host OS.
- Server virtualization: Multiple virtual servers could be run on a single physical server, in order to more fully utilize the hardware resources of the physical server.
- Duplicating specific environments: A virtual machine could, depending on the virtualization software used, be duplicated and installed on multiple hosts, or restored to a previously backed-up system state.
- Creating a protected environment: if a guest OS running on a VM becomes damaged in a way that is difficult to repair, such as may occur when studying malware or installing badly behaved software, the VM may simply be discarded without harm to the host system, and a clean copy used next time.

2.3 Full virtualization

Main article: [Full virtualization](#)

In full virtualization, the virtual machine simulates enough hardware to allow an unmodified “guest” OS (one designed for the same instruction set) to be run in isolation. This approach was pioneered in 1966 with the IBM CP-40 and CP-67, predecessors of the VM family. Examples outside the mainframe field include Parallels Workstation, Parallels Desktop for Mac, VirtualBox, Virtual Iron, Oracle VM, Virtual PC, Virtual Server, Hyper-V, VMware Workstation, VMware Server (formerly GSX Server), QEMU, Adeos, Mac-on-Linux, Win4BSD, Win4Lin Pro, and Egenera vBlade technology.

2.4 Hardware-assisted virtualization

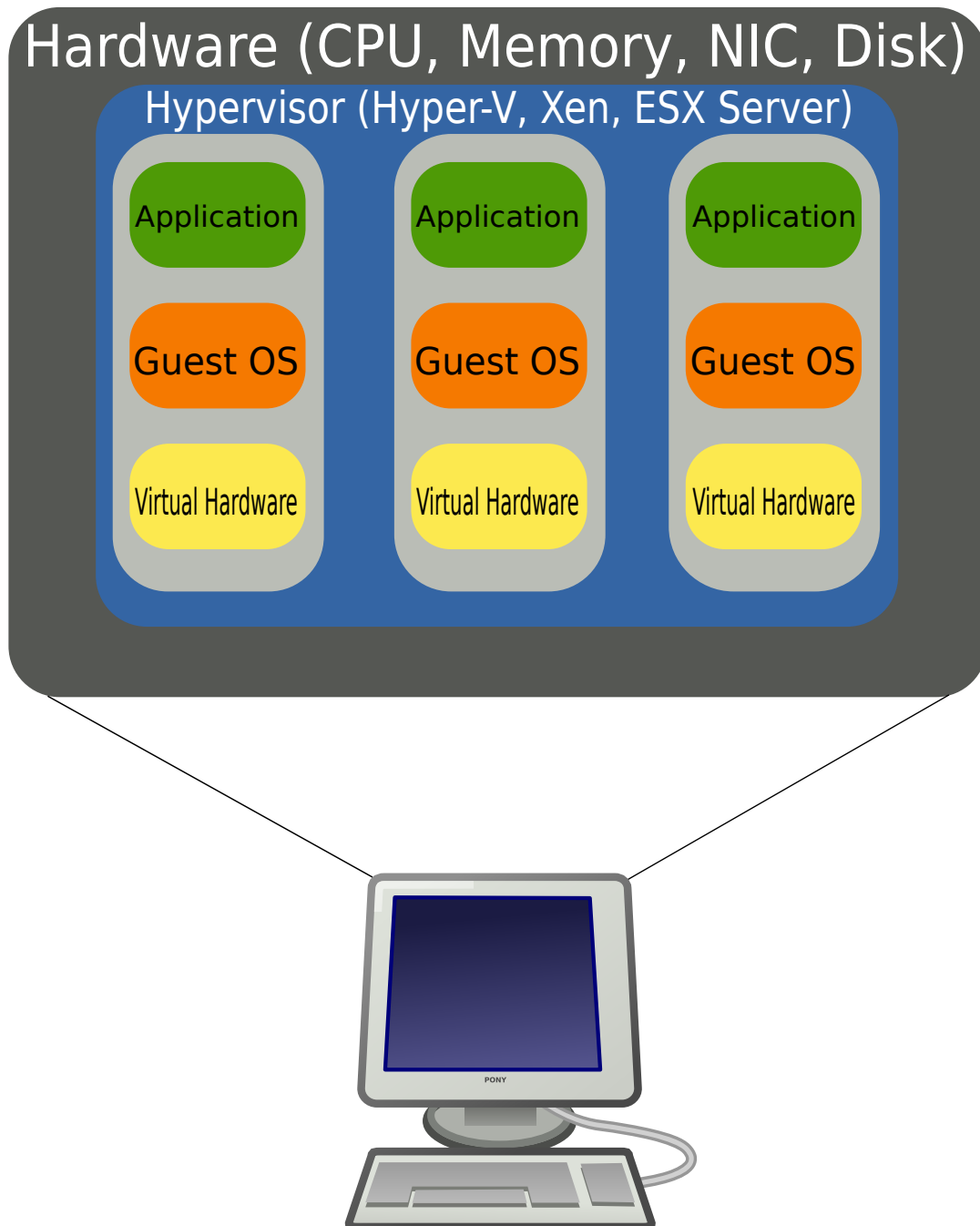
Main article: [Hardware-assisted virtualization](#)

In hardware-assisted virtualization, the hardware provides architectural support that facilitates building a virtual machine monitor and allows guest OSes to be run in isolation.* [7] Hardware-assisted virtualization was first introduced on the IBM System/370 in 1972, for use with VM/370, the first virtual machine operating system. In 2005 and 2006, Intel and AMD provided additional hardware to support virtualization. Sun Microsystems (now Oracle Corporation) added similar features in their UltraSPARC T-Series processors in 2005. Examples of virtualization platforms adapted to such hardware include Linux KVM, VMware Workstation, VMware Fusion, Microsoft Hyper-V, Microsoft Virtual PC, Xen, Parallels Desktop for Mac, Oracle VM Server for SPARC, VirtualBox and Parallels Workstation.

In 2006 first-generation 32- and 64-bit x86 hardware support was found rarely to offer performance advantages over software virtualization.* [8]

2.5 Partial virtualization

In partial virtualization, including address space virtualization, the virtual machine simulates multiple instances of much of an underlying hardware environment, particularly address spaces. Usually, this means that entire operating



Logical diagram of full virtualization.

systems cannot run in the virtual machine—which would be the sign of full virtualization—but that many applications can run. A key form of partial virtualization is address space virtualization, in which each virtual machine consists of an independent address space. This capability requires address relocation hardware, and has been present in most practical examples of partial virtualization.

Partial virtualization was an important historical milestone on the way to full virtualization. It was used in the first-generation time-sharing system CTSS, in the IBM M44/44X experimental paging system, and arguably systems like MVS and the Commodore 64 (a couple of 'task switch' programs). The term could also be used to describe any

operating system that provides separate address spaces for individual users or processes, including many that today would not be considered **virtual machine** systems. Experience with partial virtualization, and its limitations, led to the creation of the first full virtualization system (IBM's **CP-40**, the first iteration of **CP/CMS** which would eventually become IBM's **VM** family). (Many more recent systems, such as **Microsoft Windows** and **Linux**, as well as the remaining categories below, also use this basic approach.)

Partial virtualization is significantly easier to implement than full virtualization. It has often provided useful, robust virtual machines, capable of supporting important applications. Partial virtualization has proven highly successful for sharing computer resources among multiple users.

However, in comparison with full virtualization, its drawback is in situations requiring **backward compatibility** or **portability**. It can be hard to anticipate precisely which features have been used by a given application. If certain hardware features are not simulated, then any software using those features will fail.

2.6 Paravirtualization

Main article: [Paravirtualization](#)

In paravirtualization, the virtual machine does not necessarily simulate hardware, but instead (or in addition) offers a special API that can only be used by modifying the “guest” OS. For this to be possible, the “guest” OS's source code must be available. If the source code is available, it is sufficient to replace sensitive instructions with calls to VMM APIs (e.g.: “cli” with “vm_handle_cli()”), then re-compile the OS and use the new binaries. This system call to the hypervisor is called a “hypercall” in **TRANGO** and **Xen**; it is implemented via a **DIAG** (“diagnose”) hardware instruction in IBM's **CMS** under **VM** (which was the origin of the term *hypervisor*). Examples include IBM's **LPARs**,^[9] **Win4Lin 9x**, Sun's **Logical Domains**, **z/VM**, and **TRANGO**.

2.7 Operating-system-level virtualization

Main article: [Operating-system-level virtualization](#)

In operating-system-level virtualization, a physical server is virtualized at the operating system level, enabling multiple isolated and secure virtualized servers to run on a single physical server. The “guest” operating system environments share the same running instance of the operating system as the host system. Thus, the same **operating system kernel** is also used to implement the “guest” environments, and applications running in a given “guest” environment view it as a stand-alone system. The pioneer implementation was **FreeBSD jails**; other examples include **Solaris Containers**, **OpenVZ**, **Linux-VServer**, **LXC**, **AIX Workload Partitions**, **Parallels Virtuozzo Containers**, and **iCore Virtual Accounts**.

2.8 Hardware virtualization disaster recovery

A **disaster recovery (DR)** plan is good business practice for a hardware virtualization platform solution. DR of a virtualization environment can ensure high rate of availability during a wide range of situations that disrupt normal business operations. Continued operations of VMs is mission critical and a DR can compensate for concerns of hardware performance and maintenance requirements. A hardware virtualization DR environment involves hardware and software protection solutions based on business continuity needs, which include the methods described below.^[10]^[11]

Tape backup for software data long-term archival needs This common method can be used to store data offsite but can be a difficult and lengthy process to recover your data. Tape backup data is only as good as the latest copy stored. Tape backup methods will require a backup device and ongoing storage material.

Whole-file and application replication The implementation of this method will require control software and storage capacity for application and data file storage replication typically on the same site. The data is replicated on a different disk partition or separate disk device and can be a scheduled activity for most servers and is implemented more for database-type applications.

Hardware and software redundancy Ensures the highest level of disaster recovery protection for a hardware virtualization solution, by providing duplicate hardware and software replication in two distinct geographic areas.*[12]

2.9 See also

- Virtual appliance
- Application virtualization
- Virtualization for aggregation
- Instruction set simulator
- Workspace virtualization
- Desktop virtualization
- Comparison of platform virtual machines
- Dynamic infrastructure
- Popek and Goldberg virtualization requirements
- Physicalization

2.10 References

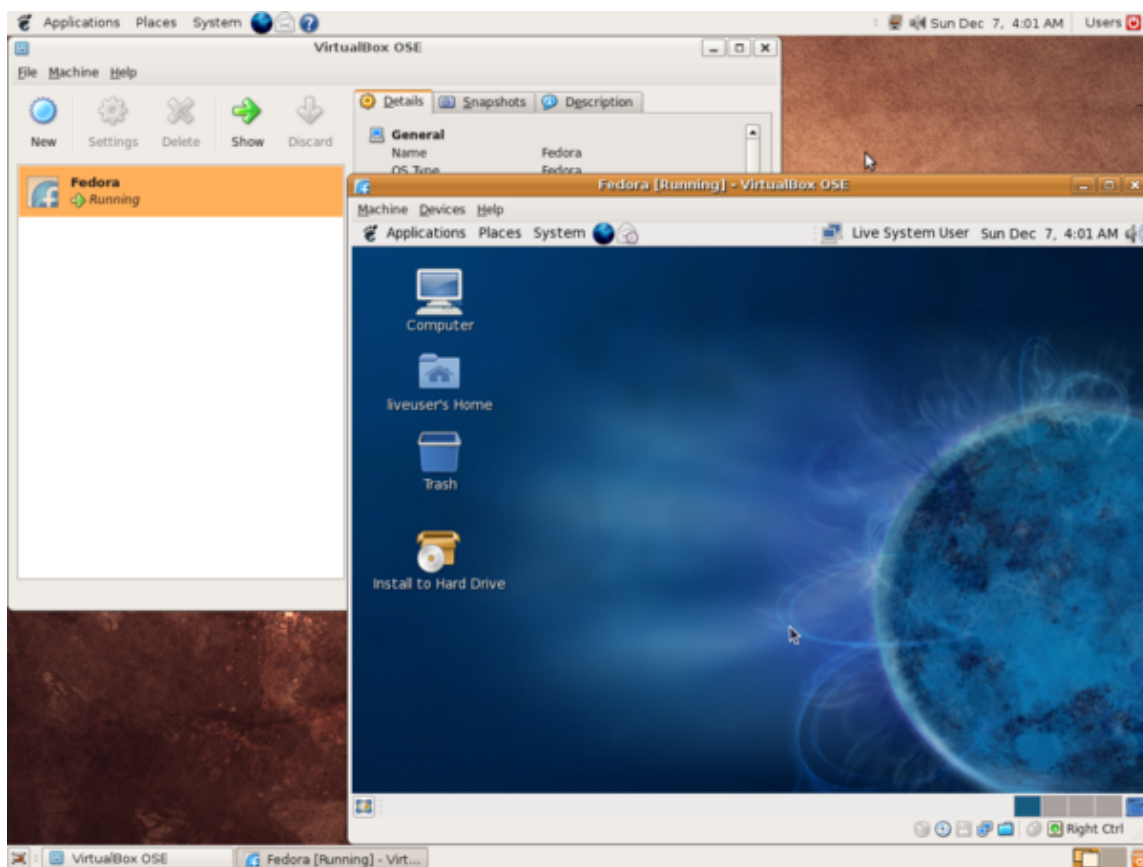
- [1] Turban, E; King, D.; Lee, J.; Viehland, D. (2008). “19” . *Electronic Commerce A Managerial Perspective* (5th ed.). Prentice-Hall. p. 27.
- [2] “Virtualization in education” . IBM. October 2007. Retrieved 6 July 2010.
- [3] Creasy, R.J. (1981). “The Origin of the VM/370 Time-sharing System” . IBM. Retrieved 26 February 2013.
- [4] Profiling Energy Usage for Efficient Consumption; Rajesh Chheda, Dan Shookowsky, Steve Stefanovich, and Joe Toscano
- [5] VMware server consolidation overview
- [6] Examining VMware Dr. Dobb’ s Journal August 2000 By Jason Nieh and Ozgur Can Leonard
- [7] Uhlig, R. et al.; “Intel virtualization technology,” *Computer* , vol.38, no.5, pp. 48-56, May 2005
- [8] A Comparison of Software and Hardware Techniques for x86 Virtualization, Keith Adams and Ole Agesen, VMWare, ASPLOS’ 06 21–25 October 2006, San Jose, California, USA “Surprisingly, we find that the first-generation hardware support rarely offers performance advantages over existing software techniques. We ascribe this situation to high VMM/guest transition costs and a rigid programming model that leaves little room for software flexibility in managing either the frequency or cost of these transitions.”
- [9] Borden, T.L. et al.; Multiple Operating Systems on One Processor Complex. *IBM Systems Journal*, vol.28, no.1, pp. 104-123, 1989
- [10] “The One Essential Guide to Disaster Recovery: How to Ensure IT and Business Continuity” . Vision Solutions, Inc. 2010.
- [11] Wold, G (2008). “Disaster Recovery Planning Process” .
- [12] “Disaster Recovery Virtualization Protecting Production Systems Using VMware Virtual Infrastructure and Double-Take” . VMWare. 2010.

2.11 External links

- [An introduction to Virtualization](#)
- [Xen and the Art of Virtualization](#)
- [Linux Virtualization Software](#)
- [Using a hypervisor to reconcile GPL and proprietary embedded code](#)
- [Server virtualization consolidation calculator](#)
- [Virtualization News, Info & Forums from Virtualization.net](#)

Chapter 3

Full virtualization



Screenshot of one virtualization environment

In computer science, **full virtualization** is a virtualization technique used to provide a certain kind of virtual machine environment, namely, one that is a complete simulation of the underlying hardware. Full virtualization requires that every salient feature of the hardware be reflected into one of several **virtual machines** – including the full instruction set, input/output operations, interrupts, memory access, and whatever other elements are used by the software that runs on the **bare machine**, and that is intended to run in a virtual machine. In such an environment, any software capable of execution on the raw hardware can be run in the virtual machine and, in particular, any operating systems. The obvious test of full virtualization is whether an operating system intended for stand-alone use can successfully run inside a virtual machine.

Other forms of **platform virtualization** allow only certain or modified software to run within a virtual machine. The concept of *full virtualization* is well established in the literature, but it is not always referred to by this specific term; see platform virtualization for terminology.

An important example of full virtualization was that provided by the control program of IBM's CP/CMS operating system. It was first demonstrated with IBM's CP-40 research system in 1967, then distributed via open source in CP/CMS in 1967-1972, and re-implemented in IBM's VM family from 1972 to the present. Each CP/CMS user was provided a simulated, stand-alone computer. Each such virtual machine had the complete capabilities of the underlying machine, and (for its user) the virtual machine was indistinguishable from a private system. This simulation was comprehensive, and was based on the *Principles of Operation* manual for the hardware. It thus included such elements as instruction set, main memory, interrupts, exceptions, and device access. The result was a single machine that could be multiplexed among many users.

Full virtualization is possible only with the right combination of hardware and software elements. For example, it was not possible with most of IBM's System/360 series with the exception being the IBM System/360-67; nor was it possible with IBM's early System/370 system until IBM added virtual memory hardware to the System/370 series in 1972.

Similarly, full virtualization was not quite possible with the x86 platform until the 2005-2006 addition of the AMD-V and Intel VT-x extensions (see x86 virtualization). Many platform virtual machines for the x86 platform came very close and claimed full virtualization even prior to the AMD-V and Intel VT-x additions. Examples include Adeos, Mac-on-Linux, Parallels Desktop for Mac, Parallels Workstation, VMware Workstation, VMware Server (formerly GSX Server), VirtualBox, Win4BSD, and Win4Lin Pro. VMware, for instance, employs a technique called binary translation to automatically modify x86 software on-the-fly to replace instructions that “pierce the virtual machine” with a different, virtual machine safe sequence of instructions; this technique provides the appearance of full virtualization.*[1]

A key challenge for full virtualization is the interception and simulation of privileged operations, such as I/O instructions. The effects of every operation performed within a given virtual machine must be kept within that virtual machine – virtual operations cannot be allowed to alter the state of any other virtual machine, the control program, or the hardware. Some machine instructions can be executed directly by the hardware, since their effects are entirely contained within the elements managed by the control program, such as memory locations and arithmetic registers. But other instructions that would “pierce the virtual machine” cannot be allowed to execute directly; they must instead be trapped and simulated. Such instructions either access or affect state information that is outside the virtual machine.

Full virtualization has proven highly successful for:

- sharing a computer system among multiple users;
- isolating users from each other (and from the control program);
- emulating new hardware to achieve improved reliability, security and productivity.

3.1 See also

- Platform virtualization
- CP/CMS
- Popek and Goldberg Virtualization Requirements
- Hardware-assisted virtualization
- Partial virtualization
- Paravirtualization
- Comparison of platform virtual machines
- Operating system-level virtualization
- LPAR
- PR/SM
- Hypervisor

- I/O virtualization
- Virtual machine

3.2 References

- [1] VMware (11 Sep 2007). “Understanding Full Virtualization, Paravirtualization, and Hardware Assist” (PDF). VMware. Retrieved 2007-12-09.

See specific sources listed under [platform virtualization](#) and (for historical sources) [CP/CMS](#).

3.3 External links

- [Compatibility is Not Transparency: VMM Detection Myths and Realities](#)

Chapter 4

Paravirtualization

In computing, **paravirtualization** is a virtualization technique that presents a software interface to virtual machines that is similar, but not identical to that of the underlying hardware.

The intent of the modified interface is to reduce the portion of the guest's execution time spent performing operations which are substantially more difficult to run in a virtual environment compared to a non-virtualized environment. The paravirtualization provides specially defined 'hooks' to allow the guest(s) and host to request and acknowledge these tasks, which would otherwise be executed in the virtual domain (where execution performance is worse). A successful paravirtualized platform may allow the **virtual machine monitor (VMM)** to be simpler (by relocating execution of critical tasks from the virtual domain to the host domain), and/or reduce the overall performance degradation of machine-execution inside the virtual-guest.

Paravirtualization requires the guest operating system to be explicitly ported for the para-API —a conventional OS distribution that is not paravirtualization-aware cannot be run on top of a paravirtualizing VMM. However, even in cases where the operating system cannot be modified, components may be available that enable many of the significant performance advantages of paravirtualization. For example, the Xen Windows GPLPV project provides a kit of paravirtualization-aware device drivers, licensed under the terms of the GPL, that are intended to be installed into a Microsoft Windows virtual-guest running on the Xen hypervisor.* [1]

4.1 History

Paravirtualization is a new term for an old idea. IBM's VM operating system has offered such a facility since 1972* [2] (and earlier as CP-67). In the VM world, this is referred to as a “DIAGNOSE code” , because it uses an instruction code used normally only by hardware maintenance software and thus undefined.

The Parallels Workstation operating system calls its equivalent a “hypercall” . All are the same thing: a system call to the hypervisor below. Such calls require support in the “guest” operating system, which has to have hypervisor-specific code to make such calls.

The term “paravirtualization” was first used in the research literature in association with the Denali Virtual Machine Manager.* [3] The term is also used to describe the Xen, L4, TRANGO, VMware, Wind River and XtratuM hypervisors. All these projects use or can use paravirtualization techniques to support high performance virtual machines on x86 hardware by implementing a virtual machine that does not implement the hard-to-virtualize parts of the actual x86 instruction set.* [4]

A hypervisor provides the virtualization abstraction of the underlying computer system. In full virtualization, a guest operating system runs unmodified on a hypervisor. However, improved performance and efficiency is achieved by having the guest operating system communicate with the hypervisor. By allowing the guest operating system to indicate its intent to the hypervisor, each can cooperate to obtain better performance when running in a virtual machine. This type of communication is referred to as paravirtualization.

In 2005, VMware proposed a paravirtualization interface, the Virtual Machine Interface (VMI), as a communication mechanism between the guest operating system and the hypervisor. This interface enabled transparent paravirtualization in which a single binary version of the operating system can run either on native hardware or on a hypervisor in paravirtualized mode. In September 2009, VMWare announced that VMI would be retired from future products.* [5]

4.2 Linux paravirtualization support

At the USENIX conference in 2006 in Boston, Massachusetts, a number of Linux development vendors (including IBM, VMware, Xen, and Red Hat) collaborated on an alternative form of paravirtualization, initially developed by the Xen group, called “paravirt-ops” .*[6] The paravirt-ops code (often shortened to pv-ops) was included in the mainline Linux kernel as of the 2.6.23 version, and provides a hypervisor-agnostic interface between the hypervisor and guest kernels. Distribution support for pv-ops guest kernels appeared starting with Ubuntu 7.04 and RedHat 9. Xen hypervisors based on any 2.6.24 or later kernel support pv-ops guests, as does VMware's Workstation product beginning with version 6.*[7]

4.3 See also

- Operating system-level virtualization
- Exokernel
- KVM
- Logical Domains
- Virtual Machine Interface (VMI)
- Hypervisor

4.4 References

- [1] “Installing signed GPLPV drivers in Windows Xen instances” . *Univention Wiki*. Retrieved 2013-04-10. The GPLPV driver is a driver for Microsoft Windows, which enables Windows DomU systems virtualised in Xen to access the network and block drivers of the Xen Dom0. This provides a significant performance and reliability gain over the standard devices emulated by Xen/Qemu/Kvm.
- [2] “VM History and Heritage” . IBM. Retrieved 2007-10-10.
- [3] A. Whitaker, M. Shaw, and S. D. Gribble, “Denali: Lightweight Virtual Machines for Distributed and Networked Applications” , Univ. of Washington Technical Report 02-02-01, (2002). (Available from Denali publications, technical reports, and talks)
- [4] Strobl, Marius (2013). *Virtualization for Reliable Embedded Systems*. Munich: GRIN Publishing GmbH. p. 54,63. ISBN 978-3-656-49071-5.
- [5] Update: Support for guest OS paravirtualization using VMware VMI to be retired from new products in 2010-2011 <http://blogs.vmware.com/guestosguide/2009/09/vmi-retirement.html>
- [6] <http://wiki.xenproject.org/wiki/XenParavirtOps>
- [7] <http://www.vmware.com/company/news/releases/050907PV.html>

4.5 External links

- Anandtech - Hardware Virtualization: the Nuts and Bolts Technical article on paravirtualization
- Virtualization.net Virtualization & Cloud Computing

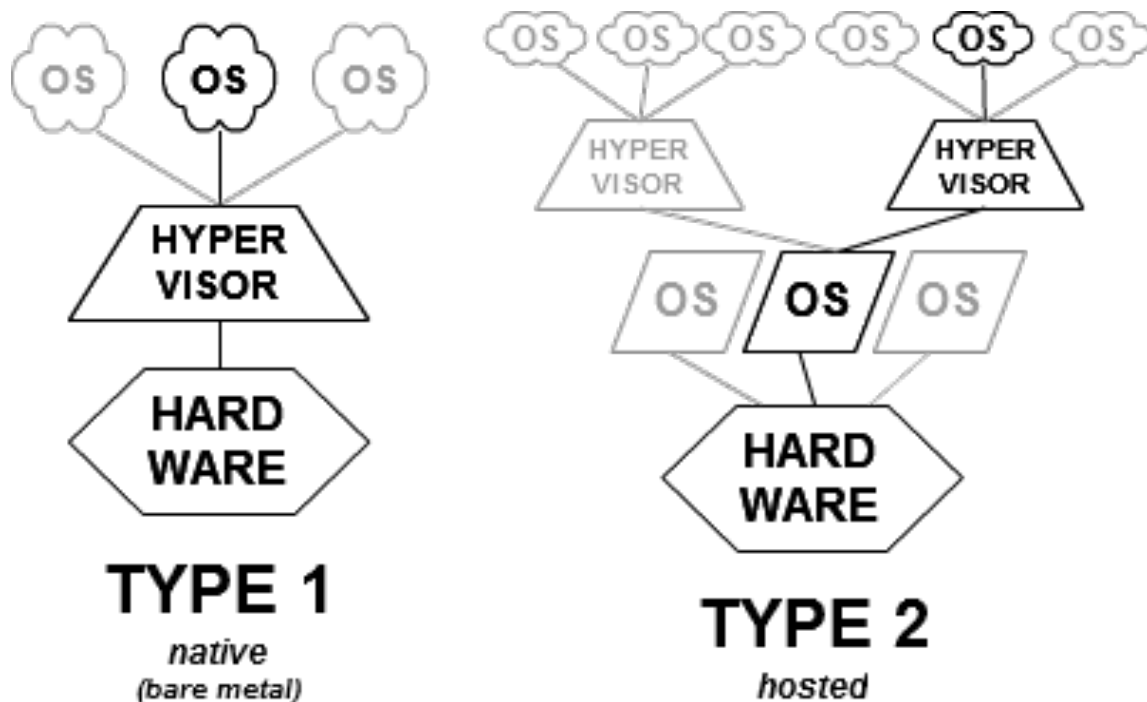
Chapter 5

Hypervisor

A **hypervisor** or **virtual machine monitor (VMM)** is a piece of computer software, firmware or hardware that creates and runs **virtual machines**.

A computer on which a hypervisor is running one or more virtual machines is defined as a *host machine*. Each virtual machine is called a *guest machine*. The hypervisor presents the guest operating systems with a **virtual operating platform** and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.

5.1 Classification



Type-1 and type-2 hypervisors

In their 1974 article “Formal Requirements for Virtualizable Third Generation Architectures” Gerald J. Popek and Robert P. Goldberg classified two types of hypervisor:*[1]

Type-1: native or bare-metal hypervisors These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. For this reason, they are sometimes called bare metal hypervisors. A guest operating system runs as a process on the host. The first hypervisors, which IBM developed

in the 1960s, were native hypervisors. These included the test software **SIMMON** and the **CP/CMS** operating system (the predecessor of IBM's **z/VM**). Modern equivalents include Oracle **VM Server for SPARC**, Oracle **VM Server for x86**, the Citrix **XenServer**, **VMware ESX/ESXi** and Microsoft **Hyper-V 2008/2012**.

Type-2: hosted hypervisors These hypervisors run on a conventional operating system just as other computer programs do. Type-2 hypervisors abstract guest operating systems from the host operating system. **VMware Workstation** and **VirtualBox** are examples of type-2 hypervisors.

However, the distinction between these two types is not necessarily clear. Linux's **Kernel-based Virtual Machine (KVM)** and **FreeBSD's bhyve** are kernel modules* [2] that effectively convert the host operating system to a type-1 hypervisor.* [3] Nevertheless, since Linux distributions and FreeBSD are still general-purpose operating systems, with other applications competing for VM resources, KVM and bhyve can also be categorized as type-2 hypervisors.* [4]

In 2012, a US software development company called **LynuxWorks** proposed a type-0 (zero) hypervisor—one with no kernel or operating system whatsoever* [5] [6]—which might not be entirely possible.* [7]

5.2 Mainframe origins

The first hypervisors providing full virtualization were the test tool **SIMMON** and IBM's one-off research **CP-40** system, which began production use in January 1967, and became the first version of IBM's **CP/CMS** operating system. **CP-40** ran on a **S/360-40** that was modified at the **IBM Cambridge Scientific Center** to support **Dynamic Address Translation**, a key feature that allowed virtualization. Prior to this time, computer hardware had only been virtualized enough to allow multiple user applications to run concurrently (see **CTSS** and **IBM M44/44X**). With **CP-40**, the hardware's *supervisor state* was virtualized as well, allowing multiple operating systems to run concurrently in separate virtual machine contexts.

Programmers soon re-implemented **CP-40** (as **CP-67**) for the **IBM System/360-67**, the first production computer-system capable of full virtualization. IBM first shipped this machine in 1966; it included page-translation-table hardware for virtual memory, and other techniques that allowed a full virtualization of all kernel tasks, including I/O and interrupt handling. (Note that its “official” operating system, the ill-fated **TSS/360**, did not employ full virtualization.) Both **CP-40** and **CP-67** began production use in 1967. **CP/CMS** was available to IBM customers from 1968 to 1972, in source code form without support.

CP/CMS formed part of IBM's attempt to build robust time-sharing systems for its mainframe computers. By running multiple operating systems concurrently, the hypervisor increased system robustness and stability: Even if one operating system crashed, the others would continue working without interruption. Indeed, this even allowed beta or experimental versions of operating systems – or even of new hardware* [8] – to be deployed and debugged, without jeopardizing the stable main production system, and without requiring costly additional development systems.

IBM announced its **System/370** series in 1970 without any virtualization features, but added them in the August 1972 Advanced Function announcement. Virtualization has been featured in all successor systems (all modern-day IBM mainframes, such as the **zSeries** line, retain backwards-compatibility with the 1960s-era IBM **S/360** line) The 1972 announcement also included **VM/370**, a reimplementation of **CP/CMS** for the **S/370**. Unlike **CP/CMS**, IBM provided support for this version (though it was still distributed in source code form for several releases). **VM** stands for *Virtual Machine*, emphasizing that all, and not just some, of the hardware interfaces are virtualized. Both **VM** and **CP/CMS** enjoyed early acceptance and rapid development by universities, corporate users, and time-sharing vendors, as well as within IBM. Users played an active role in ongoing development, anticipating trends seen in modern open source projects. However, in a series of disputed and bitter battles, time-sharing lost out to batch processing through IBM political infighting, and **VM** remained IBM's “other” mainframe operating system for decades, losing to **MVS**. It enjoyed a resurgence of popularity and support from 2000 as the **z/VM** product, for example as the platform for **Linux for zSeries**.

As mentioned above, the **VM** control program includes a *hypervisor-call* handler that intercepts **DIAG** (“Diagnose”) instructions used within a virtual machine. This provides fast-path non-virtualized execution of file-system access and other operations (**DIAG** is a model-dependent privileged instruction, not used in normal programming, and thus is not virtualized. It is therefore available for use as a signal to the “host” operating system). When first implemented in **CP/CMS** release 3.1, this use of **DIAG** provided an operating system interface that was analogous to the **System/360 Supervisor Call** instruction (**SVC**), but that did not require altering or extending the system's virtualization of **SVC**.

In 1985 IBM introduced the **PR/SM** hypervisor to manage logical partitions (**LPAR**).

5.3 Unix and Linux servers

Several factors led to a resurgence around 2005 in the use of virtualization technology among Unix and Linux server vendors.*[9]

- Expanding hardware capabilities, allowing each single machine to do more simultaneous work
- Efforts to control costs and to simplify management through consolidation of servers
- The need to control large multiprocessor and cluster installations, for example in server farms and render farms
- The improved security, reliability, and device independence possible from hypervisor architectures
- The ability to run complex, OS-dependent applications in different hardware or OS environments

Major Unix vendors, including Sun Microsystems, HP, IBM, and SGI, have been selling virtualized hardware since before 2000. These have generally been large systems with hefty, server-class price-tags (in the multi-million dollar range at the high end), although virtualization has also been available on some low- and mid-range systems, such as IBM's pSeries servers, Sun/Oracle's T-series CoolThreads servers and HP Superdome series machines.

Although Solaris has always been the only guest domain OS officially supported by Sun/Oracle on their Logical Domains hypervisor, as of late 2006, Linux (Ubuntu and Gentoo), and FreeBSD have been ported to run on top of the hypervisor (and can all run simultaneously on the same processor, as fully virtualized independent guest OSes). Wind River "Carrier Grade Linux" also runs on Sun's Hypervisor.*[10] Full virtualization on SPARC processors proved straightforward: since its inception in the mid-1980s Sun deliberately kept the SPARC architecture clean of artifacts that would have impeded virtualization. (Compare with virtualization on x86 processors below.)*[11]

HP calls its technology to host multiple OS technology on its Itanium powered systems "Integrity Virtual Machines" (Integrity VM). Itanium can run HP-UX, Linux, Windows and OpenVMS. Except for OpenVMS, to be supported in a later release, these environments are also supported as virtual servers on HP's Integrity VM platform. The HP-UX operating system hosts the Integrity VM hypervisor layer that allows for many important features of HP-UX to be taken advantage of and provides major differentiation between this platform and other commodity platforms - such as processor hotswap, memory hotswap, and dynamic kernel updates without system reboot. While it heavily leverages HP-UX, the Integrity VM hypervisor is really a hybrid that runs on bare-metal while guests are executing. Running normal HP-UX applications on an Integrity VM host is heavily discouraged, because Integrity VM implements its own memory management, scheduling and I/O policies that are tuned for virtual machines and are not as effective for normal applications. HP also provides more rigid partitioning of their Integrity and HP9000 systems by way of VPAR and nPar technology, the former offering shared resource partitioning and the latter offering complete I/O and processing isolation. The flexibility of virtual server environment (VSE) has given way to its use more frequently in newer deployments.

IBM provides virtualization partition technology known as logical partitioning (LPAR) on System/390, zSeries, pSeries and iSeries systems. For IBM's Power Systems, the Power Hypervisor (PowerVM) functions as a native (bare-metal) hypervisor in firmware and provides isolation between LPARs. Processor capacity is provided to LPARs in either a dedicated fashion or on an entitlement basis where unused capacity is harvested and can be re-allocated to busy workloads. Groups of LPARs can have their processor capacity managed as if they were in a "pool" - IBM refers to this capability as Multiple Shared-Processor Pools (MSPPs) and implements it in servers with the POWER6 processor. LPAR and MSPP capacity allocations can be dynamically changed. Memory is allocated to each LPAR (at LPAR initiation or dynamically) and is address-controlled by the POWER Hypervisor. For real-mode addressing by operating systems (AIX, Linux, IBM i), the POWER processors (POWER4 onwards) have designed virtualization capabilities where a hardware address-offset is evaluated with the OS address-offset to arrive at the physical memory address. Input/Output (I/O) adapters can be exclusively "owned" by LPARs or shared by LPARs through an appliance partition known as the Virtual I/O Server (VIOS). The Power Hypervisor provides for high levels of reliability, availability and serviceability (RAS) by facilitating hot add/replace of many parts (model dependent: processors, memory, I/O adapters, blowers, power units, disks, system controllers, etc.) It is interesting to note that because this PowerVM hypervisor is integral and part of every single POWER system IBM has made since the POWER4 systems, that every benchmark ever run on those systems is technically virtualized and as the benchmark results indicate this virtualization works extremely well. Furthermore it is extremely secure and in fact to date there has never been a single reported security flaw reported in the PowerVM hypervisor itself.

Similar trends have occurred with x86/x86_64 server platforms, where open-source projects such as Xen have led virtualization efforts. These include hypervisors built on Linux and Solaris kernels as well as custom kernels. Since these technologies span from large systems down to desktops, they are described in the next section.

5.4 x86 systems

Main article: [x86 virtualization](#)

Starting in 2005, CPU vendors have added hardware virtualization assistance to their products, for example: Intel VT-x (codenamed Vanderpool) and AMD-V (codenamed Pacifica).

An alternative approach requires modifying the guest operating-system to make system calls to the hypervisor, rather than executing machine I/O instructions that the hypervisor simulates. This is called paravirtualization in Xen, a “hypercall” in Parallels Workstation, and a “DIAGNOSE code” in IBM's VM. VMware supplements the slowest rough corners of virtualization with device drivers for the guest. All are really the same thing, a system call to the hypervisor below. Some microkernels such as Mach and L4 are flexible enough such that “paravirtualization” of guest operating systems is possible.

In June 2008, Microsoft delivered a new Type-1 hypervisor called Hyper-V (codenamed “Viridian” and previously referred to as “Windows Server virtualization”); the design features OS integration at the lowest level.*[12] Versions of Windows beginning with Windows Vista include extensions to boost performance when running on top of the Hyper-V hypervisor.

5.5 Embedded systems

Embedded hypervisors, targeting embedded systems and certain real-time operating system (RTOS) environments, are designed with different requirements when compared to desktop and enterprise systems, including robustness, security and real-time capabilities. The resource-constrained nature of many embedded systems, especially battery-powered mobile systems, imposes a further requirement for small memory-size and low overhead. Finally, in contrast to the ubiquity of the x86 architecture in the PC world, the embedded world uses a wider variety of architectures and less standardized environments. Support for virtualization requires memory protection (in the form of a memory management unit or at least a memory protection unit) and a distinction between user mode and privileged mode, which rules out most microcontrollers. This still leaves x86, MIPS, ARM and PowerPC as widely deployed architectures on medium- to high-end embedded systems.*[13]

As manufacturers of embedded systems usually have the source code to their operating systems, they have less need for full virtualization in this space. Instead, the performance advantages of paravirtualization make this usually the virtualization technology of choice. Nevertheless, ARM and MIPS have recently added full virtualization support as an IP option and has included it in their latest high-end processors and architecture versions, such as ARM Cortex-A15 MPCore and ARMv8 EL2.

Other differences between virtualization in server/desktop and embedded environments include requirements for efficient sharing of resources across virtual machines, high-bandwidth, low-latency inter-VM communication, a global view of scheduling and power management, and fine-grained control of information flows.*[14]

5.6 Security implications

The use of hypervisor technology by malware and rootkits installing themselves as a hypervisor below the operating system can make them more difficult to detect because the malware could intercept any operations of the operating system (such as someone entering a password) without the anti-malware software necessarily detecting it (since the malware runs below the entire operating system). Implementation of the concept has allegedly occurred in the SubVirt laboratory rootkit (developed jointly by Microsoft and University of Michigan researchers*[15]) as well as in the Blue Pill malware package. However, such assertions have been disputed by others who claim that it would be possible to detect the presence of a hypervisor-based rootkit.*[16]

In 2009, researchers from Microsoft and North Carolina State University demonstrated a hypervisor-layer anti-rootkit called **Hooksafe** that can provide generic protection against kernel-mode rootkits.* [17]

5.7 References

- [1] Popek, Gerald J.; Goldberg, Robert P. (1974). “Formal Requirements for Virtualizable Third Generation Architectures” . *Communications of the ACM* **17** (7): 412–421. doi:10.1145/361011.361073.
- [2] Dexter, Michael. “Hands-on bhyve” . *CallForTesting.org*. Retrieved 2013-09-24.
- [3] Graziano, Charles (2011). “A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project” . *Graduate Theses and Dissertations*. Iowa State University. Retrieved 2013-01-29.
- [4] Pariseau, Beth (15 April 2011). “KVM reignites Type 1 vs. Type 2 hypervisor debate” . *SearchServerVirtualization*. TechTarget. Retrieved 2013-01-29.
- [5] Keegan, Will (2012). “The Rise of the Type Zero Hypervisor” . *Embedded Innovator*. Intel. Retrieved 2014-07-27. The Type Zero hypervisor is a new concept that is even smaller than Type 1. Type Zero is a bare-metal architecture that removes the need for a support OS. By shedding the support OS, the Type Zero hypervisor drastically reduces the size and computational overhead imposed on virtualization platforms.
- [6] Vizard, Mike (19 May 2011). “Desktop Virtualization sans the Hypervisor” . *IT Business Edge*. Quinstreet Enterprise.
- [7] Beaver, Steve; Haletky, Edward (26 July 2012). “Type 0 Hypervisor – Fact or Fiction” . *The Virtualization Practice*.
- [8] See History of CP/CMS for virtual-hardware simulation in the development of the System/370
- [9] (virtualization quickly becoming open source 'killer app')
- [10] Wind River To Support Sun's Breakthrough UltraSPARC T1 Multithreaded Next-Generation Processor
- [11] Complementary and Alternative Technologies to Trusted Computing (TC-Erg./-A.), Part 1, A study on behalf of the German Federal Office for Information Security (BSI), Lothar Fritsch, Rani Husseiki, Ammar Alkassar
- [12] Peter Galli. “Microsoft Sheds More Light on Windows Hypervisor Technology.” April 5, 2006.
- [13] Strobl, Marius (2013). *Virtualization for Reliable Embedded Systems*. Munich: GRIN Publishing GmbH. pp. 5–6. ISBN 978-3-656-49071-5.
- [14] Gernot Heiser (April 2008). “Proc. 1st Workshop on Isolation and Integration in Embedded Systems (IIES'08)”. pp. 11–16. lchapter= ignored (help)
- [15] “SubVirt: Implementing malware with virtual machines” . University of Michigan, Microsoft. 2006-04-03. Retrieved 2008-09-15.
- [16] “Debunking Blue Pill myth” . Virtualization.info. Retrieved 2010-12-10.
- [17] Zhi Wang, Xuxian Jiang, Weidong Cui & Peng Ning (11 August 2009). “Countering Kernel Rootkits with Lightweight Hook Protection” . Microsoft/North Carolina State University. Retrieved 2009-11-11.

5.8 External links

- Virtual systems overview Description of Hypervisors at IBM's Systems Software Information Center

Chapter 6

Hardware-assisted virtualization

In computing, **hardware-assisted virtualization** is a platform virtualization approach that enables efficient full virtualization using help from hardware capabilities, primarily from the host processors. Full virtualization is used to simulate a complete hardware environment, or **virtual machine**, in which an unmodified guest operating system (using the same instruction set as the host machine) executes in complete isolation. Hardware-assisted virtualization was added to x86 processors (Intel VT-x or AMD-V) in 2006.

Hardware-assisted virtualization is also known as **accelerated virtualization**; Xen calls it **hardware virtual machine (HVM)**, and Virtual Iron calls it **native virtualization**.

6.1 History

See also: [Timeline of virtualization development](#)

Hardware-assisted virtualization first appeared on the **IBM System/370** in 1972, for use with **VM/370**, the first virtual-machine operating system. With the increasing demand for high-definition computer graphics (e.g. **CAD**), virtualization of mainframes lost some attention in the late 1970s, when the upcoming **minicomputers** fostered resource allocation through **distributed computing**, encompassing the commoditization of **microcomputers**.

IBM offer hardware virtualization for their **POWER CPUs** under **AIX** (e.g. **System p**) and for their **IBM-Mainframes System z**. IBM refers to their specific form of hardware virtualization as “**logical partition**”, or more commonly as **LPAR**.

The increase in compute capacity per x86 server and in particular the substantial increase in modern networks' bandwidths rekindled interest in data-center based computing which is based on virtualization techniques. The primary driver was the potential for server consolidation: virtualization allowed a single server to cost-efficiently consolidate compute power on multiple underutilized dedicated servers. **Cloud computing** as the new synonym for the said data center based computing (or mainframe-like computing, respectively) through high bandwidth networks is the most visible hallmark of a return to the roots of computing. It is closely connected to virtualization.

The initial implementation x86 architecture did not meet the **Popek and Goldberg** virtualization requirements to achieve “**classical virtualization**”:

- **equivalence**: a program running under the **virtual machine monitor (VMM)** should exhibit a behavior essentially identical to that demonstrated when running on an equivalent machine directly
- **resource control** (also called **safety**): the VMM must be in complete control of the virtualized resources
- **efficiency**: a statistically dominant fraction of machine instructions must be executed without VMM intervention

This made it difficult to implement a virtual machine monitor for this type of processor. Specific limitations included the inability to trap on some privileged instructions.* [1]

To compensate for these architectural limitations, designers have accomplished virtualization of the x86 architecture through two methods: full virtualization or paravirtualization.* [2] Both create the illusion of physical hardware to achieve the goal of operating system independence from the hardware but present some trade-offs in performance and complexity.

1. Paravirtualization is a technique in which the hypervisor provides an API and the OS of the guest virtual machine calls that API, requiring OS modifications.
2. Full virtualization was implemented in first-generation x86 VMMs. It relies on **binary translation** to trap and virtualize the execution of certain sensitive, non-virtualizable instructions. With this approach, critical instructions are discovered (statically or dynamically at run-time) and replaced with traps into the VMM to be emulated in software. Binary translation can incur a large performance overhead in comparison to a virtual machine running on natively virtualized architectures such as the IBM System/370. **VirtualBox**, **VMware Workstation** (for 32-bit guests only), and **Microsoft Virtual PC**, are well-known commercial implementations of full virtualization.

In 2005 and 2006, **Intel** and **AMD** (working independently) created new processor extensions to the x86 architecture called **Intel VT-x** and **AMD-V**, respectively (On the **Itanium** architecture, hardware-assisted virtualization is known as **VT-i**). The first generation of x86 processors to support these extensions were released in late 2005 early 2006:

- On November 13, 2005, Intel released two models of Pentium 4 (Model 662 and 672) as the first Intel processors to support VT-x.
- On May 23, 2006, AMD released the Athlon 64 (“Orleans”), the Athlon 64 X2 (“Windsor”) and the Athlon 64 FX (“Windsor”) as the first AMD processors to support this technology.

Well-known implementations of hardware-assisted x86 virtualization include **VMware Workstation** (for 64-bit guests only), **Xen 3.x** (including derivatives like **Virtual Iron**), **Linux KVM** and **Microsoft Hyper-V**.

6.2 Pros

Hardware-assisted virtualization reduces the maintenance overhead of paravirtualization as it reduces (ideally, eliminates) the changes needed in the guest operating system. It is also considerably easier to obtain better performance. A practical benefit of hardware-assisted virtualization has been cited by **VMware engineers*** [3] and **Virtual Iron**.

6.3 Cons

Hardware-assisted virtualization requires explicit support in the host CPU, which is not available on all x86/x86_64 processors.

A “pure” hardware-assisted virtualization approach, using entirely unmodified guest operating systems, involves many VM traps, and thus high CPU overheads, limiting scalability and the efficiency of server consolidation.* [4] This performance hit can be mitigated by the use of paravirtualized drivers; the combination has been called “hybrid virtualization” .* [5]

In 2006 first-generation 32- and 64-bit x86 hardware support was found rarely to offer performance advantages over software virtualization.* [6]

6.4 See also

- Further refinements of hardware-assisted virtualization are possible using an **IOMMU**; this allows native-speed access to dedicated hardware from a guest operating system, including **DMA-capable hardware**
- **Rapid Virtualization Indexing**
- **Extended Page Table**

- Other virtualization techniques include operating system-level virtualization, as practiced by Parallels Virtuozzo Containers, and application virtualization.
- Nanokernel
- Hardware emulation
- Emulator
- Joint Test Action Group
- Background Debug Mode interface
- In-circuit emulator

6.5 References

- [1] Adams, Keith. “A Comparison of Software and Hardware Techniques for x86 Virtualization” . Retrieved 20 January 2013.
- [2] Chris Barclay, *New approach to virtualizing x86s*, Network World, 10/20/2006
- [3] See <http://x86vmm.blogspot.com/2005/12/graphics-and-io-virtualization.html>
- [4] See <http://www.valinux.co.jp/documents/tech/presentlib/2007/2007xenconf/Intel.pdf>
- [5] Jun Nakajima and Asit K. Mallick, *Hybrid-Virtualization—Enhanced Virtualization for Linux*, in *Proceedings of the Linux Symposium*, Ottawa, June 2007, <http://ols.108.redhat.com/2007/Reprints/nakajima-Reprint.pdf>
- [6] A Comparison of Software and Hardware Techniques for x86 Virtualization, Keith Adams and Ole Agesen, VMware, ASPLOS’06 October 21–25, 2006, San Jose, California, USA “Surprisingly, we find that the first-generation hardware support rarely offers performance advantages over existing software techniques. We ascribe this situation to high VMM/guest transition costs and a rigid programming model that leaves little room for software flexibility in managing either the frequency or cost of these transitions.

6.6 Further reading

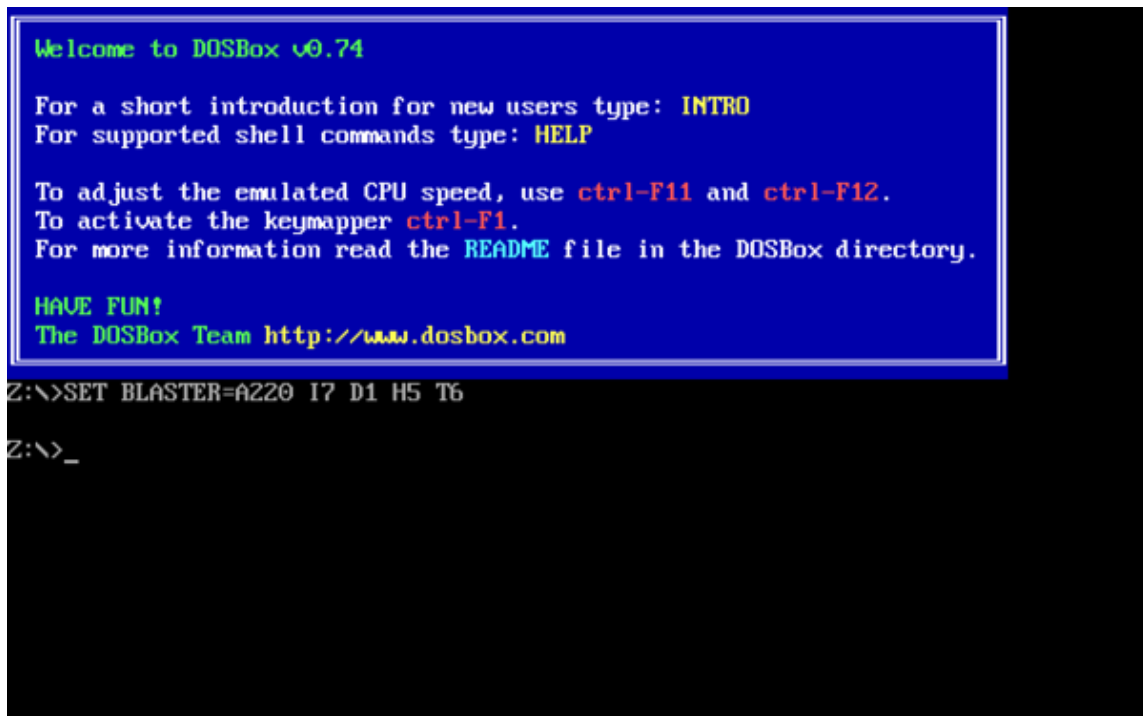
- Fisher-Ogden, John. “Hardware Support for Efficient Virtualization” (PDF). UCSD. Retrieved 2010-08-05.
- Smith, Jim; Nair, Ravi (2005). *Virtual Machines*. Morgan Kaufmann. 8.5 : Performance Enhancement of System Virtual Machines. ISBN 1-55860-910-5.
- Osisek, D. L.; Jackson, K. M.; Gum, P. H. (1991). “ESA/390 interpretive-execution architecture, foundation for VM/ESA” (PDF). *IBM Systems Journal* **30** (1).
- Adams, Keith; Agesen, Ole (2006-21-2006). “A Comparison of Software and Hardware Techniques for x86 Virtualization” (PDF). International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, 2006. ACM 1-59593-451-0/06/0010. Retrieved 2006-12-22. Check date values in: |date= (help)
- “Performance Evaluation of AMD RVI Hardware Assist” (PDF). *VMware*.
- “Performance Evaluation of Intel EPT Hardware Assist” (PDF). *VMware*.

Chapter 7

Emulator

This article is about emulators in computing. For a line of digital musical instruments, see E-mu Emulator. For the Transformers character, see Circuit Breaker (Transformers)#Shattered Glass. For other uses, see Emulation (disambiguation).

In computing, an **emulator** is hardware or software or both that duplicates (or *emulates*) the functions of one



```
Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>_
```

DOSBox emulates the command-line interface of DOS.

computer system (the *guest*) in another computer system (the *host*), different from the first one, so that the emulated behavior closely resembles the behavior of the real system (the guest).

The above described focus on exact reproduction of behavior is in contrast to some other forms of computer simulation, in which an abstract model of a system is being simulated. For example, a computer simulation of a hurricane or a chemical reaction is not emulation.

7.1 Emulators in computing

Emulation refers to the ability of a computer program in an electronic device to **emulate** (imitate) another program or device. Many printers, for example, are designed to emulate Hewlett-Packard LaserJet printers because so much

software is written for HP printers. If a non-HP printer emulates an HP printer, any software written for a real HP printer will also run in the non-HP printer emulation and produce equivalent printing.

A **hardware emulator** is an emulator which takes the form of a hardware device. Examples include the DOS-compatible card installed in some old-world Macintoshes like Centris 610 or Performa 630 that allowed them to run PC programs and FPGA-based hardware emulators.

In a theoretical sense, the **Church-Turing thesis** implies that (under the assumption that enough memory is available) any operating environment can be emulated within any other. However, in practice, it can be quite difficult, particularly when the exact behavior of the system to be emulated is not documented and has to be deduced through **reverse engineering**. It also says nothing about timing constraints; if the emulator does not perform as quickly as the original hardware, the emulated software may run much more slowly than it would have on the original hardware, possibly triggering time interrupts that alter performance.

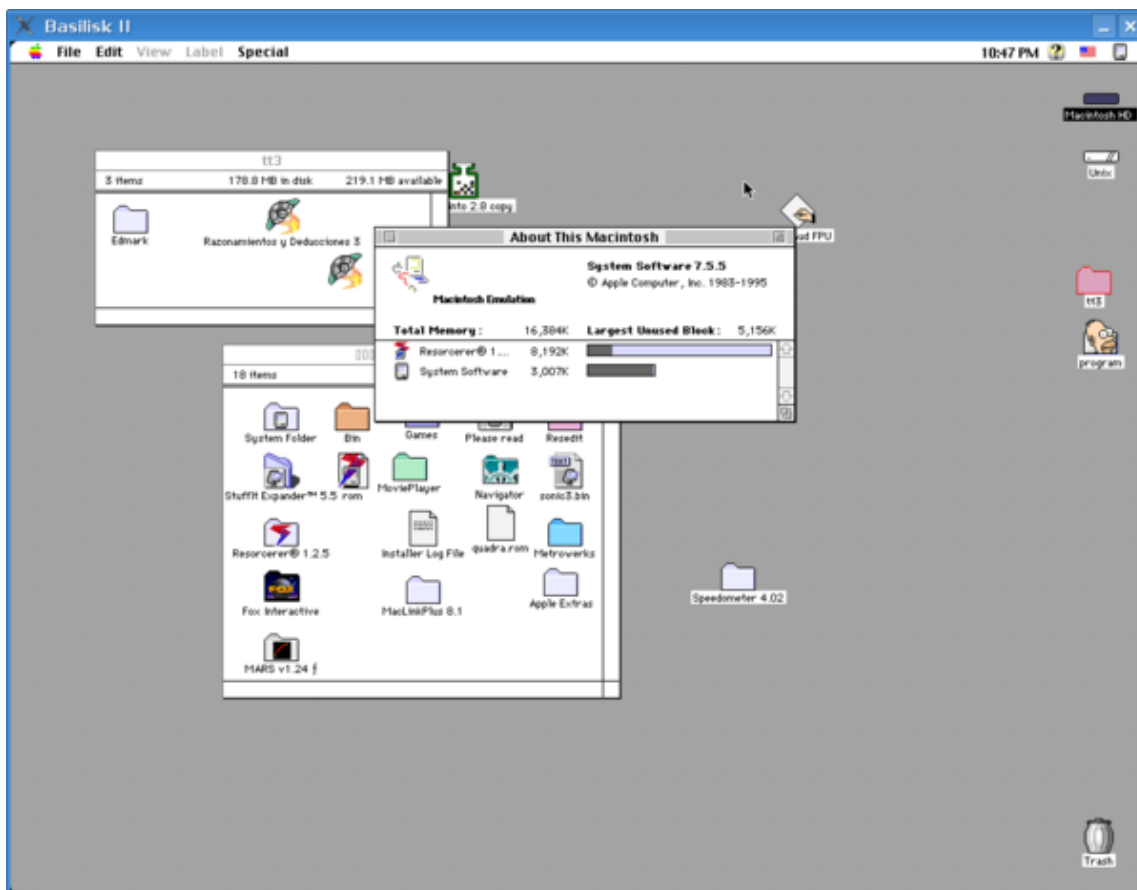
7.2 Emulation in preservation

Emulation is a strategy in **digital preservation** to combat **obsolescence**. Emulation focuses on recreating an original computer environment, which can be time-consuming and difficult to achieve, but valuable because of its ability to maintain a closer connection to the authenticity of the digital object. * [2]

Emulation addresses the original **hardware** and **software** environment of the digital object, and recreates it on a current machine. * [3] The emulator allows the user to have access to any kind of **application** or **operating system** on a current **platform**, while the **software** runs as it did in its original environment. * [4] Jeffery Rothenberg, an early proponent of emulation as a **digital preservation** strategy states, “the ideal approach would provide a single extensible, long-term solution that can be designed once and for all and applied uniformly, automatically, and in synchrony (for example, at every refresh cycle) to all types of documents and media” . * [5] He further states that this should not only apply to out of date systems, but also be upwardly mobile to future unknown systems. * [6] Practically speaking, when a certain application is released in a new version, rather than address **compatibility** issues and **migration** for every digital object created in the previous version of that **application**, one could create an emulator for the **application**, allowing access to all of said digital objects.

7.2.1 Benefits

- Potentially better graphics quality than original hardware.
- Potentially additional features original hardware didn't have.
- Save states
- Emulators allow users to play games for discontinued consoles.
- Emulators maintain the original look, feel, and behavior of the digital object, which is just as important as the digital data itself. * [7]
- Despite the original cost of developing an emulator, it may prove to be the more cost efficient solution over time. * [8]
- Reduces **labor** hours, because rather than continuing an ongoing task of continual **data migration** for every digital object, once the library of past and present **operating systems** and **application software** is established in an emulator, these same technologies are used for every document using those **platforms**. * [4]
- Many emulators have already been developed and released under **GNU General Public License** through the **open source** environment, allowing for wide scale collaboration. * [9]
- Emulators allow software exclusive to one system to be used on another. For example, a **PlayStation 2** exclusive video game could be played on a **PC** using an emulator. This is especially useful when the original system is difficult to obtain, or incompatible with modern equipment (e.g. old video game consoles which connect via analog outputs may be unable to connect to modern TVs which may only have digital inputs).



Basilisk II emulates a Macintosh 68k using interpretation code and dynamic recompilation.

7.2.2 Obstacles

- **Intellectual property** - Many technology vendors implemented non-standard features during program development in order to establish their niche in the market, while simultaneously applying ongoing upgrades to remain competitive. While this may have advanced the technology industry and increased vendor's market share, it has left users lost in a preservation nightmare with little supporting documentation due to the proprietary nature of the hardware and software.*[10]
- **Copyright laws** are not yet in effect to address saving the documentation and specifications of proprietary software and hardware in an emulator module.*[11]
- Emulators are often used as a **copyright infringement** tool, since they allow users to play video games without having to buy the console, and rarely make any attempt to prevent the use of illegal copies. This leads to a number of legal uncertainties regarding emulation, and leads to software being programmed to refuse to work if it can tell the host is an emulator; some video games in particular will continue to run, but not allow the player to progress beyond some late stage in the game, often appearing to be faulty or just extremely difficult.*[12]*[13] These protections make it more difficult to design emulators, since they must be accurate enough to avoid triggering the protections, whose effects may not be obvious.
- Emulators require better hardware than the original system has.

7.3 Emulators in new media art

Because of its primary use of digital formats, new media art relies heavily on emulation as a preservation strategy. Artists such as Cory Arcangel specialize in resurrecting obsolete technologies in their artwork and recognize the importance of a decentralized and deinstitutionalized process for the preservation of digital culture.

In many cases, the goal of emulation in new media art is to preserve a digital medium so that it can be saved indefinitely and reproduced without error, so that there is no reliance on hardware that ages and becomes obsolete. The paradox is that the emulation and the emulator have to be made to work on future computers.*[14]

7.4 Emulation in future systems design

Main article: [Full system simulation](#)

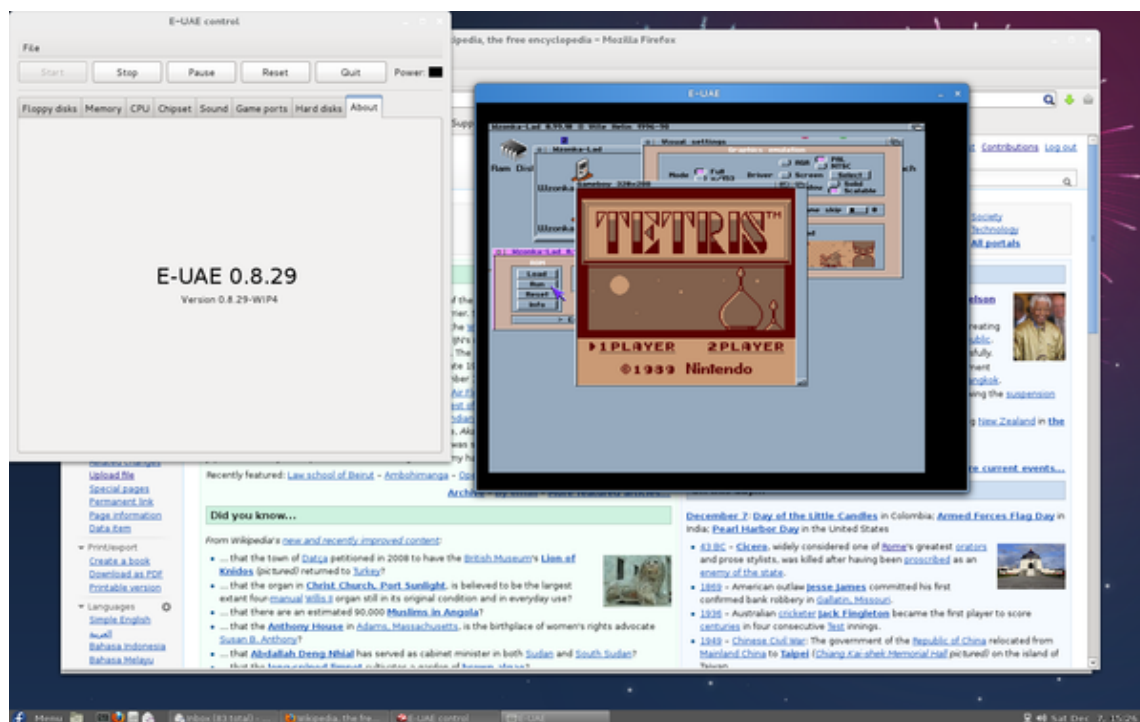
Emulation techniques are commonly used during the design and development of new systems. It eases the development process by providing the ability to detect, recreate and repair flaws in the design even before the system is actually built.*[15] It is particularly useful in the design of multi-cores systems, where concurrency errors can be very difficult to detect and correct without the controlled environment provided by virtual hardware.*[16] This also allows the software development to take place before the hardware is ready,*[17] thus helping to validate design decisions.

7.5 Types of emulators



Windows XP running an Acorn Archimedes emulator, which is in turn running a Sinclair ZX Spectrum emulator.

Most emulators just emulate a hardware architecture—if operating system firmware or software is required for the desired software, it must be provided as well (and may itself be emulated). Both the OS and the software will then be interpreted by the emulator, rather than being run by native hardware. Apart from this interpreter for the emulated binary machine's language, some other hardware (such as input or output devices) must be provided in virtual form



Tetris running on the Wzonga-Lad GameBoy emulator on AmigaOS, itself running on E-UAE on a modern Fedora Linux system.

as well; for example, if writing to a specific memory location should influence what is displayed on the screen, then this would need to be emulated.

While emulation could, if taken to the extreme, go down to the atomic level, basing its output on a simulation of the actual circuitry from a virtual power source, this would be a highly unusual solution. Emulators typically stop at a simulation of the documented hardware specifications and digital logic. Sufficient emulation of some hardware platforms requires extreme accuracy, down to the level of individual clock cycles, undocumented features, unpredictable analog elements, and implementation bugs. This is particularly the case with classic home computers such as the Commodore 64, whose software often depends on highly sophisticated low-level programming tricks invented by game programmers and the demoscene.

In contrast, some other platforms have had very little use of direct hardware addressing. In these cases, a simple compatibility layer may suffice. This translates system calls for the emulated system into system calls for the host system e.g., the Linux compatibility layer used on *BSD to run closed source Linux native software on FreeBSD, NetBSD and OpenBSD. For example, while the Nintendo 64 graphic processor was fully programmable, most games used one of a few pre-made programs, which were mostly self-contained and communicated with the game via FIFO; therefore, many emulators do not emulate the graphic processor at all, but simply interpret the commands received from the CPU as the original program would.

Developers of software for embedded systems or video game consoles often design their software on especially accurate emulators called simulators before trying it on the real hardware. This is so that software can be produced and tested before the final hardware exists in large quantities, so that it can be tested without taking the time to copy the program to be debugged at a low level and without introducing the side effects of a debugger. In many cases, the simulator is actually produced by the company providing the hardware, which theoretically increases its accuracy.

Math coprocessor emulators allow programs compiled with math instructions to run on machines that don't have the coprocessor installed, but the extra work done by the CPU may slow the system down. If a math coprocessor isn't installed or present on the CPU, when the CPU executes any coprocessor instruction it will make a determined interrupt (coprocessor not available), calling the math emulator routines. When the instruction is successfully emulated, the program continues executing.

7.6 Structure of an emulator

Typically, an emulator is divided into **modules** that correspond roughly to the emulated computer's subsystems. Most often, an emulator will be composed of the following modules:

- a CPU emulator or CPU simulator (the two terms are mostly interchangeable in this case), unless the target being emulated has the same CPU architecture as the host, in which case a **virtual machine** layer may be used instead
- a memory subsystem module
- various I/O devices emulators

Buses are often not emulated, either for reasons of performance or simplicity, and virtual peripherals communicate directly with the CPU or the memory subsystem.

7.6.1 Memory subsystem

It is possible for the memory subsystem emulation to be reduced to simply an array of elements each sized like an emulated word; however, this model falls very quickly as soon as any location in the computer's logical memory does not match **physical memory**.

This clearly is the case whenever the emulated hardware allows for advanced memory management (in which case, the **MMU** logic can be embedded in the memory emulator, made a module of its own, or sometimes integrated into the CPU simulator).

Even if the emulated computer does not feature an MMU, though, there are usually other factors that break the equivalence between logical and physical memory: many (if not most) architectures offer **memory-mapped I/O**; even those that do not often have a block of logical memory mapped to **ROM**, which means that the memory-array module must be discarded if the read-only nature of ROM is to be emulated. Features such as **bank switching** or **segmentation** may also complicate memory emulation.

As a result, most emulators implement at least two procedures for writing to and reading from logical memory, and it is these procedures' duty to map every access to the correct location of the correct object.

On a **base-limit addressing** system where memory from address 0 to address *ROMSIZE-1* is read-only memory, while the rest is RAM, something along the line of the following procedures would be typical:

```
void WriteMemory(word Address, word Value) { word RealAddress; RealAddress = Address + BaseRegister; if
((RealAddress < LimitRegister) && (RealAddress > ROMSIZE)) { Memory[RealAddress] = Value; } else { Rai-
seInterrupt(INT_SEGFAULT); } }
word ReadMemory(word Address) { word RealAddress; RealAddress=Address+BaseRegister; if (RealAddress <
LimitRegister) { return Memory[RealAddress]; } else { RaiseInterrupt(INT_SEGFAULT); return NULL; } }
```

7.6.2 CPU simulator

The CPU simulator is often the most complicated part of an emulator. Many emulators are written using “pre-packaged” CPU simulators, in order to concentrate on good and efficient emulation of a specific machine.

The simplest form of a CPU simulator is an **interpreter**, which is a computer program that follows the execution flow of the emulated program code and, for every machine code instruction encountered, executes operations on the host processor that are semantically equivalent to the original instructions.

This is made possible by assigning a **variable** to each **register** and **flag** of the simulated CPU. The logic of the simulated CPU can then more or less be directly translated into software algorithms, creating a software re-implementation that basically mirrors the original hardware implementation.

The following example illustrates how CPU simulation can be accomplished by an interpreter. In this case, interrupts are checked-for before every instruction executed, though this behavior is rare in real emulators for performance reasons (it is generally faster to use a subroutine to do the work of an interrupt).

```
void Execute(void) { if (Interrupt != INT_NONE) { SuperUser = TRUE; WriteMemory(++StackPointer, ProgramCounter); ProgramCounter = InterruptPointer; } switch (ReadMemory(ProgramCounter++)) { /* * Handling of every valid instruction * goes here... */ default: Interrupt = INT_ILLEGAL; } }
```

Interpreters are very popular as computer simulators, as they are much simpler to implement than more time-efficient alternative solutions, and their speed is more than adequate for emulating computers of more than roughly a decade ago on modern machines.

However, the speed penalty inherent in interpretation can be a problem when emulating computers whose processor speed is on the same *order of magnitude* as the host machine. Until not many years ago, emulation in such situations was considered completely impractical by many.

What allowed breaking through this restriction were the advances in *dynamic recompilation* techniques. Simple *a priori* translation of emulated program code into code runnable on the host architecture is usually impossible because of several reasons:

- code may be *modified while in RAM*, even if it is modified only by the emulated operating system when loading the code (for example from disk)
- there may not be a way to reliably distinguish *data* (which should not be translated) from *executable code*.

Various forms of dynamic recompilation, including the popular *Just In Time compiler (JIT)* technique, try to circumvent these problems by waiting until the processor control flow jumps into a location containing untranslated code, and only then (“just in time”) translates a block of the code into host code that can be executed. The translated code is kept in a *code cache*, and the original code is not lost or affected; this way, even data segments can be (meaninglessly) translated by the recompiler, resulting in no more than a waste of translation time.

Speed may not be desirable as some older games were not designed with the speed of faster computers in mind. A game designed for a 30 MHz PC with a level timer of 300 game seconds might only give the player 30 seconds on a 300 MHz PC. Other programs, such as some DOS programs, may not even run on faster computers. Particularly when emulating computers which were “closed-box”, in which changes to the core of the system were not typical, software may use techniques that depend on specific characteristics of the computer it ran on (i.e. its CPU's speed) and thus precise control of the speed of emulation is important for such applications to be properly emulated.

7.6.3 I/O

Most emulators do not, as mentioned earlier, emulate the main system bus; each I/O device is thus often treated as a special case, and no consistent interface for virtual peripherals is provided.

This can result in a performance advantage, since each I/O module can be tailored to the characteristics of the emulated device; designs based on a standard, unified I/O API can, however, rival such simpler models, if well thought-out, and they have the additional advantage of “automatically” providing a plug-in service through which third-party virtual devices can be used within the emulator.

A unified I/O API may not necessarily mirror the structure of the real hardware bus: bus design is limited by several electric constraints and a need for hardware *concurrency* management that can mostly be ignored in a software implementation.

Even in emulators that treat each device as a special case, there is usually a common basic infrastructure for:

- managing interrupts, by means of a procedure that sets flags readable by the CPU simulator whenever an interrupt is raised, allowing the virtual CPU to “poll for (virtual) interrupts”
- writing to and reading from physical memory, by means of two procedures similar to the ones dealing with logical memory (although, contrary to the latter, the former *can* often be left out, and direct references to the memory array be employed instead)

7.7 Emulation versus simulation

The word “emulator” was coined in 1963 at IBM*[18] during development of the NPL (IBM 360) product line, using a “new combination of software, microcode, and hardware”.*[19] They discovered that using microcode hardware instead of software simulation, to execute programs written for earlier IBM computers, dramatically increased simulation speed. Earlier, IBM provided simulators for, e.g., the 650 on the 705.*[20]

In addition to simulators, IBM had compatibility features on the 709 and 7090,*[21] for which it provided the IBM 709 computer with a program to run legacy programs written for the IBM 704 on the 709 and later on the IBM 7090. This program used the instructions added by the compatibility feature*[22] to trap instructions requiring special handling; all other 704 instructions ran the same on a 7090. The compatibility feature on the 1410*[23] only required setting a console toggle switch, not a support program.

In 1963, when microcode was first used to speed up this simulation process, IBM engineers coined the term “emulator” to describe the concept.

It has recently become common to use the word “emulate” in the context of software. However, before 1980, “emulation” referred only to emulation with a hardware or microcode assist, while “simulation” referred to pure software emulation.*[24] For example, a computer specially built for running programs designed for another architecture is an emulator. In contrast, a simulator could be a program which runs on a PC, so that old Atari games can be simulated on it. Purists continue to insist on this distinction, but currently the term “emulation” often means the complete imitation of a machine executing binary code while “simulation” often refers to computer simulation, where a computer program is used to simulate an abstract model. Computer simulation is used in virtually every scientific and engineering domain and Computer Science is no exception, with several projects simulating abstract models of computer systems, such as network simulation.

7.8 Logic simulators

Main article: Logic simulation

Logic simulation is the use of a computer program to simulate the operation of a digital circuit such as a processor. This is done after a digital circuit has been designed in logic equations, but before the circuit is fabricated in hardware.

7.9 Functional simulators

Main article: High level emulation

Functional simulation is the use of a computer program to simulate the execution of a second computer program written in symbolic assembly language or compiler language, rather than in binary machine code. By using a functional simulator, programmers can execute and trace selected sections of source code to search for programming errors (bugs), without generating binary code. This is distinct from simulating execution of binary code, which is software emulation.

The first functional simulator was written by Autonetics about 1960 for testing assembly language programs for later execution in military computer D-17B. This made it possible for flight programs to be written, executed, and tested before D-17B computer hardware had been built. Autonetics also programmed a functional simulator for testing flight programs for later execution in the military computer D-37C.

7.10 Video game console emulators

Main article: Video game console emulator

Video game console emulators are programs that allow a personal computer or video game console to emulate another video game console. They are most often used to play older video games on personal computers and more contemporary video game consoles, but they are also used to translate games into other languages, to modify existing games,

and in the development process of home brew demos and new games for older systems. The internet has helped in the spread of console emulators, as most - if not all - would be unavailable for sale in retail outlets. Examples of console emulators that have been released in the last 2 decades are: Dolphin, PCSX2, PPSSPP, Zsnes, Kega Fusion, Desmume, Epsxe, Project64, Visual Boy Advance, NullDC and Nestopia.

7.11 Terminal emulators

Main article: [Terminal emulator](#)

Terminal emulators are software programs that provide modern computers and devices interactive access to applications running on mainframe computer operating systems or other host systems such as HP-UX or OpenVMS. Terminals such as the IBM 3270 or VT100 and many others, are no longer produced as physical devices. Instead, software running on modern operating systems simulates a “dumb” terminal and is able to render the graphical and text elements of the host application, send keystrokes and process commands using the appropriate terminal protocol. Some terminal emulation applications include Attachmate Reflection, IBM Personal Communications, Stromasys CHARON-VAX/AXP and Micro Focus Rumba.

7.12 In literature

Vernor Vinge's 1999 novel *A Deepness in the Sky* depicts a human interstellar culture that relies on software written over five thousand years, some from “before Humankind ever left Earth” . Its computers still “can run most of them” through “a million million circuitous threads of inheritance ... layers upon layers of support” . “Down at the very bottom” of the culture’s “incredibly complex” timekeeping systems, for example, “was a little program that ran a counter [from] the 0-second of one of Humankind's first computer operating systems”. Programmer-archaeologists find and modify old software written hundreds of years ago (one character states that “all of them are buggy”) to solve modern problems.*[25]

7.13 Legal controversy

See article Console emulator —Legal issues

7.14 See also

- The list of emulators
- The list of video game emulators
- The list of computer system emulators
- Computer simulation is the larger field of modeling real-world phenomenon (e.g. physics and economy) using computers.
- Other uses of the term “emulator” in the field of computer science:
 - Console emulator
 - Flash emulator
 - Instruction set simulator
 - Network emulation
 - Server emulator
 - Terminal emulator
- Semulation

- Logic simulation
- Functional simulation
- Translation:
 - Binary translation
- In-circuit emulator (ICE)
 - Joint Test Action Group
 - Background Debug Mode interface
- QEMU
- Q (emulator)
- Hardware emulation
- Hardware-assisted virtualization
- Virtual machine

7.15 Notes

7.16 References

- [1] Warick, Mike (April 1988). "MS-DOS Emulation For The 64" . *Compute!*. p. 43. Retrieved 10 November 2013.
- [2] "What is emulation?". *Koninklijke Bibliotheek*. Retrieved 2007-12-11.
- [3] van der Hoeven, Jeffrey, Bram Lohman, and Remco Verdegem. "Emulation for Digital Preservation in Practice: The Results." *The International Journal of Digital Curation* 2.2 (2007): 123-132.
- [4] Muira, Gregory. "Pushing the Boundaries of Traditional Heritage Policy: maintaining long-term access to multimedia content." *IFLA Journal* 33 (2007): 323-326.
- [5] Rothenberg, Jeffrey (1998). "Criteria for an Ideal Solution." *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation.* . *Council on Library and Information Resources*. Washington, DC. Retrieved 2008-03-08.
- [6] Rothenberg, Jeffrey. "The Emulation Solution." *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation.* Washington, DC: Council on Library and Information Resources, 1998. Council on Library and Information Resources. 2008. 28 Mar. 2008 <http://www.clir.org/pubs/reports/rothenberg/contents.html>
- [7] Muira, Gregory." Pushing the Boundaries of Traditional Heritage Policy: maintaining long-term access to multimedia content." *IFLA Journal* 33 (2007): 323-326.
- [8] Granger, Stewart. *Digital Preservation & Emulation: from theory to practice.* Proc. of the ichim01 Meeting, vol. 2, 3 –7 Sept. 2001. Milano, Italy. Toronto: Archives and Museum Informatics, University of Toronto, 2001. 28 Mar. 2008 <http://www.leeds.ac.uk/cedars/pubconf/papers/ichim01SG.html>
- [9] van der Hoeven, Jeffrey, Bram Lohman, and Remco Verdegem. "Emulation for Digital Preservation in Practice: The Results." *The International Journal of Digital Curation* 2.2 (2007): 123-132.
- [10] Granger, Stewart. "Emulation as a Digital Preservation Strategy." *D-Lib Magazine* 6.19 (2000). 29 Mar 2008 <http://www.dlib.org/dlib/october00/granger/10granger.html>
- [11] Rothenberg, Jeffrey. "The Emulation Solution." *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation.* Washington, DC: Council on Library and Information Resources, 1998. Council on Library and Information Resources. 2008. 28 Mar. 2008
- [12] http://tcrf.net/Pok%C3%A9mon_Black_and_White#Anti-Piracy
- [13] http://tcrf.net/Mega_Man_Star_Force#Anti-Piracy

- [14] “Echoes of Art: Emulation as preservation strategy” . Retrieved 2007-12-11.
- [15] Peter Magnusson (2004). “Full System Simulation: Software Development's Missing Link” .
- [16] “Debugging and Full System Simulation” .
- [17] Vania Joloboff (2009). “Full System Simulation of Embedded Systems” .
- [18] Pugh, Emerson W. (1995). *Building IBM: Shaping an Industry and Its Technology*. MIT. p. 274. ISBN 0-262-16147-8.
- [19] Pugh, Emerson W.; et al. (1991). *IBM's 360 and Early 370 Systems*. MIT. ISBN 0-262-16123-0. pages 160-161
- [20] Simulation of the IBM 650 on the IBM 705
- [21] Data Processing System - Compatibility feature for IBM 704 programs
- [22] “System Compatibility Operations” . *Reference Manual IBM 7090 Data Processing System*. March 1962. pp. 65–66. A22-6528-4.
- [23] “System Compatibility Operations” . *IBM 1410 Principles of Operation*. March 1962. pp. 56–57, 98–100. A22-0526-3.
- [24] S. G. Tucker, “Emulation of Large Systems” , *Communications of the ACM (CACM)* Vol. 8, No. 12, Dec. 1965, pp. 753-761
- [25] Vinge, Vernor (1999). *A Deepness in the Sky*. Tor. pp. 224–227. ISBN 9780812536355.

7.17 External links

- (archived copy) is a repertory of emulators and their respective histories (site closed in 2010 due to copyright issues).
- Emulator at DMOZ

Chapter 8

Snapshot (computer storage)

In computer systems, a **snapshot** is the *state* of a system at a particular point in time. The term was coined as an analogy to that in photography. It can refer to an *actual copy* of the state of a system or to a capability provided by certain systems.

8.1 Rationale

A full backup of a large data set may take a long time to complete. On multi-tasking or multi-user systems, there may be writes to that data while it is being backed up. This prevents the backup from being atomic and introduces a *version skew* that may result in data corruption. For example, if a user moves a file into a directory that has already been backed up, then that file would be completely missing on the backup media, since the backup operation had already taken place before the addition of the file. Version skew may also cause corruption with files which change their size or contents underfoot while being read.

One *approach* to safely backing up live data is to temporarily disable write access to data during the backup, either by stopping the accessing applications or by using the *locking API* provided by the operating system to enforce exclusive read access. This is tolerable for low-availability systems (on desktop computers and small workgroup servers, on which regular *downtime* is acceptable). High-availability 24/7 systems, however, cannot bear service stoppages.

To avoid downtime, high-availability systems may instead perform the backup on a *snapshot*—a *read-only* copy of the data set frozen at a *point in time*—and allow applications to continue writing to their data. Most snapshot implementations are efficient and can create snapshots in $O(1)$. In other words, the time and I/O needed to create the snapshot does not increase with the size of the data set; by contrast, the time and I/O required for a direct backup is proportional to the size of the data set. In some systems once the initial snapshot is taken of a data set, subsequent snapshots copy the changed data only, and use a system of pointers to reference the initial snapshot. This method of pointer-based snapshots consumes less disk capacity than if the data set was repeatedly cloned.

Read-write snapshots are sometimes called *branching snapshots*, because they implicitly create diverging versions of their data. Aside from backups and data recovery, read-write snapshots are frequently used in *virtualization*, *sandboxing* and *virtual hosting* setups because of their usefulness in managing changes to large sets of files.

8.2 Implementations

8.2.1 Volume managers

Some Unix systems have snapshot-capable *logical volume managers*. These implement *copy-on-write* on entire *block devices* by copying changed blocks—just before they are to be overwritten within “parent” volumes—to other storage, thus preserving a self-consistent past image of the block device. Filesystems on such snapshot images can later be mounted as if they were on a read-only media.

Some volume managers also allow creation of *writable* snapshots, extending the copy-on-write approach by disassociating any blocks modified within the snapshot from their “parent” blocks in the original volume. Such a scheme

could be also described as performing additional copy-on-write operations triggered by the writes to snapshots.

On Linux, Logical Volume Manager (LVM) allows creation of both read-only and read-write snapshots. Writable snapshots were introduced with the LVM version 2 (LVM2).*[1]

8.2.2 File systems

Some file systems, such as WAFL,* [note 1] fossil for Plan 9 from Bell Labs, and ODS-5, internally track old versions of files and make snapshots available through a special namespace. Others, like UFS2, provide an operating system API for accessing file histories. In NTFS, access to snapshots is provided by the Volume Shadow-copying Service (VSS) in Windows XP and Windows Server 2003 and Shadow Copy in Windows Vista. Melio FS provides snapshots via the same VSS interface for shared storage.*[2] Snapshots have also been available in the NSS (Novell Storage Services) file system on NetWare since version 4.11, and more recently on Linux platforms in the Open Enterprise Server product.

EMC's Isilon OneFS clustered storage platform implements a single scalable file system that supports read-only snapshots at the file or directory level. Any file or directory within the file system can be snapshotted and the system will implement a copy-on-write or point-in-time snapshot dynamically based on which method is determined to be optimal for the system.

On Linux, the Btrfs and OCFS2 file systems support creating snapshots (cloning) of individual files. Additionally, Btrfs also supports the creation of snapshots of subvolumes. On AIX, JFS2 also support snapshots.

Sun Microsystems ZFS has a hybrid implementation which tracks read-write snapshots at the block level, but makes branched file sets nameable to user applications as “clones” .

Time Machine, included in Apple's Mac OS X v10.5 operating system, is not a snapshotting scheme but a system-level incremental backup service: it merely watches mounted volumes for changes and copies changed files periodically to a specially-designated volume using hard links.

8.2.3 In databases

The SQL specification mandates four levels of transaction isolation. In the highest, *SERIALIZABLE*, a snapshot is implicitly created at the start of every transaction. The backup utilities for many popular SQL databases use this feature to generate self-consistent dumps of table data.

A database snapshot provides a read-only, static view of a source database as it existed at snapshot creation, minus any uncommitted transactions. Uncommitted transactions are rolled back in a newly created database snapshot because the Database Engine runs recovery after the snapshot has been created (transactions in the database are not affected).

Database snapshots are dependent on the source database. The snapshots of a database must be on the same server instance as the database. Furthermore, if that database becomes unavailable for any reason, all of its database snapshots also become unavailable.

Snapshots can be used for reporting purposes. Also, in the event of a user error on a source database, you can revert the source database to the state it was in when the snapshot was created. Data loss is confined to updates to the database since the snapshot's creation. Also, creating a database snapshot can be useful immediately before making a major change to a database, such as changing the schema or the structure of a table. For more information on the uses of snapshots, see Typical Uses of Database Snapshots.

Understanding how snapshots work is helpful though not essential to using them. Database snapshots operate at the data-page level. Before a page of the source database is modified for the first time, the original page is copied from the source database to the snapshot. This process is called a copy-on-write operation. The snapshot stores the original page, preserving the data records as they existed when the snapshot was created. Subsequent updates to records in a modified page do not affect the contents of the snapshot. The same process is repeated for every page that is being modified for the first time. In this way, the snapshot preserves the original pages for all data records that have ever been modified since the snapshot was taken.

To store the copied original pages, the snapshot uses one or more sparse files. Initially, a sparse file is an essentially empty file that contains no user data and has not yet been allocated disk space for user data. As more and more pages are updated in the source database, the size of the file grows. When a snapshot is taken, the sparse file takes up little disk space. As the database is updated over time, however, a sparse file can grow into a very large file.

8.2.4 In virtualization

Virtualization environments host a guest operating system inside a virtual machine; some of them (including [VMware](#), [XenServer](#), [VirtualBox](#), [Parallels Desktop](#), [QEMU](#) and [Virtual PC](#)) can perform whole-system snapshots by dumping the entire machine state to a backing file, and redirecting future guest writes to a second file, which then acts as a copy-on-write table.

8.2.5 Other applications

Software transactional memory is a scheme which applies the same concepts to data structures held only in memory.

8.3 See also

- [Application checkpointing](#)
- [System image](#)
- [LVM snapshots \(Linux\)](#)
- [Persistence \(computer science\)](#)
- [R1Soft Hot Copy \(Linux\)](#)
- [Microsoft Volume Shadow Copy](#)
- [Storage Hypervisor](#)

8.4 Notes

- [1] [WAFL](#) is not a file system. [WAFL](#) is a file layout that provides mechanisms that enable a variety of file systems and technologies that want to access disk blocks.

8.5 References

- [1] [“LVM HOWTO”](#) . 3.8. *Snapshots*. [tldp.org](#). Retrieved 2013-09-29.
- [2] [“Optimized Storage Solution for Enterprise Scale Hyper-V Deployments”](#) . Microsoft. March 2010. p. 15. Retrieved 25 October 2012.

8.6 External links

- [Garimella, Neeta \(2006-04-26\). “Understanding and exploiting snapshot technology for data protection, Part 1: Snapshot technology overview”](#) .
- [Harwood, Mike \(2003-09-24\). “Storage Basics: Backup Strategies”](#) .
- [Project web page of rsnapshot](#)

Chapter 9

Migration (virtualization)

In the context of *virtualization*, where a *guest* simulation of an entire computer is actually merely a software *virtual machine* (VM) running on a *host* computer under a *hypervisor*, **migration** (also known as **teleportation**^[1]) is the process by which a *running* virtual machine is moved from one physical host to another, with little or no disruption in service.

9.1 Subjective effects

Ideally, the process is completely transparent, resulting in no disruption of service (or *downtime*). In practice, there is always some minor pause in availability, though it may be low enough that only *hard real-time* systems are affected. Virtualization is far more frequently used with network services and user applications, and these can generally tolerate the brief delays which may be involved. The perceived impact, if any, is similar to a longer-than-usual *kernel* delay.

9.2 Objective effects

The actual process is heavily dependent on the particular virtualization package in use, but in general, the process is as follows:

1. Regular *snapshots* of the VM (its simulated *hard disk* storage, its *memory*, and its *virtual peripherals*) are taken in the background by the hypervisor, or by a set of administrative *scripts*.
2. Each new snapshot adds a differential overlay file to the top of a stack that, as a whole, fully describes the machine. Only the topmost overlay can be written to.
3. Since the older overlays are read-only, they are safe to copy to another machine—the backup host. This is done at regular intervals, and each overlay need only be copied once.
4. When a migration operation is requested, the virtual machine is paused, and its current state is saved to disk.
5. These new, final overlay files are transferred to the backup host.
6. Since this new current state consists only of changes made since the last backup synchronization, for many applications there is very little to transfer, and this happens very quickly.
7. The hypervisor on the new host resumes the guest virtual machine.
8. Ideally, the administrative scripts resume backup operations, the new host becomes the primary, and the previous host now receives the backup copies, readying it for a subsequent migration operation.

Note that in practice, regular maintenance operations are required to “merge down” the snapshot stack into a more manageable number of files, to improve performance and optimize storage (redundant overwrites are merged out).

9.3 Relation to failover

Migration is similar to the failover capability some virtualization suites provide. In true failover, the host may have suddenly completely failed, which precludes the latest state of the VM having been copied to the backup host. However, the backup host has everything except for the very latest changes, and may indeed be able to resume operation from its last known coherent state.

Because the operations are so similar, systems that provide one capability may provide the other.

9.4 References

- [1] “Sun Brings 'Teleportation' to VirtualBox Virtualization Software” . eWeek.com. 30 Nov 2009. Retrieved 24 Apr 2012.

Chapter 10

Operating-system-level virtualization

Operating-system-level virtualization is a server virtualization method where the kernel of an operating system allows for multiple isolated user space instances, instead of just one. Such instances (often called **containers**, virtualization engines (VE), virtual private servers (VPS), or jails) may look and feel like a real server from the point of view of its owners and users.

On Unix-like operating systems, this technology can be seen as an advanced implementation of the standard chroot mechanism. In addition to isolation mechanisms, the kernel often provides resource management features to limit the impact of one container's activities on the other containers.

10.1 Uses

Operating-system-level virtualization is commonly used in virtual hosting environments, where it is useful for securely allocating finite hardware resources amongst a large number of mutually-distrusting users. System administrators may also use it, to a lesser extent, for consolidating server hardware by moving services on separate hosts into containers on the one server.

Other typical scenarios include separating several applications to separate containers for improved security, hardware independence, and added resource management features. The improved security provided by the use of a chroot mechanism, however, is nowhere near ironclad.* [1] Operating-system-level virtualization implementations capable of live migration can also be used for dynamic load balancing of containers between nodes in a cluster.

10.1.1 Overhead

Operating-system-level virtualization usually imposes little to no overhead, because programs in virtual partitions use the operating system's normal system call interface and do not need to be subjected to emulation or be run in an intermediate virtual machine, as is the case with whole-system virtualizers (such as VMware ESXi, QEMU or Hyper-V) and paravirtualizers (such as Xen or UML). This form of virtualization also does not require support in hardware to perform efficiently.

10.1.2 Flexibility

Operating-system-level virtualization is not as flexible as other virtualization approaches since it cannot host a guest operating system different from the host one, or a different guest kernel. For example, with Linux, different distributions are fine, but other operating systems such as Windows cannot be hosted.

Solaris partially overcomes the above described limitation with its branded zones feature, which provides the ability to run an environment within a container that emulates an older Solaris 8 or 9 version in a Solaris 10 host. Linux branded zones (referred to as “lx” branded zones) are also available on x86-based Solaris systems, providing a complete Linux userspace and support for the execution of Linux applications; additionally, Solaris provides utilities needed to install Red Hat Enterprise Linux 3.x or CentOS 3.x Linux distributions inside “lx” zones.* [2]* [3] However,

in 2010 Linux branded zones were removed from Solaris; in 2014 they were reintroduced in Illumos, which is the open source Solaris fork, supporting 32-bit Linux kernels.*[4]

10.1.3 Storage

Some operating-system-level virtualization implementations provide file-level copy-on-write (CoW) mechanisms. (Most commonly, a standard file system is shared between partitions, and those partitions that change the files automatically create their own copies.) This is easier to back up, more space-efficient and simpler to cache than the block-level copy-on-write schemes common on whole-system virtualizers. Whole-system virtualizers, however, can work with non-native file systems and create and roll back snapshots of the entire system state.

10.2 Implementations

10.3 See also

- Application virtualization
- CoreOS
- Hypervisor
- Portable application creators
- Platform virtualization
- Storage Hypervisor

10.4 References

- [1] “How to break out of a chroot() jail” . 2002. Retrieved 7 May 2013.
- [2] “System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones” . Oracle Corporation. 2010. Retrieved 2014-09-02. lchapter= ignored (help)
- [3] “System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones” . Oracle Corporation. 2010. Retrieved 2014-09-02. lchapter= ignored (help)
- [4] Bryan Cantrill (2014-09-28). “The dream is alive! Running Linux containers on an illumos kernel” . *slideshare.net*. Retrieved 2014-10-10.
- [5] Root user can easily escape from chroot. Chroot was never supposed to be used as a security mechanism.
- [6] “Docker drops LXC as default execution environment” . *InfoQ*.
- [7] Utilizing the CFQ scheduler, you get a separate queue per guest.
- [8] Networking is based on isolation, not virtualization.
- [9] 14 user capabilities are considered safe within a container. The rest may cannot be granted to processes within that container without allowing that process to potentially interfere with things outside that container. *Linux-VServer Paper, Secure Capabilities*.
- [10] Graber, Stéphane (1 January 2014). “LXC 1.0: Security features [6/10]”. Retrieved 12 February 2014. LXC now has support for user namespaces. [...] LXC is no longer running as root so even if an attacker manages to escape the container, he’ d find himself having the privileges of a regular user on the host
- [11] Available since kernel 2.6.18-028stable021. Implementation is based on CFQ disk I/O scheduler, but it is a two-level schema, so I/O priority is not per-process, but rather per-container. See *OpenVZ wiki: I/O priorities for VE* for details.
- [12] Each container can have its own IP addresses, firewall rules, routing tables and so on. Three different networking schemes are possible: route-based, bridge-based, and assigning a real network device (NIC) to a container.

- [13] Each container may have root access without possibly affecting other containers. .
- [14] Available since version 4.0, January 2008.
- [15] Pijewski, Bill. “Our ZFS I/O Throttle” .
- [16] See OpenSolaris Network Virtualization and Resource Control and Network Virtualization and Resource Control (Cross-bow) FAQ for details.
- [17] Cold migration (shutdown-move-restart) is implemented.
- [18] Non-global zones are restricted so they may not affect other zones via a capability-limiting approach. The global zone may administer the non-global zones. (Oracle Solaris 11.1 Administration, Oracle Solaris Zones, Oracle Solaris 10 Zones and Resource Management E29024.pdf, pages 356–360. Available within archive)
- [19] Check the “allow.quotas” option and the “Jails and File Systems” section on the FreeBSD jail man page for details.
- [20] “Hierarchical_Resource_Limits - FreeBSD Wiki” . Wiki.freebsd.org. 2012-10-27. Retrieved 2014-01-15.
- [21] “3.5. Limiting your program's environment” . Freebsd.org. Retrieved 2014-01-15.
- [22] Available since TL 02. See Fix pack information for: WPAR Network Isolation for details.
- [23] See Live Application Mobility in AIX 6.1

10.5 External links

- An introduction to Virtualization
- A short intro to three different virtualization techniques

Chapter 11

Application virtualization

Application virtualization is software technology that encapsulates application software from the underlying operating system on which it is executed. A fully virtualized application is not installed in the traditional sense,* [1] although it is still executed as if it were. The application behaves at runtime like it is directly interfacing with the original operating system and all the resources managed by it, but can be isolated or **sandboxed** to varying degrees.

In this context, the term “virtualization” refers to the artifact being encapsulated (application), which is quite different from its meaning in hardware virtualization, where it refers to the artifact being abstracted (physical hardware).

11.1 Description

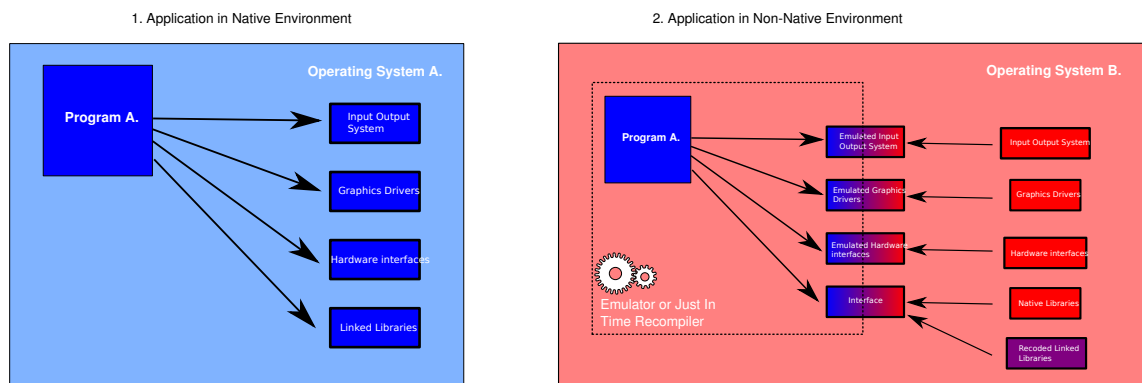


Illustration of an application running in a native environment and running in an application virtualization environment

Modern operating systems such as Microsoft Windows and Linux can include limited application virtualization. For example, Windows 7 provides Windows XP Mode that enables older Windows XP application to run unmodified on Windows 7.

Full application virtualization requires a virtualization layer.* [2] Application virtualization layers replace part of the runtime environment normally provided by the operating system. The layer intercepts all disk operations of virtualized applications and transparently redirects them to a virtualized location, often a single file.* [3] The application remains unaware that it accesses a virtual resource instead of a physical one. Since the application is now working with one file instead of many files spread throughout the system, it becomes easy to run the application on a different computer and previously incompatible applications can be run side-by-side. Examples of this technology for the Windows platform include AppZero, BoxedApp, Cameyo, Ceedo, AppliDis, Evalaze, InstallFree, 2X Software, Citrix XenApp, Systancia, Novell ZENworks Application Virtualization, Numecent Application Jukebox, Microsoft Application Virtualization, Software Virtualization Solution, Spoon (formerly Xenocode), Symantec Workspace Virtualization and Workspace Streaming, VMware ThinApp, P-apps, Sandboxie and Oracle Secure Global Desktop.

11.2 Related technologies

Technology categories that fall under application virtualization include:

- **Application streaming.** Pieces of the application's code, data, and settings are delivered when they're first needed, instead of the entire application being delivered before startup. Running the packaged application may require the installation of a lightweight client application. Packages are usually delivered over a protocol such as HTTP, CIFS or RTSP.* [4]
- **Remote Desktop Services** (also called terminal services, server based computing, and presentation virtualization) is a component of Microsoft Windows that allows a user to access applications and data hosted on a remote computer over a network. Remote Desktop Services sessions run in a single shared server operating system (e.g. Windows Server 2008 R2, Windows Server 2012, etc.) and are accessed using the RDP Remote Desktop Protocol.
- **Desktop virtualization** is an umbrella term that describes software technologies that improve portability, manageability and compatibility of a personal computer's desktop environment by separating part or all of the desktop environment and associated applications from the physical client device that is used to access it. A common implementation of this approach is to host multiple desktop operating system instances on a server hardware platform running a hypervisor. This is generally referred to as “Virtual Desktop Infrastructure” or “VDI” .

11.3 Benefits of application virtualization

- Allows applications to run in environments that do not suit the native application:
 - e.g. Wine allows some Microsoft Windows applications to run on Linux.
 - e.g. CDE, a lightweight application virtualization, allows Linux applications to run in a distribution agnostic way.* [5]* [6]
- May protect the operating system and other applications from poorly written or buggy code and in some cases provide memory protection and IDE style debugging features, for example as in the IBM OLIVER.
- Uses fewer resources than a separate virtual machine.
- Run applications that are not written correctly, for example applications that try to store user data in a read-only system-owned location.
- Run incompatible applications side-by-side, at the same time* [4] and with minimal regression testing against one another.* [7]
- Reduce system integration and administration costs by maintaining a common software baseline across multiple diverse computers in an organization.
- Implement the security principle of least privilege by removing the requirement for end-users to have Administrator privileges in order to run poorly written applications.
- Simplified operating system migrations.* [4]
- Improved security, by isolating applications from the operating system.* [4]
- Allows applications to be copied to portable media and then imported to client computers without need of installing them, so called Portable software.* [6]* [8]

11.4 Limitations of application virtualization

- Not all software can be virtualized. Some examples include applications that require a device driver and 16-bit applications that need to run in shared memory space.*[9]
- Some types of software such as anti-virus packages and applications that require heavy OS integration, such as Stardock's WindowBlinds or TGTSoft's StyleXP are difficult to virtualize.
- Only file and registry-level compatibility issues between legacy applications and newer operating systems can be addressed by application virtualization. For example, applications that don't manage the heap correctly will not execute on Windows Vista as they still allocate memory in the same way, regardless of whether they are virtualized or not.*[10] For this reason, specialist application compatibility fixes (shims) may still be needed, even if the application is virtualized.*[11]
- Moreover, in software licensing, application virtualization bears great licensing pitfalls mainly because both the application virtualization software and the virtualized applications must be correctly licensed.*[12]

11.5 See also

- Application streaming
- Desktop virtualization
- Workspace virtualization
- Portable application creators
- Comparison of application virtual machines
- Emulator
- Software as a service
- Shim (computing)
- Virtual application

11.6 References

- [1] "Microsoft Application Virtualization Technical Overview" . Microsoft.
- [2] Amir Husain. "How to build an Application Virtualization Framework" . VDIworks. Retrieved 2008-07-01.
- [3] Coby Gurr (2008-01-28). "Facilitating Microsoft Windows Vista Migration Through Application Virtualization" (PDF). Dell. Retrieved 2008-06-19.
- [4] "Desktop Virtualization Comes of Age" (PDF). Credit Suisse. 2007-11-26. Retrieved 2008-03-03.
- [5] Guo, Philip J. (2011). "CDE: lightweight application virtualization for Linux" . Retrieved 2012-11-26. *CDE implements a form of lightweight application virtualization that allows you to easily distribute portable software, to deploy applications to the cloud, to make computational experiments reproducible, and to run software on non-native Linux distros without conflicts.*
- [6] Guo, Philip J. (2011-06-01). "CDE: Using System Call Interposition to Automatically Create Portable Software Packages." (pdf). *Proceedings of the 2011 USENIX Annual Technical Conference*. Retrieved 2012-11-26.
- [7] "Overview Series: Windows Vista Application Compatibility" . Microsoft. Retrieved 2008-06-19.
- [8] Domagoj Pernar (October 2009). "Application Virtualization Download repository, and how to make applications portable" . V-irtualization.com. Retrieved 2009-10-30.
- [9] Peter Varhol (September 2007). "Application Virtualization Hits Its Stride" . Redmondmag.com. Archived from the original on 10 June 2008. Retrieved 2008-06-19.

- [10] Adrian Marinescu (2006-07-14). “Windows Vista Heap Management Enhancements” (PDF). Microsoft. Retrieved 2008-06-19.
- [11] Chris Jackson (2008-05-01). “Can You Shim Applications Virtualized in SoftGrid?”. Microsoft. Archived from the original on 2 June 2008. Retrieved 2008-06-24.
- [12] “Licensing pitfalls in application virtualization” . OMTCO Operations Management Technology Consulting GmbH. Retrieved 20 May 2013.

Chapter 12

Portable application

A **portable application (portable app)**, sometimes also called **standalone**, is a program designed to run on a compatible computer without being installed in a way that modifies the computer's configuration information. This type of application can be stored on any storage device, including internal mass storage and external storage such as USB drives and floppy disks – storing its program files and any configuration information and data on the storage medium alone. If no configuration information is required a portable program can be run from read-only storage such as CD-ROMs and DVD-ROMs. Some applications are available in both installable and portable versions.

Like any application, portable applications must be compatible with the computer system hardware and operating system.

Depending on the operating system, *portability* is more or less complex to implement; to operating systems such as AmigaOS, all applications are by definition portable. Portable apps are distinct from *software portability*, source code written to be compilable into different executable programs for different computing platforms.

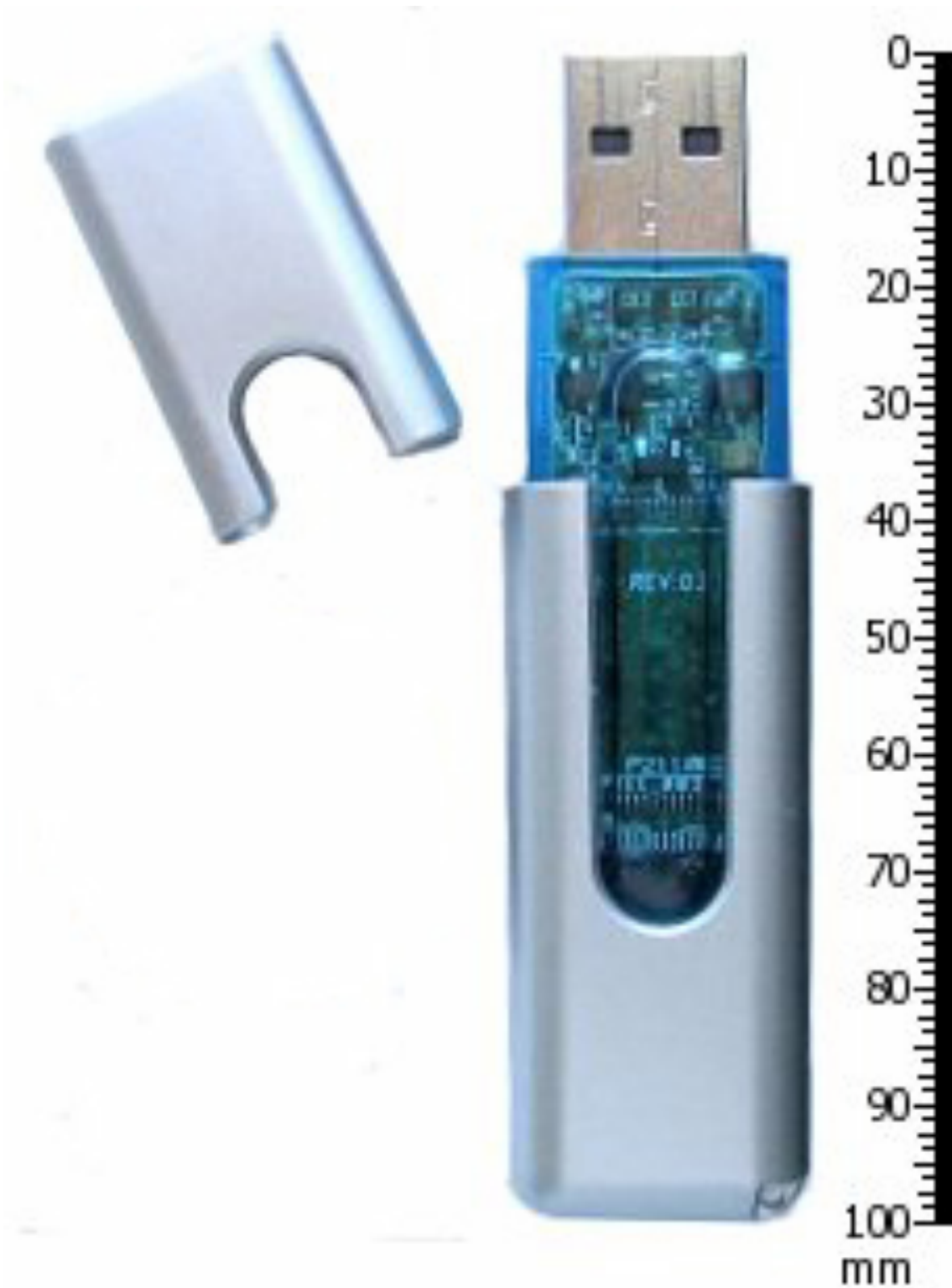
12.1 Portable Windows applications

A portable application does not leave its files or settings on the host computer, which may be convenient or required if you have no administrator privileges on a computer. Typically the application does not write to the Windows registry or store its configuration files (such as an INI file) in the user's profile; instead, it stores its configuration files in the program's directory. Another requirement, since file paths will often differ on changing computers due to variation in Windows drive letter assignments, is the need for applications to store them in a *relative* format. While some applications have options to support this behavior, many programs are not designed to do this. A common technique for such programs is the use of a launcher program to copy necessary settings and files to the host computer when the application starts and move them back to the application's directory when it closes.

An alternative strategy for achieving application portability within Windows, without requiring application source code changes, is **application virtualization**: An application is “sequenced” or “packaged” against a runtime layer that transparently intercepts its file system and registry calls, then redirects these to other persistent storage without the application's knowledge. This approach leaves the application itself unchanged, yet portable.

The same approach is used for individual application components: run-time libraries, COM components or ActiveX, not only for the entire application.* [1] As a result, when individual components are ported in such manner they are able to be: integrated into original portable applications, repeatedly instantiated (virtually installed) with different configurations/settings on the same operating system (OS) without mutual conflicts. As the ported components do not affect the OS-protected related entities (registry and files), the components will not require administrative privileges for installation and management.

Microsoft saw the need for an application-specific registry for its Windows operating system as far back as 2005.* [2] It eventually incorporated some of this technology, using the techniques mentioned above, via its Application Compatibility Database * [3] using its Detours * [4] code library, into Windows XP. It did not, however, make any of this technology available via one of its system APIs.



A USB drive can carry portable applications

12.2 Portability on Linux and UNIX-like systems

See also: Autopackage, RUNZ and Zero Install

Programs written with a Unix-like base in mind often do not make any assumptions. Whereas many Windows

programs assume the user is an administrator—something very prevalent in the days of Windows 95/98/ME (and to some degree in Windows XP/2000, though not in Windows Vista or Windows 7)—such would quickly result in “Permission denied” errors in Unix-like environments since users will be in an unprivileged state much more often. Programs are therefore generally designed around using the HOME environment variable to store settings (e.g. \$HOME/.w3m for the w3m browser). The dynamic linker provides an environment variable LD_LIBRARY_PATH that programs can use to load libraries from non-standard directories. Assuming /mnt contains the portable programs and configuration, a command line may look like:

```
HOME=/mnt/home/user LD_LIBRARY_PATH=/mnt/usr/lib /mnt/usr/bin/w3m www.example.com
```

A Linux application without need for a user-interaction (e.g. adapting a script or environment variable) on varying directory paths can be achieved with the GCC Linker option \$ORIGIN which allows a relative library search path.*[5]

Not all programs honor this – some completely ignore \$HOME and instead do a user look-up in /etc/passwd to find the home directory, therefore thwarting portability.

Some Linux distributions already have native support for portable apps (Super OS, with RUNZ files).

There are also cross-distro package formats that don't require admin rights to run, like Autopackage, CDE or CARE, but with only limited acceptance and support in the Linux community.*[6]*[7]*[8]

12.3 Portable cross-platform applications

Cross-platform portability of applications can be achieved using an abstraction layer. The first well-known one was the Java programming language. One project which focuses on the Java platform is called jPort*[9] and is accessible from the jPort web site. The main goal was to provide free applications written in Java using a Java-enabled application launcher. The concept is based on the assumption that the host system already runs Java Virtual Machine.

Likewise, the concept can be transferred to well-defined scripting languages like Python, Perl, or Ruby, where the application is interpreted instead of compiled to machine code.

12.4 See also

- Comparison of application launchers
- Application virtualization
- Windows To Go
- Ceedo
- WinPenPack
- LiberKey
- List of portable software
- Live USB
- Platform virtualization software
- Portable application creators
- PortableApps.com
- Portable-VirtualBox
- RUNZ
- Spoon (software)
- VMware ThinApp
- Virtual appliance
- U3

12.5 References

- [1] “Portable Application Conversion Technology” . Sphinx Software. Archived from the original on September 7, 2010. Retrieved January 19, 2012.
- [2] “Portable Application Registry” . ip.com. Retrieved January 19, 2012.
- [3] Ionescu, Alex. “Secrets of the Application Compatibility Database (SDB) – Part 1” . Retrieved January 19, 2012.
- [4] “Detours” . Microsoft Research. Retrieved January 19, 2012.
- [5] Hustvedt, Eskild (2009-02-08). “Our new way to meet the LGPL” . Archived from the original on 2009-02-20. Retrieved 2011-03-09. *You can use a special keyword \$ORIGIN to say ‘relative to the actual location of the executable’ . Suddenly we found we could use -rpath \$ORIGIN/lib and it worked. The game was loading the correct libraries, and so was stable and portable, but was also now completely in the spirit of the LGPL as well as the letter!*
- [6] Vining, Nicholas (2010-10-13). “Dear Linux Community: We Need To Talk.” . Gaslamp Games. Retrieved 2011-01-30. *The Linux community, in their infinite wisdom, proceeds to flame the hell out of CDE. [...] “We should all just be using package management.” Here is what I want to say, and let my words be carried down from the mountaintops, written on tiny stone tablets: Package management is not a universal panacea.*
- [7] Byfield, Bruce (2007-02-12). “Autopackage struggling to gain acceptance” . linux.com. Archived from the original on 2008-03-31. Retrieved 2012-01-21. *If Hearn is correct, the real lesson of Autopackage is not how to improve software installation, but the difficulty -- perhaps the impossibility -- of large-scale changes in Linux architecture this late in its history. It's a sobering, disappointing conclusion to a project that once seemed so promising.*
- [8] “AppImages” . Elementary Project. Archived from the original on December 13, 2010. Retrieved January 19, 2012.
- [9] “jPort. Java portable desktop” . jwork.org. Retrieved September 1, 2013..

Chapter 13

Memory virtualization

In computer science, **memory virtualization** decouples volatile random access memory (RAM) resources from individual systems in the data center, and then aggregates those resources into a virtualized memory pool available to any computer in the cluster. The memory pool is accessed by the operating system or applications running on top of the operating system. The distributed memory pool can then be utilized as a high-speed cache, a messaging layer, or a large, shared memory resource for a CPU or a GPU application.

13.1 Description

Memory virtualization allows networked, and therefore distributed, servers to share a pool of memory to overcome physical memory limitations, a common bottleneck in software performance. With this capability integrated into the network, applications can take advantage of a very large amount of memory to improve overall performance, system utilization, increase memory usage efficiency, and enable new use cases. Software on the memory pool nodes (servers) allows nodes to connect to the memory pool to contribute memory, and store and retrieve data. Management software and the technologies of **memory overcommitment** manage shared memory, data insertion, eviction and provisioning policies, data assignment to contributing nodes, and handles requests from client nodes. The memory pool may be accessed at the application level or operating system level. At the application level, the pool is accessed through an API or as a networked file system to create a high-speed shared memory cache. At the operating system level, a page cache can utilize the pool as a very large memory resource that is much faster than local or networked storage.

Memory virtualization implementations are distinguished from **shared memory** systems. Shared memory systems do not permit abstraction of memory resources, thus requiring implementation with a single operating system instance (i.e. not within a clustered application environment).

Memory virtualization is also different from storage based on flash memory such as **solid-state drives (SSDs)** - SSDs and other similar technologies replace hard-drives (networked or otherwise), while memory virtualization replaces or complements traditional RAM.

13.2 Benefits

- Improves memory utilization via the sharing of scarce resources
- Increases efficiency and decreases run time for data intensive and I/O bound applications
- Allows applications on multiple servers to share data without replication, decreasing total memory needs
- Lowers latency and provides faster access than other solutions such as SSD, SAN or NAS

13.3 Products

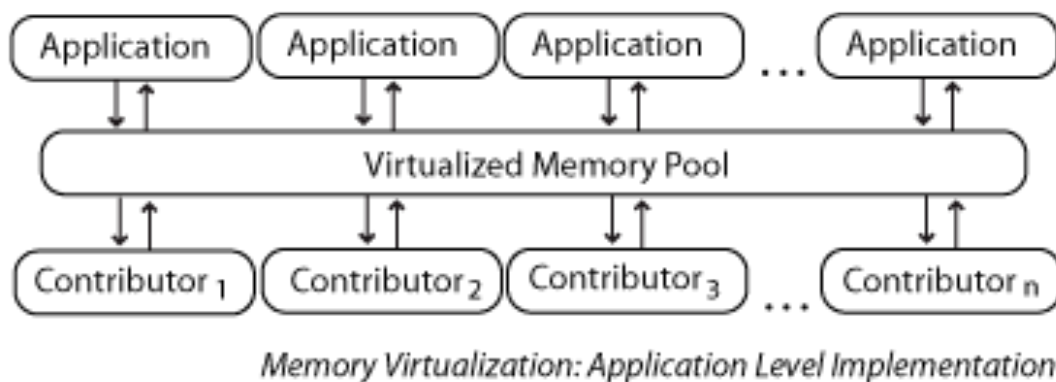
- **RNA networks Memory Virtualization Platform** - A low latency memory pool, implemented as a shared cache and a low latency messaging solution.

- **ScaleMP** - A platform to combine resources from multiple computers for the purpose of creating a single computing instance.
- **Wombat Data Fabric** – A memory based messaging fabric for delivery of market data in financial services.
- **Oracle Coherence** is a Java-based in-memory data-grid product by Oracle
- **AppFabric Caching Service** is a distributed cache platform for in-memory caches spread across multiple systems, developed by Microsoft.
- **IBM Websphere extremeScale** is a Java based distributed cache much like Oracle Coherence
- **GigaSpaces XAP** is a Java based in-memory computing software platform like Oracle Coherence and VMWare Gemfire

13.4 Implementations

13.4.1 Application level integration

In this case, applications running on connected computers connect to the memory pool directly through an API or the file system.



Cluster implementing memory virtualization at the application level. Contributors 1...n contribute memory to the pool. Applications read and write data to the pool using Java or C APIs, or a file system API.

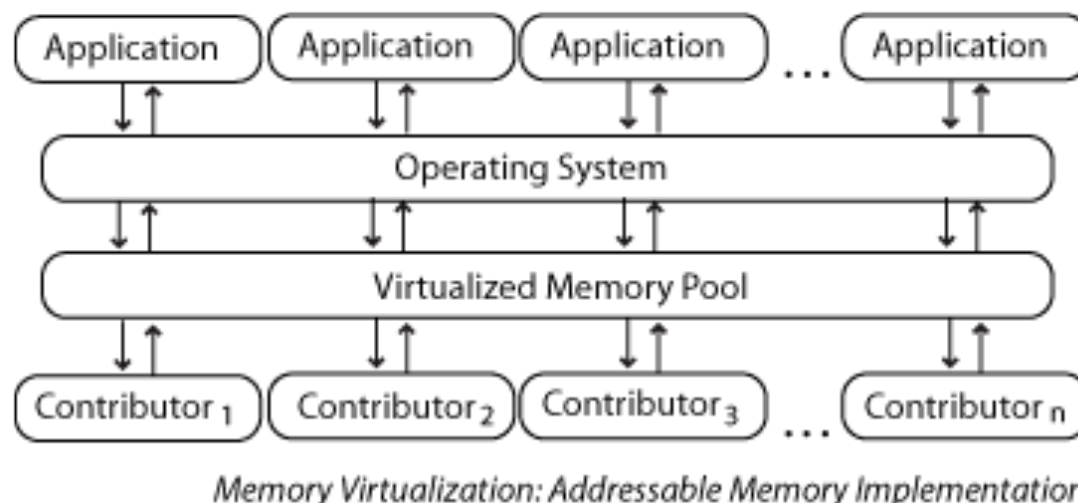
13.4.2 Operating System Level Integration

In this case, the operating system connects to the memory pool, and makes pooled memory available to applications.

13.5 Background

Memory virtualization technology follows from memory management architectures and virtual memory techniques. In both fields, the path of innovation has moved from tightly coupled relationships between logical and physical resources to more flexible, abstracted relationships where physical resources are allocated as needed.

Virtual memory systems abstract between physical RAM and virtual addresses, assigning virtual memory addresses both to physical RAM and to disk-based storage, expanding addressable memory, but at the cost of speed. NUMA and SMP architectures optimize memory allocation within multi-processor systems. While these technologies dynamically manage memory within individual computers, memory virtualization manages the aggregated memory of multiple networked computers as a single memory pool.



Cluster implementing memory virtualization. Contributors 1...n contribute memory to the pool. The operating system connects to the memory pool through the page cache system. Applications consume pooled memory via the operating system.

In tandem with memory management innovations, a number of virtualization techniques have arisen to make the best use of available hardware resources. Application virtualization was demonstrated in mainframe systems first. The next wave was storage virtualization, as servers connected to storage systems such as NAS or SAN in addition to, or instead of, on-board hard disk drives. Server virtualization, or Full virtualization, partitions a single physical server into multiple virtual machines, consolidating multiple instances of operating systems onto the same machine for the purpose of efficiency and flexibility. In both storage and server virtualization, the applications are unaware that the resources they are using are virtual rather than physical, so efficiency and flexibility are achieved without application changes. In the same way, memory virtualization allocates the memory of an entire networked cluster of servers among the computers in that cluster.

13.6 See also

- **Virtual memory** - Traditional memory virtualization on a single computer, typically using the translation lookaside buffer (TLB) to translate between virtual and physical memory addresses
 - Virtual memory management in **Hypervisors** - Hypervisors manage the physical memory of one server, dynamically apportioning memory among operating system instances (VMware ESX, Xen Hypervisor)
- **In Memory Data Grid** - Employs data partitioning in a distributed environment, virtualizing and scaling out the storage of large data sets (Gemfire, GigaSpaces, memcached, Oracle Coherence)
- **In-memory database** - Provides faster and more predictable performance than disk-based databases
- **I/O virtualization** - Creates virtual network and storage endpoints which allow network and storage data to travel over the same fabrics (XSigo I/O Director)
- **Storage virtualization** - Abstracts logical storage from physical storage (NAS, SAN, File Systems (NFS, cluster FS), Volume Management, RAID)
- **Virtualization management hardware** - Hardware solution to accelerate hypervisors (3Leaf Management Solution)
- **RAM disk** - Virtual storage device within a single computer, limited to capacity of local RAM.
- **InfiniBand**
- **10 Gigabit Ethernet**
- **Distributed shared memory**

- Remote direct memory access (RDMA)
- Locality of reference
- Single-system image
- Distributed cache

13.7 References

- Oleg Goldshmidt, Virtualization: Advanced Operating Systems *down*
- Startup RNA Networks Virtualizes Memory Across Multiple Servers, InformationWeek, Published 2/13/2009, Retrieved 3/24/2009
- Five Virtualization Trends to Watch, ComputerWorld, Published 2/3/2009, Retrieved 3/24/2009
- RNA networks and Memory Virtualization, ZDnet, Published 2/2/2009, Retrieved 3/24/2009
- Sorting out the different layers of virtualization, Dan Kusnetzky, ZDnet, Published 1/28/2007, Retrieved 3/24/2009

Chapter 14

Storage virtualization

In computer science, **storage virtualization** uses **virtualization** to enable better functionality and more advanced features in computer data storage systems.

Broadly speaking, a 'storage system' is also known as a storage array or **disk array** or a filer. Storage systems typically use special hardware and software along with disk drives in order to provide very fast and reliable storage for computing and data processing. Storage systems are complex, and may be thought of as a special purpose computer designed to provide storage capacity along with advanced data protection features. Disk drives are only one element within a storage system, along with hardware and special purpose embedded software within the system.

Storage systems can provide either block accessed storage, or file accessed storage. Block access is typically delivered over Fibre Channel, iSCSI, SAS, FICON or other protocols. File access is often provided using NFS or CIFS protocols.

Within the context of a storage system, there are two primary types of virtualization that can occur:

- **Block virtualization** used in this context refers to the abstraction (separation) of **logical storage** (partition) from **physical storage** so that it may be accessed without regard to physical storage or heterogeneous structure. This separation allows the administrators of the storage system greater flexibility in how they manage storage for end users.* [1]
- **File virtualization** addresses the **NAS** challenges by eliminating the dependencies between the data accessed at the file level and the location where the files are physically stored. This provides opportunities to optimize storage use and server consolidation and to perform non-disruptive file migrations.

14.1 Block virtualization

14.1.1 Address space remapping

Virtualization of storage helps achieve location independence by abstracting the physical location of the data. The virtualization system presents to the user a logical space for data storage and handles the process of mapping it to the actual physical location.

It is possible to have multiple layers of virtualization or mapping. It is then possible that the output of one layer of virtualization can then be used as the input for a higher layer of virtualization. Virtualization maps space between back-end resources, to front-end resources. In this instance, 'back-end' refers to a **logical unit number (LUN)** that is not presented to a computer, or host system for direct use. A 'front-end' LUN or volume is presented to a host or computer system for use.

The actual form of the mapping will depend on the chosen implementation. Some implementations may limit the granularity of the mapping which may limit the capabilities of the device. Typical granularities range from a single physical disk down to some small subset (multiples of megabytes or gigabytes) of the physical disk.

In a block-based storage environment, a single block of information is addressed using a LUN identifier and an offset within that LUN - known as a **logical block addressing (LBA)**.

14.1.2 Meta-data

The virtualization software or device is responsible for maintaining a consistent view of all the mapping information for the virtualized storage. This mapping information is often called **meta-data** and is stored as a mapping table.

The address space may be limited by the capacity needed to maintain the mapping table. The level of granularity, and the total addressable space both directly impact the size of the meta-data, and hence the mapping table. For this reason, it is common to have trade-offs, between the amount of addressable capacity and the granularity or access granularity.

One common method to address these limits is to use multiple levels of virtualization. In several storage systems deployed today, it is common to utilize three layers of virtualization.*[2]

Some implementations do not use a mapping table, and instead calculate locations using an algorithm. These implementations utilize dynamic methods to calculate the location on access, rather than storing the information in a mapping table.

14.1.3 I/O redirection

The virtualization software or device uses the meta-data to re-direct I/O requests. It will receive an incoming I/O request containing information about the location of the data in terms of the logical disk (vdisk) and translates this into a new I/O request to the physical disk location.

For example the virtualization device may :

- Receive a read request for vdisk LUN ID=1, LBA=32
- Perform a meta-data look up for LUN ID=1, LBA=32, and finds this maps to physical LUN ID=7, LBA0
- Sends a read request to physical LUN ID=7, LBA0
- Receives the data back from the physical LUN
- Sends the data back to the originator as if it had come from vdisk LUN ID=1, LBA32

14.1.4 Capabilities

Most implementations allow for heterogeneous management of multi-vendor storage devices within the scope of a given implementation's support matrix. This means that the following capabilities are not limited to a single vendor's device (as with similar capabilities provided by specific storage controllers) and are in fact possible across different vendors' devices.

14.1.5 Replication

Main article: [Storage replication](#)

Data replication techniques are not limited to virtualization appliances and as such are not described here in detail. However most implementations will provide some or all of these replication services.

When storage is virtualized, replication services must be implemented above the software or device that is performing the virtualization. This is true because it is only above the virtualization layer that a true and consistent image of the logical disk (vdisk) can be copied. This limits the services that some implementations can implement - or makes them seriously difficult to implement. If the virtualization is implemented in the network or higher, this renders any replication services provided by the underlying storage controllers useless.

- Remote data replication for **disaster recovery**
 - Synchronous Mirroring - where I/O completion is only returned when the remote site acknowledges the completion. Applicable for shorter distances (<200 km)

- Asynchronous Mirroring - where I/O completion is returned before the remote site has acknowledged the completion. Applicable for much greater distances (>200 km)
- Point-In-Time Snapshots to copy or clone data for diverse uses
 - When combined with **thin provisioning**, enables space-efficient snapshots

14.1.6 Pooling

The physical storage resources are aggregated into storage pools, from which the logical storage is created. More storage systems, which may be heterogeneous in nature, can be added as and when needed, and the virtual storage space will scale up by the same amount. This process is fully transparent to the applications using the storage infrastructure.

14.1.7 Disk management

The software or device providing storage virtualization becomes a common disk manager in the virtualized environment. Logical disks (vdisks) are created by the virtualization software or device and are mapped (made visible) to the required host or server, thus providing a common place or way for managing all volumes in the environment.

Enhanced features are easy to provide in this environment :

- Thin Provisioning to maximize storage utilization
 - This is relatively easy to implement as physical storage is only allocated in the mapping table when it is used.
- Disk expansion and shrinking
 - More physical storage can be allocated by adding to the mapping table (assuming the using system can cope with online expansion)
 - Similarly disks can be reduced in size by removing some physical storage from the mapping (uses for this are limited as there is no guarantee of what resides on the areas removed)

14.1.8 Benefits

Non-disruptive data migration

One of the major benefits of abstracting the host or server from the actual storage is the ability to **migrate** data while maintaining concurrent I/O access.

The host only knows about the logical disk (the mapped LUN) and so any changes to the meta-data mapping is transparent to the host. This means the actual data can be moved or replicated to another physical location without affecting the operation of any client. When the data has been copied or moved, the meta-data can simply be updated to point to the new location, therefore freeing up the physical storage at the old location.

The process of moving the physical location is known as **data migration**. Most implementations allow for this to be done in a non-disruptive manner, that is concurrently while the host continues to perform I/O to the logical disk (or LUN).

The mapping granularity dictates how quickly the meta-data can be updated, how much extra capacity is required during the migration, and how quickly the previous location is marked as free. The smaller the granularity the faster the update, less space required and quicker the old storage can be freed up.

There are many day to day tasks a storage administrator has to perform that can be simply and concurrently performed using data migration techniques.

- Moving data off an over-utilized storage device.
- Moving data onto a faster storage device as needs require
- Implementing an Information Lifecycle Management policy
- Migrating data off older storage devices (either being scrapped or off-lease)

Improved utilization

Utilization can be increased by virtue of the pooling, migration, and thin provisioning services. This allows users to avoid over-buying and over-provisioning storage solutions. In other words, this kind of utilization through a shared pool of storage can be easily and quickly allocated as it is needed to avoid constraints on storage capacity that often hinder application performance.* [3]

When all available storage capacity is pooled, system administrators no longer have to search for disks that have free space to allocate to a particular host or server. A new logical disk can be simply allocated from the available pool, or an existing disk can be expanded.

Pooling also means that all the available storage capacity can potentially be used. In a traditional environment, an entire disk would be mapped to a host. This may be larger than is required, thus wasting space. In a virtual environment, the logical disk (LUN) is assigned the capacity required by the using host.

Storage can be assigned where it is needed at that point in time, reducing the need to *guess* how much a given host will need in the future. Using **Thin Provisioning**, the administrator can create a very large thin provisioned logical disk, thus the using system thinks it has a very large disk from day 1. It presents a logical view of the physical storage

Fewer points of management

With storage virtualization, multiple independent storage devices, even if scattered across a network, appear to be a single monolithic storage device and can be managed centrally.

However, traditional storage controller management is still required. That is, the creation and maintenance of **RAID** arrays, including error and fault management.

14.1.9 Risks

Backing out a failed implementation

Once the abstraction layer is in place, only the virtualizer knows where the data actually resides on the physical medium. Backing out of a virtual storage environment therefore requires the reconstruction of the logical disks as contiguous disks that can be used in a traditional manner.

Most implementations will provide some form of back-out procedure and with the data migration services it is at least possible, but time consuming.

Interoperability and vendor support

Interoperability is a key enabler to any virtualization software or device. It applies to the actual physical storage controllers and the hosts, their operating systems, multi-pathing software and connectivity hardware.

Interoperability requirements differ based on the implementation chosen. For example, virtualization implemented within a storage controller adds no extra overhead to host based interoperability, but will require additional support of other storage controllers if they are to be virtualized by the same software.

Switch based virtualization may not require specific host interoperability —if it uses packet cracking techniques to redirect the I/O.

Network based appliances have the highest level of interoperability requirements as they have to interoperate with all devices, storage and hosts.

Complexity

Complexity affects several areas :

- Management of environment : Although a virtual storage infrastructure benefits from a single point of logical disk and replication service management, the physical storage must still be managed. Problem determination and fault isolation can also become complex, due to the abstraction layer.

- Infrastructure design : Traditional design ethics may no longer apply, virtualization brings a whole range of new ideas and concepts to think about (as detailed here)
- The software or device itself : Some implementations are more complex to design and code - network based, especially in-band (symmetric) designs in particular —these implementations actually handle the I/O requests and so latency becomes an issue.

Meta-data management

Information is one of the most valuable assets in today's business environments. Once virtualized, the meta-data are the glue in the middle. If the meta-data are lost, so is all the actual data as it would be virtually impossible to reconstruct the logical drives without the mapping information.

Any implementation must ensure its protection with appropriate levels of back-ups and replicas. It is important to be able to reconstruct the meta-data in the event of a catastrophic failure.

The meta-data management also has implications on performance. Any virtualization software or device must be able to keep all the copies of the meta-data atomic and quickly updateable. Some implementations restrict the ability to provide certain fast update functions, such as point-in-time copies and caching where super fast updates are required to ensure minimal latency to the actual I/O being performed.

Performance and scalability

In some implementations the performance of the physical storage can actually be improved, mainly due to caching. Caching however requires the visibility of the data contained within the I/O request and so is limited to in-band and symmetric virtualization software and devices. However these implementations also directly influence the latency of an I/O request (cache miss), due to the I/O having to flow through the software or device. Assuming the software or device is efficiently designed this impact should be minimal when compared with the latency associated with physical disk accesses.

Due to the nature of virtualization, the mapping of logical to physical requires some processing power and lookup tables. Therefore every implementation will add some small amount of latency.

In addition to response time concerns, throughput has to be considered. The bandwidth into and out of the meta-data lookup software directly impacts the available system bandwidth. In asymmetric implementations, where the meta-data lookup occurs before the information is read or written, bandwidth is less of a concern as the meta-data are a tiny fraction of the actual I/O size. In-band, symmetric flow through designs are directly limited by their processing power and connectivity bandwidths.

Most implementations provide some form of scale-out model, where the inclusion of additional software or device instances provides increased scalability and potentially increased bandwidth. The performance and scalability characteristics are directly influenced by the chosen implementation.

14.1.10 Implementation approaches

- Host-based
- Storage device-based
- Network-based

Host-based

Main article: [Logical volume management](#)

Host-based virtualization requires additional software running on the host, as a privileged task or process. In some cases volume management is built into the operating system, and in other instances it is offered as a separate product. Volumes (LUN's) presented to the host system are handled by a traditional physical device driver. However, a software

layer (the volume manager) resides above the disk device driver intercepts the I/O requests, and provides the meta-data lookup and I/O mapping.

Most modern operating systems have some form of logical volume management built-in (in Linux called **Logical Volume Manager** or LVM; in Solaris and FreeBSD, ZFS's zpools layer; in Windows called **Logical Disk Manager** or LDM), that performs virtualization tasks.

Note: Host based volume managers were in use long before the term *storage virtualization* had been coined.

Pros

- Simple to design and code
- Supports any storage type
- Improves storage utilization without thin provisioning restrictions

Cons

- Storage utilization optimized only on a per host basis
- Replication and data migration only possible locally to that host
- Software is unique to each operating system
- No easy way of keeping host instances in sync with other instances
- Traditional Data Recovery following a server disk drive crash is impossible

Specific examples

- Technologies:
 - Logical volume management
 - File systems, e.g., (hard links, CIFS/NFS)
 - Automatic mounting, e.g., (autofs)

Storage device-based

Like host-based virtualization, several categories have existed for years and have only recently been classified as virtualization. Simple data storage devices, like single **hard disk drives**, do not provide any virtualization. But even the simplest **disk arrays** provide a logical to physical abstraction, as they use **RAID** schemes to join multiple disks in a single array (and possibly later divide the array it into smaller volumes).

Advanced disk arrays often feature cloning, snapshots and remote replication. Generally these devices do not provide the benefits of data migration or replication across heterogeneous storage, as each vendor tends to use their own proprietary protocols.

A new breed of disk array controllers allows the downstream attachment of other storage devices. For the purposes of this article we will only discuss the later style which do actually virtualize other storage devices.

Concept A primary storage controller provides the services and allows the direct attachment of other storage controllers. Depending on the implementation these may be from the same or different vendors.

The primary controller will provide the pooling and meta-data management services. It may also provide replication and migration services across those controllers which it is .

Pros

- No additional hardware or infrastructure requirements
- Provides most of the benefits of storage virtualization
- Does not add latency to individual I/Os

Cons

- Storage utilization optimized only across the connected controllers
- Replication and data migration only possible across the connected controllers and same vendors device for long distance support
- Downstream controller attachment limited to vendors support matrix
- I/O Latency, non cache hits require the primary storage controller to issue a secondary downstream I/O request
- Increase in storage infrastructure resource, the primary storage controller requires the same bandwidth as the secondary storage controllers to maintain the same throughput

Network-based

Storage virtualization operating on a network based device (typically a standard server or smart switch) and using iSCSI or FC Fibre channel networks to connect as a SAN. These types of devices are the most commonly available and implemented form of virtualization.

The virtualization device sits in the SAN and provides the layer of abstraction between the hosts performing the I/O and the storage controllers providing the storage capacity.

Pros

- True heterogeneous storage virtualization
- Caching of data (performance benefit) is possible when in-band
- Single management interface for all virtualized storage
- Replication services across heterogeneous devices

Cons

- Complex interoperability matrices - limited by vendors support
- Difficult to implement fast meta-data updates in switched-based devices
- Out-of-band requires specific host based software
- In-band may add latency to I/O
- In-band the most complication to design and code

Appliance-based vs. switch-based There are two commonly available implementations of network-based storage virtualization, **appliance-based** and **switch-based**. Both models can provide the same services, disk management, metadata lookup, data migration and replication. Both models also require some processing hardware to provide these services.

Appliance based devices are dedicated hardware devices that provide SAN connectivity of one form or another. These sit between the hosts and storage and in the case of in-band (symmetric) appliances can provide all of the benefits and services discussed in this article. I/O requests are targeted at the appliance itself, which performs the meta-data mapping before redirecting the I/O by sending its own I/O request to the underlying storage. The in-band appliance can also provide caching of data, and most implementations provide some form of clustering of individual appliances to maintain an atomic view of the metadata as well as cache data.

Switch based devices, as the name suggests, reside in the physical switch hardware used to connect the SAN devices. These also sit between the hosts and storage but may use different techniques to provide the metadata mapping, such as packet cracking to snoop on incoming I/O requests and perform the I/O redirection. It is much more difficult to ensure atomic updates of metadata in a switched environment and services requiring fast updates of data and metadata may be limited in switched implementations.

In-band vs. out-of-band

In-band, also known as *symmetric*, virtualization devices actually sit in the data path between the host and storage. All I/O requests and their data pass through the device. Hosts perform I/O to the virtualization device and never interact with the actual storage device. The virtualization device in turn performs I/O to the storage device. Caching of data, statistics about data usage, replications services, data migration and thin provisioning are all easily implemented in an in-band device.

Out-of-band, also known as *asymmetric*, virtualization devices are sometimes called **meta-data servers**. These devices only perform the meta-data mapping functions. This requires additional software in the host which knows to first request the location of the actual data. Therefore an I/O request from the host is intercepted before it leaves the host, a meta-data lookup is requested from the meta-data server (this may be through an interface other than the SAN) which returns the physical location of the data to the host. The information is then retrieved through an actual I/O request to the storage. Caching is not possible as the data never passes through the device.

14.2 File based virtualization

See also: [Virtual file system](#), [Installable File System](#) and [Filesystem in Userspace](#)

14.3 See also

- [Archive](#)
- [Automated tiered storage](#)
- [Storage hypervisor](#)
- [Storage Virtualization – Specific Implementations](#)
- [Backup](#)
- [Computer data storage](#)
- [Data proliferation](#)
- [Disk storage](#)
- [Information Lifecycle Management](#)
- [Information repository](#)

- Magnetic tape data storage
- Repository
- Spindle

14.4 References

[1] SearchStorage.com Definitions

[2] Need Citation

[3] “Stop Over-Provisioning with Storage Resource Management” . Dell.com. Retrieved 2012-06-30.

14.5 External links

- InfoStor Article - Virtualization, the foundation of data center transformation

Chapter 15

Network virtualization

In computing, **network virtualization** is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a **virtual network**. Network virtualization involves **platform virtualization**, often combined with resource virtualization.

Network virtualization is categorized as either **external virtualization**, combining many networks or parts of networks into a virtual unit, or **internal virtualization**, providing network-like functionality to software containers on a single network server.

In software testing, software developers use network virtualization to test software under development in a simulation of the network environments in which the software is intended to operate. As a component of application performance engineering, network virtualization enables developers to emulate connections between applications, services, dependencies, and end users in a test environment without having to physically test the software on all possible hardware or system software. Of course, the validity of the test depends on the accuracy of the network virtualization in emulating real hardware and operating systems.

15.1 Components

Various equipment and software vendors offer network virtualization by combining any of the following:

- Network hardware, such as switches and **network adapters**, also known as network interface cards (NICs)
- Network elements, such as firewalls and load balancers
- Networks, such as **virtual LANs (VLANs)** and containers such as **virtual machines (VMs)**
- Network storage devices
- Network machine-to-machine elements, such as telecommunications devices
- Network mobile elements, such as laptop computers, tablet computers, and smart phones
- Network media, such as **Ethernet** and **Fibre Channel**

15.2 External virtualization

External network virtualization combines or subdivides one or more **local area networks (LANs)** into virtual networks to improve a large network's or data center's efficiency. A virtual local area network (VLAN) and **network switch** comprise the key components. Using this technology, a **system administrator** can configure systems physically attached to the same local network into separate virtual networks. Conversely, an administrator can combine systems on separate local networks into a VLAN spanning the segments of a large network.

15.3 Internal virtualization

also called Virtual Channel **Internal network virtualization** configures a single system with software containers, such as Xen hypervisor control programs, or pseudo-interfaces, such as a VNIC, to emulate a physical network with software. This can improve a single system's efficiency by isolating applications to separate containers or pseudo-interfaces.* [1]

15.3.1 Examples

Citrix and Vyatta have built a virtual network protocol stack combining Vyatta's routing, firewall, and VPN functions with Citrix's Netscaler load balancer, branch repeater wide area network (WAN) optimization, and secure sockets layer VPN.

OpenSolaris network virtualization provides a so-called “network in a box” (see OpenSolaris Network Virtualization and Resource Control).

Microsoft Virtual Server uses virtual machines to make a “network in a box” for x86 systems. These containers can run different operating systems, such as Microsoft Windows or Linux, either associated with or independent of a specific network interface controller (NIC).

15.4 Use in testing

Network virtualization may be used in application development and testing to mimic real-world hardware and system software. In **application performance engineering**, network virtualization enables emulation of connections between applications, services, dependencies, and end users for software testing.

15.5 Wireless network virtualization

Wireless network virtualization can have a very broad scope ranging from spectrum sharing, infrastructure virtualization, to air interface virtualization. Similar to wired network virtualization, in which physical infrastructure owned by one or more providers can be shared among multiple service providers, wireless network virtualization needs the physical wireless infrastructure and radio resources to be abstracted and isolated to a number of virtual resources, which then can be offered to different service providers. In other words, virtualization, regardless of wired or wireless networks, can be considered as a process splitting the entire network system. However, the distinctive properties of the wireless environment, in terms of time-various channels, attenuation, mobility, broadcast, etc., make the problem more complicated. Furthermore, wireless network virtualization depends on specific access technologies, and wireless network contains much more access technologies compared to wired network virtualization and each access technology has its particular characteristics, which makes convergence, sharing and abstraction difficult to achieve. Therefore, it may be inaccurate to consider wireless network virtualization as a subset of network virtualization.* [2]

15.6 External links

- Global Environment for Network Innovations
- Future Internet Research and Experimentation
- AKARI Project

15.7 See also

- 2X_Software
- Application performance engineering

- Avocent
- I/O virtualization
- IEEE 802.1aq - Shortest Path Bridging (SPB)
- Network Functions Virtualization
- Network switch
- NVGRE
- Platform virtualization
- Virtual LAN
- Virtual machine
- Virtual private network
- VXLAN

15.8 Further reading

- Esposito, Flavio; Matta, Ibrahim; Ishakian, Vatche (2011). “Slice Embedding Solutions for Distributed Service Architectures” . *ACM Computing Surveys* **26** (1). Retrieved March 2014.
- Chowdhury, N.M. Mosharaf Kabir; Boutaba, Raouf (2010). “A survey of network virtualization” . *Computer Networks* **54** (5): 862–876. doi:10.1016/j.comnet.2009.10.017. ISSN 1389-1286.
- Berl, Andreas; Fischer, Andreas; de Meer, Hermann (2009). “Using System Virtualization to Create Virtualized Networks” . *Electronic Communications of the EASST* **17**: 1–12. ISSN 1863-2122.
- Fischer, Andreas; Botero, Juan Felipe; Beck, Michael Till; de Meer, Hermann; Hesselbach, Xavier (2013). “Virtual Network Embedding: A Survey” . *IEEE Communications Surveys & Tutorials*: 1–19. doi:10.1109/SURV.2013.013013.0 ISSN 1553-877X.

15.9 References

- Victor Moreno and Kumar Reddy (2006). *Network Virtualization*. Indianapolis: Cisco Press.
- NetworkVirtualization.com | News retrieved 3 June 2008

[1] A. Galis, S. Clayman, A. Fischer, A. Paler, Y. Al-Hazmi, H. De Meer, A. Cheniour, O. Mornard, J. Patrick Gelas and L. Lefevre, et al. “Future Internet Management Platforms for Network Virtualisation and Service Clouds”- ServiceWave 2010, December 2010, <http://servicewave.eu/2010/joint-demonstration-evening/> and in “Towards A Service-Based Internet” Lecture Notes in Computer Science, 2010, Volume 6481/2010, 235-237, doi:10.1007/978-3-642-17694-4_39

[2]

Chapter 16

Software-defined networking

Not to be confused with ISDN (Integrated Services Digital Network).

Software-defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of lower-level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The inventors and vendors of these systems claim that this simplifies networking.*[1]

SDN requires some method for the control plane to communicate with the data plane. One such mechanism, OpenFlow, is often misunderstood to be equivalent to SDN, but other mechanisms could also fit into the concept.

16.1 History

The origins of software-defined networking (SDN) began shortly after Sun Microsystems released Java in 1995.*[2]*[3]

One of the first and most notable SDN projects was AT&T's GeoPlex.*[4] AT&T Labs Geoplex project members Michah Lerner, George Vanecek, Nino Vidovic, Dado Vrsalovic leveraged the network APIs and dynamic aspects of the Java language as a means to implement middleware networks. "GeoPlex is not an operating system, nor does it attempt to compete with one. It is networking middleware that uses one or more operating systems running on computers connected to the Internet. GeoPlex is a service platform that manages networks and on-line services. .. GeoPlex maps all of the IP network activities into one or more services" *[5]

As noted, GeoPlex did not concern itself with operating systems running on networking hardware switches, and routers. AT&T wanted a "soft switch" that could reconfigure physical switches in the network and load them with new services from an OSS. However when provisioning services GeoPlex could not reach deeply into the physical devices to perform reconfiguration. The operating systems running on networked devices in the physical network therefore became a barrier to early SDN-like service delivery.

In 1998, Mark Medovich, a senior scientist of Sun Microsystems and Javasoft, left Sun to launch a Silicon Valley soft switch startup WebSprocket. Medovich designed a new network operating system, and an object oriented structured runtime model that could be modified by a networked compiler and class loader in real time. With this approach, applications could be written with Java threads that inherited WebSprocket kernel, network, and device classes and later modified by the networked compiler/class-loader. WebSprocket's platform was designed such that devices had the ability to instantiate network stack(s), interfaces, and protocols as multiple threads.*[6]

In July 2000, WebSprocket released VMFoundry, the Java to bare metal structured runtime compiler, and VM-Server, a networked device compiler/classloader application server.*[7] Custom networked devices were preloaded with images created by VMFoundry then deployed on the network and connected to VMServer via UDP or TCP services plane, which could proactively or reactively load or extended network protocol methods and classes on the target system. WebSprocket's version of SDN, therefore was not confined to a set of limited actions managed by an SDN controller. Rather, WebSprocket's "control plane" contained code that could change, override, extend, or enhance Network protocols on operating networked systems.*[8] Bill Yount (Stanford University Network) visited WebSprocket's Sunnyvale lab to see a demonstration and expressed great enthusiasm about by the entire concept,

especially the VMServer (SDN Controller) and prophetically stated SDN (WebSprocket) as " 10 years ahead of its time" . In Summer of 2000, Ericsson's advanced network research engineers saw an immediate need and visited WebSprocket to design and architect features of a next generation soft switch thus taking first steps to build the world's first commercial soft switch.

Sometime during 2000, the Gartner Group recognized the emergence of programmable networks as the next big thing for the Internet and introduced the "Supranet", the fusion of the physical and the digital (virtual) worlds as "internet of things" . and by October 2000 the Gartner Group selected WebSprocket as one of the top emerging technologies in the world.*[9]

In early 2001, Ericsson and WebSprocket entered into a license contract to create the first commercial soft switch. Ericsson's entire (SCS) call control software stack was ported by Joe Kulig (WebSprocket) in a matter of days, a feat that astounded Ericsson. An international consortium was formed to develop standards for the "Supranet" . In March 2001, Kurt Dewitt, Supranet Consortium Chairman and Business Development Director for Ericsson's Data Broadband and Optical Networks Division, announced the selection of WebSprocket as the enabling technology of the Supranet Transaction Server (STS), a comprehensive framework to deliver any networked service.*[10]

In April and May 2001, Anjaneya Prasad Calyam, a graduate student at the Ohio State University and researcher at OARnet, ran the first SDN test and developed the first practical SDN use case for Internet2. After successful completion of Calyam's tests, OARnet issued the following statement on May 8, 2001:

“We have witnessed the successful first step to the fulfillment of smart, interoperable networks through the deployment of Supranet Transaction Server. A technology first was accomplished as a new set of instructions was dynamically transmitted across the network, changing the behavior of the requesting computer. There was no need to take down any part of the system and there was no interruption of service. Our testing will continue and we anticipate further advancement of the next generation Internet through our partnership with Websprocket” – Pankaj Shah (Managing Director, OARnet)*[11]

Practical use cases of SDN as reconfigurable and extensible protocol services delivery from WebSprocket's platform are contemplated in Calyam's Masters Thesis. *[12]

The telecom market deflated in 2001 and Ericsson's soft switch development program came to an end, thus stalling the only known commercial SDN soft switch R&D effort at that time.

Software-defined networking (SDN) was continued with work done in 2003 by Bob Burke and Zac Carman developing the Content Delivery Control Network patent application that eventually was issued as two US patents: 8,122,128 *[13] and 8,799,468.*[14] In this seminal inception, SDN, named service preference architecture (SPA) in their patent, was described as a collection of network embedded computing techniques used to control the operation of Network Elements, namely content servers, routers, switches and gateways, with the objective being to safeguard content from theft (P2P) or unwanted interception and to efficiently deliver content for paid services. CableLabs later specified Digital Cable and CableCARD using what we now know as SDN, which debuted in 2007. SDN was again moved ahead in work done at UC Berkeley and Stanford University around 2008.*[15]

The Open Networking Foundation was founded in 2011 to promote SDN and OpenFlow.

At the 2014 Interop and Tech Field Day, software-defined networking was demonstrated by Avaya using Shortest path bridging and OpenStack as an automated campus, extending automation from the data center to the end device, removing manual provisioning from service delivery.*[16]*[17]

16.2 Concept

Software-defined networking (SDN) is an architecture purporting to be dynamic, manageable, cost-effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications. SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.*[18]

The OpenFlow protocol is a foundational element for building SDN solutions. The SDN architecture is:

- *Directly programmable*: Network control is directly programmable because it is decoupled from forwarding functions.

- *Agile*: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- *Centrally managed*: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- *Programmatically configured*: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- *Open standards-based and vendor-neutral*: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

16.3 Limitations of other networking technologies

Meeting current market requirements is virtually impossible with traditional network architectures. Faced with flat or reduced budgets, enterprise IT departments are trying to squeeze the most from their networks using device-level management tools and manual processes. Carriers face similar challenges as demand for mobility and bandwidth explodes; profits are being eroded by escalating capital equipment costs and flat or declining revenue. Existing network architectures were not designed to meet the requirements of today's users, enterprises, and carriers; rather network designers are constrained by the limitations of current networks, which include:

Complexity that leads to stasis Networking technology to date has consisted largely of discrete sets of protocols designed to connect hosts reliably over arbitrary distances, link speeds, and topologies. To meet business and technical needs over the last few decades, the industry has evolved networking protocols to deliver higher performance and reliability, broader connectivity, and more stringent security. Protocols tend to be defined in isolation, however, with each solving a specific problem and without the benefit of any fundamental abstractions. This has resulted in one of the primary limitations of today's networks: complexity. For example, to add or move any device, IT must touch multiple switches, routers, firewalls, Web authentication portals, etc. and update ACLs, VLANs, quality of services (QoS), and other protocol-based mechanisms using device-level management tools. In addition, network topology, vendor switch model, and software version all must be taken into account.

Due to this complexity, today's networks are relatively static as IT seeks to minimize the risk of service disruption. The static nature of networks is in stark contrast to the dynamic nature of today's server environment, where server virtualization has greatly increased the number of hosts requiring network connectivity and fundamentally altered assumptions about the physical location of hosts. Prior to virtualization, applications resided on a single server and primarily exchanged traffic with select clients. Today, applications are distributed across multiple virtual machines (VMs), which exchange traffic flows with each other. VMs migrate to optimize and rebalance server workloads, causing the physical end points of existing flows to change (sometimes rapidly) over time. VM migration challenges many aspects of traditional networking, from addressing schemes and namespaces to the basic notion of a segmented, routing-based design.

In addition to adopting virtualization technologies, many enterprises today operate an IP converged network for voice, data, and video traffic. While existing networks can provide differentiated QoS levels for different applications, the provisioning of those resources is highly manual. IT must configure each vendor's equipment separately, and adjust parameters such as network bandwidth and QoS on a per-session, per-application basis. Because of its static nature, the network cannot dynamically adapt to changing traffic, application, and user demands.

Inconsistent policies To implement a network-wide policy, IT may have to configure thousands of devices and mechanisms. For example, every time a new virtual machine is brought up, it can take hours, in some cases days, for IT to reconfigure ACLs across the entire network. The complexity of today's networks makes it very difficult for IT to apply a consistent set of access, security, QoS, and other policies to increasingly mobile users, which leaves the enterprise vulnerable to security breaches, non-compliance with regulations, and other negative consequences.

Inability to scale As demands on the data center rapidly grow, so too must the network grow. However, the network becomes vastly more complex with the addition of hundreds or thousands of network devices that must be configured and managed. IT has also relied on link oversubscription to scale the network, based on predictable traffic patterns; however, in today's virtualized data centers, traffic patterns are incredibly dynamic and therefore unpredictable.

Mega-operators, such as Google, Yahoo!, and Facebook, face even more daunting scalability challenges. These service providers employ large-scale parallel processing algorithms and associated datasets across their entire computing pools. As the scope of end-user applications increases (for example, crawling and indexing the entire World Wide Web to instantly return search results to users), the number of computing elements explodes and data-set exchanges among compute nodes can reach petabytes. These companies need so-called hyperscale networks that can provide high-performance, low-cost connectivity among hundreds of thousands—potentially millions—of physical servers. Such scaling cannot be done with manual configuration.

To stay competitive, carriers must deliver ever-higher value, better-differentiated services to customers. Multi-tenancy further complicates their task, as the network must serve groups of users with different applications and different performance needs. Key operations that appear relatively straightforward, such as steering a customer's traffic flows to provide customized performance control or on-demand delivery, are very complex to implement with existing networks, especially at carrier scale. They require specialized devices at the network edge, thus increasing capital and operational expenditure as well as time-to-market to introduce new services.

Vendor dependence Carriers and enterprises seek to deploy new capabilities and services in rapid response to changing business needs or user demands. However, their ability to respond is hindered by vendors' equipment product cycles, which can range to three years or more. Lack of standard, open interfaces limits the ability of network operators to tailor the network to their individual environments.

This mismatch between market requirements and network capabilities has brought the industry to a tipping point. In response, the industry has created the software-defined networking (SDN) architecture and is developing associated standards.

16.4 The need for a new network architecture

The explosion of mobile devices and content, server virtualization, and advent of cloud services are among the trends driving the networking industry to reexamine traditional network architectures.* [19] Many conventional networks are hierarchical, built with tiers of Ethernet switches arranged in a tree structure. This design made sense when client-server computing was dominant, but such a static architecture is ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments. Some of the key computing trends driving the need for a new network paradigm include:

Changing traffic patterns Within the enterprise data center, traffic patterns have changed significantly. In contrast to client-server applications where the bulk of the communication occurs between one client and one server, today's applications access different databases and servers, creating a flurry of “east-west” machine-to-machine traffic before returning data to the end user device in the classic “north-south” traffic pattern. At the same time, users are changing network traffic patterns as they push for access to corporate content and applications from any type of device (including their own), connecting from anywhere, at any time. Finally, many enterprise data centers managers are contemplating a utility computing model, which might include a private cloud, public cloud, or some mix of both, resulting in additional traffic across the wide area network.

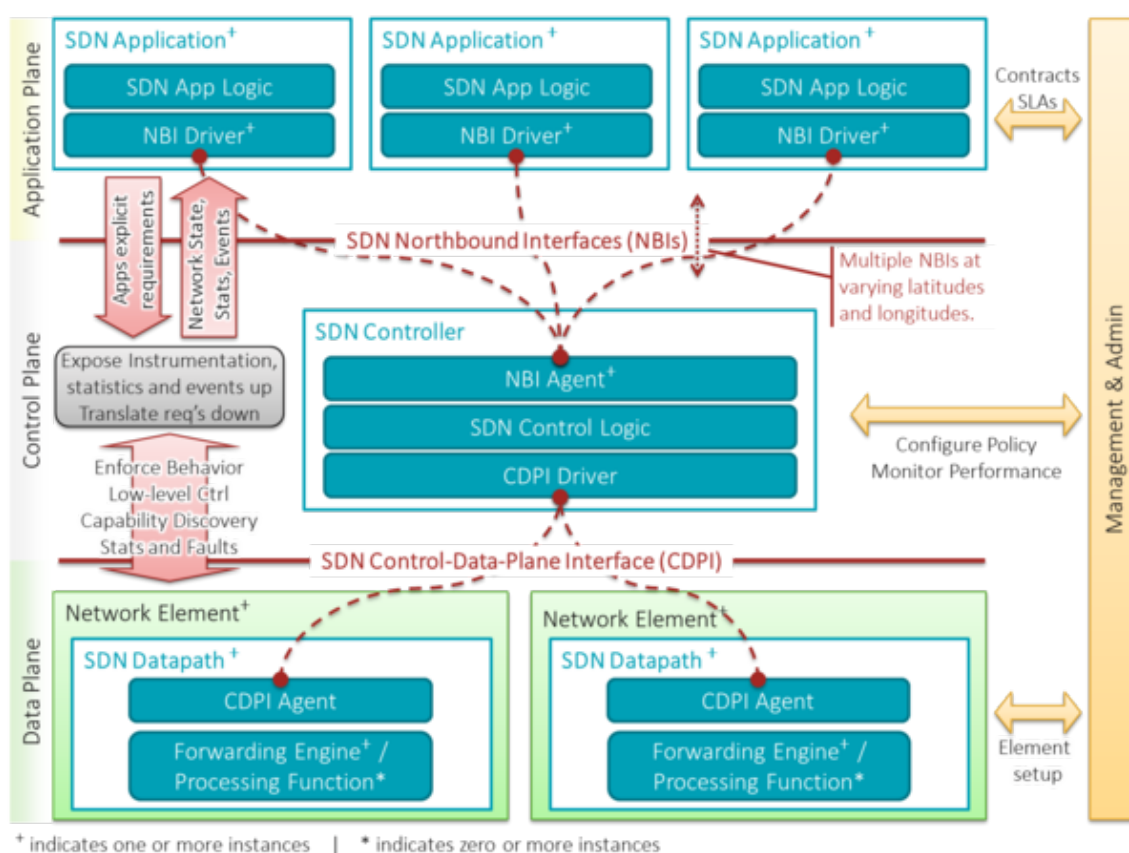
The “consumerization of IT” Users are increasingly employing mobile personal devices such as smartphones, tablets, and notebooks to access the corporate network. IT is under pressure to accommodate these personal devices in a fine-grained manner while protecting corporate data and intellectual property and meeting compliance mandates.

The rise of cloud services Enterprises have enthusiastically embraced both public and private cloud services, resulting in unprecedented growth of these services. Enterprise business units now want the agility to access applications, infrastructure, and other IT resources on demand and à la carte. To add to the complexity, IT's planning for cloud services must be done in an environment of increased security, compliance, and auditing

requirements, along with business reorganizations, consolidations, and mergers that can change assumptions overnight. Providing self-service provisioning, whether in a private or public cloud, requires elastic scaling of computing, storage, and network resources, ideally from a common viewpoint and with a common suite of tools.

“Big data” means more bandwidth Handling today’s “big data” or mega datasets requires massive parallel processing on thousands of servers, all of which need direct connections to each other. The rise of mega datasets is fueling a constant demand for additional network capacity in the data center. Operators of hyperscale data center networks face the daunting task of scaling the network to previously unimaginable size, maintaining any-to-any connectivity without going broke.

16.5 Architectural components



A high-level overview of the software-defined networking architecture

The following list defines and explains the architectural components:*[20]

SDN Application (SDN App) SDN Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller via a northbound interface (NBI). In addition they may consume an abstracted view of the network for their internal decision making purposes. An SDN Application consists of one SDN Application Logic and one or more NBI Drivers. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

SDN Controller The SDN Controller is a logically centralized entity in charge of (i) translating the requirements from the SDN Application layer down to the SDN Datapaths and (ii) providing the SDN Applications with an abstract view of the network (which may include statistics and events). An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver. Definition

as a logically centralized entity neither prescribes nor precludes implementation details such as the federation of multiple controllers, the hierarchical connection of controllers, communication interfaces between controllers, nor virtualization or slicing of network resources.

SDN Datapath The SDN Datapath is a logical network device that exposes visibility and uncontended control over its advertised forwarding and data processing capabilities. The logical representation may encompass all or a subset of the physical substrate resources. An SDN Datapath comprises a CDPI agent and a set of one or more traffic forwarding engines and zero or more traffic processing functions. These engines and functions may include simple forwarding between the datapath's external interfaces or internal traffic processing or termination functions. One or more SDN Datapaths may be contained in a single (physical) network element—an integrated physical combination of communications resources, managed as a unit. An SDN Datapath may also be defined across multiple physical network elements. This logical definition neither prescribes nor precludes implementation details such as the logical to physical mapping, management of shared physical resources, virtualization or slicing of the SDN Datapath, interoperability with non-SDN networking, nor the data processing functionality, which can include L4-7 functions.

SDN Control to Data-Plane Interface (CDPI) The SDN CDPI is the interface defined between an SDN Controller and an SDN Datapath, which provides at least (i) programmatic control of all forwarding operations, (ii) capabilities advertisement, (iii) statistics reporting, and (iv) event notification. One value of SDN lies in the expectation that the CDPI is implemented in an open, vendor-neutral and interoperable way.

SDN Northbound Interfaces (NBI) SDN NBIs are interfaces between SDN Applications and SDN Controllers and typically provide abstract network views and enable direct expression of network behavior and requirements. This may occur at any level of abstraction (latitude) and across different sets of functionality (longitude). One value of SDN lies in the expectation that these interfaces are implemented in an open, vendor-neutral and interoperable way.

16.6 SDN deployment models

Proactive vs Reactive* [21]* [22] If flows arrive at a switch, a flow table lookup is performed. Depending on the flow table implementation this is done in a software flow table if a vSwitch is used or in an ASIC if it's implemented in hardware. In the case when no matching flow is found a request to the controller for further instructions is sent. In reactive mode the controller acts after these requests and creates and installs a rule in the flow table for the corresponding packet if necessary. In proactive mode the controller populates flow table entries for all possible traffic matches possible for this switch in advance. This mode can be compared with typical routing table entries today, where all static entries are installed ahead in time. Following this no request is sent to the controller since all incoming flows will find a matching entry. A major advantage in proactive mode is that all packets are forwarded in line rate (considering all flow table entries in TCAM) and no delay is added.

In addition there exists a hybrid mode that follows the flexibility of a reactive mode for a set of traffic and the low-latency forwarding (proactive mode) for the rest of the traffic.

16.7 Applications

One application of SDN is the *infrastructure as a service* (IaaS).

This extension means that SDN virtual networking combined with virtual compute (VMs) and virtual storage can emulate elastic resource allocation as if each such enterprise application was written like a *Google* or *Facebook* application. In the vast majority of these applications resource allocation is statically mapped in inter process communication (IPC). However if such mapping can be expanded or reduced to large (many cores) or small VMs the behavior would be much like one of the purpose built large Internet applications.

Other uses in the consolidated data-center include consolidation of spare capacity stranded in static partition of racks to pods. Pooling these spare capacities results in significant reduction of computing resources. Pooling the active resources increases average utilization.

The use of SDN distributed and global edge control also includes the ability to balance load on lots of links leading from the racks to the switching spine of the data-center. Without SDN this task is done using traditional link-state updates that update all locations upon change in any location. Distributed global SDN measurements may extend the cap on the scale of physical clusters. Other data-center uses being listed are distributed application load balancing, distributed fire-walls, and similar adaptations to original networking functions that arise from dynamic, any location or rack allocation of compute resources.

Other uses of SDN in enterprise or carrier managed network services (MNS) address the traditional and geo-distributed campus network. These environments were always challenged by the complexities of additions, changes, mergers, acquisitions, and movement of users. Based on SDN principles, it is expected that these identity and policy management challenges could be addressed using global definitions decoupled from the physical interfaces of the network infrastructure. On the other hand, existing infrastructure of potentially thousands of switches and routers can remain intact.

It has been noted that this “overlay” approach raises a high likelihood of inefficiency and low performance by ignoring the characteristics of the underlying infrastructure. Hence, carriers have identified the gaps in overlays and asked for them to be filled by SDN solutions that take traffic, topology, and equipment into account.*[23]

Security using SDN paradigm

SDN architecture may enable, facilitate or enhance network-related security applications due to the controller’s central view of the network, and its capacity to reprogram the data plane at any time. While security of SDN architecture itself remains an open question that has already been studied a couple of times in the research community,*[24]*[25]*[26] the following paragraphs only focus on the security applications made possible or revisited using SDN.

Several research works on SDN have already investigated security applications built upon the SDN controller, with different aims in mind. Distributed Denial of Service (DDoS) detection and mitigation,*[27]*[28] as well as bot-net*[29] and worm propagation,*[30] are some concrete use-cases of such applications: basically, the idea consists in periodically collecting network statistics from the forwarding plane of the network in a standardized manner (e.g. using Openflow), and then apply classification algorithms on those statistics in order to detect any network anomalies. If an anomaly is detected, the application instructs the controller how to reprogram the data plane in order to mitigate it.

Another kind of security applications leverages the SDN controller by implementing some moving target defense (MTD) algorithms. MTD algorithms are typically used to make any attack on a given system or network more difficult than usual by periodically hiding or changing key properties of that system or network. In traditional networks, implementing MTD algorithms is not a trivial task since it is difficult to build a central authority able of determining - for each part of the system to be protected - which key properties are hid or changed. In an SDN network, such tasks become more straightforward thanks to the centrality of the controller. One application can for example periodically assign virtual IPs to hosts within the network, and the mapping virtual IP/real IP is then performed by the controller.*[31] Another application can simulate some fake opened/closed/filtered ports on random hosts in the network in order to add significant noise during reconnaissance phase (e.g. scanning) performed by an attacker.*[32]

Additional value regarding security in SDN enabled networks can also be gained using FlowVisor*[33] and FlowChecker*[34] respectively. The former tries to use a single hardware forwarding plane sharing multiple separated logical networks. Following this approach the same hardware resources can be used for production and development purposes as well as separating monitoring, configuration and internet traffic, where each scenario can have its own logical topology which is called slice. In conjunction with this approach FlowChecker*[33] realizes the validation of new OpenFlow rules that are deployed by users using their own slice.

Developing applications for software defined networks requires comprehensive checks of possible programming errors. Since SDN controller applications are mostly deployed in large scale scenarios a programming model checking solution requires scalability. These functionalities are provided among others through NICE*[35]

16.8 Access control

Remote access to the control plane is made available to administrators or users of the network, typically with a role-based access control (RBAC) in order to provide security.

16.9 See also

- Active Networking
- Frenetic (programming language)
- IEEE 802.1aq
- Intel Data Plane Development Kit (DPDK)
- Network Functions Virtualization
- OpenFlow
- OpenStack
- Software-defined data center
- OpenDaylight Project
- Software-defined Protection
- List of SDN controller software

16.10 References

- [1] “Software-Defined Networking: The New Norm for Networks” . *White paper*. Open Networking Foundation. April 13, 2012. Retrieved August 22, 2013.
- [2] Sun Pegs Telecom with JTONE
- [3] Sun facilitates Java use for public network operators
- [4] “CERIAS : GeoPlex: Universal Service Platform for IP Network-based Services - 10/17/1997” . *Cerias.purdue.edu*. Retrieved 26 October 2014.
- [5] “Middleware Networks” . *Dl.acm.org*. Retrieved 26 October 2014.
- [6] “Design Automation of Supranet Systems: Benefits for Hardware Design and Bringup” (PDF). *S3us-west-2.amazonaws.com*. Retrieved 22 November 2014.
- [7] “Websprocket Announces VMServer - World's First Proxy Java Virtual Machine; Enables 1,000's of Connected Clients To Use Single Java Virtual Machine.” . *Thefreelibrary.com*. Retrieved 26 October 2014.
- [8] “Installation Guide” . *Web.archive.org*. Retrieved 22 November 2014.
- [9] “Top Emerging Technologies Announced During Gartner Symposium/ITxpo 2000; New Emerging Technologies Research Highlights Trends in Wearable Computing, Profiling and Privacy.” . *Thefreelibrary.com*. Retrieved 26 October 2014.
- [10] “Websprocket Selected By Supranet Consortium to Enable the Internet With Smart Packet Technology; Platform Unifies Supranet Management Through Java and Oracle.” . *Thefreelibrary.com*. Retrieved 26 October 2014.
- [11] “Software Defined Network SDN” . *Scribd.com*. Retrieved 26 October 2014.
- [12] “PERFORMANCE MEASUREMENT AND ANALYSIS OF H.323 VIDEOCONFERENCE TRAFFIC” (PDF). *Adec.edu*. Retrieved 22 November 2014.
- [13] “United States Patent: 8122128” . *Patft.uspto.gov*. Retrieved 26 October 2014.
- [14] “United States Patent: 8799468” . *Patft.uspto.gov*. Retrieved 26 October 2014.
- [15] “Prof. Scott Shenker - Gentle Introduction to Software-Defined Networking - Technion lecture” . YouTube. 2012-06-26. Retrieved 2014-01-23.
- [16] “Interop 2014: Avaya to showcase Automated Campus part of SDN initiative” . Info Tech Lead. 26 March 2014.
- [17] “Avaya Software Defined Data Center” . Tech Field Day. Feb 2014. Retrieved 25 June 2014.

- [18] “Software-Defined Networking (SDN) Definition” . *Opennetworking.org*. Retrieved 26 October 2014.
- [19] “White Papers” . *Opennetworking.org*. Retrieved 26 October 2014.
- [20] “SDN Architecture Overview” (PDF). *Opennetworking.org*. Retrieved 22 November 2014.
- [21] “OpenFlow: Proactive vs Reactive” . *NetworkStatic.net*. Retrieved 2014-07-01.
- [22] “Reactive, Proactive, Predictive: SDN Models | F5 DevCentral” . *Devcentral.f5.com*. Retrieved 2014-01-23.
- [23] “Google's software-defined/OpenFlow backbone drives WAN links to 100% utilization” . *Networkworld.com*. 2012-06-07. Retrieved 2014-01-23.
- [24] Kreutz, Diego and Ramos, Fernando and Verissimo, Paulo (2013). “Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking” . pp. 50–60. lchapter= ignored (help)
- [25] Scott-Hayward, Sandra and O'Callaghan, Gemma and Sezer, Sakir (2013). “Future Networks and Services (SDN4FNS), 2013 IEEE SDN for” . pp. 1–7. lchapter= ignored (help)
- [26] Benton, Kevin and Camp, L Jean and Small, Chris (2013). “Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking” . pp. 151–152. lchapter= ignored (help)
- [27] Giotis, K and Argyropoulos, Christos and Androulidakis, Georgios and Kalogeras, Dimitrios and Maglaris, Vasilis (2014). “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments” . *Computer Networks* **62**: 122–136.
- [28] Braga, Rodrigo and Mota, Edjard and Passito, Alexandre (2010). “Local Computer Networks (LCN), 2010 IEEE 35th Conference on” . pp. 408–415. lchapter= ignored (help)
- [29] Feamster, Nick (2010). “Proceedings of the 2010 ACM SIGCOMM workshop on Home networks” . pp. 37–42. lchapter= ignored (help)
- [30] Jin, Ruofan and Wang, Bing (2013). “Research and Educational Experiment Workshop (GREE), 2013 Second GENI” . 81–88. lchapter= ignored (help)
- [31] Jafarian, Jafar Haadi and Al-Shaer, Ehab and Duan, Qi (2012). “Proceedings of the first workshop on Hot topics in software defined networks” . pp. 127–132. lchapter= ignored (help)
- [32] Kampanakis, Panos and Perros, Harry and Beyene, Tsegereda. “SDN-based solutions for Moving Target Defense network protection” . Retrieved 23 July 2014.
- [33] Sherwood, Rob and Gibb, Glen and Yap, Kok-Kiong and Appenzeller, Guido and Casado, Martin and McKeown, Nick and Parulkar, Guru (2009). “Flowvisor: A network virtualization layer” . *OpenFlow Switch Consortium, Tech. Rep.*
- [34] Al-Shaer, Ehab and Al-Haj, Saeed (2010). “Proceedings of the 3rd ACM workshop on Assurable and usable security configuration” . pp. 37–44. lchapter= ignored (help)
- [35] Canini, Marco and Venzano, Daniele and Peresini, Peter and Kostic, Dejan and Rexford, Jennifer and others (2012). “A NICE Way to Test OpenFlow Applications” . NSDI. pp. 127–140.

16.11 External links

- [Open Networking Foundation's definition of SDN](#)
- [Prof. Nick Feamster's Coursera Course on SDN](#)
- [OpenFlow-enabled SDN and Network Functions Virtualization](#)
- [SDN Security Considerations in the Data Center](#)
- [Floodlight, an open source Java based OpenFlow controller](#)
- [Network Function Virtualization \(NFV\)](#)
- [Decoding SDN](#)
- [Software-defined networking \(SDN\) for the non-technical](#)
- [Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure – a report from the ATIS Technology and Operations Council](#)

Chapter 17

Network Functions Virtualization

Network Functions Virtualization (NFV) is a network architecture concept that proposes using IT virtualization related technologies to virtualize entire classes of network node functions into building blocks that may be connected, or chained, to create communication services.

NFV relies upon, but differs from, traditional server virtualization techniques such as those used in enterprise IT. A virtualized network function, or VNF, may consist of one or more virtual machines running different software and processes, on top of industry standard high volume servers, switches and storage, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function.

For example, a virtualized session border controller function could be deployed to protect a network without the typical cost and complexity of obtaining and installing physical units. Other examples of NFV include virtualized load balancers, firewalls, intrusion detection devices and WAN accelerators. [1]

17.1 Background

Product development within the telecommunications industry has traditionally followed rigorous standards for stability, protocol adherence and quality. While this model worked well in the past, it inevitably led to long product cycles, a slow pace of development and reliance on proprietary or specialist hardware. The rise of significant competition in communications services, from fast-moving organizations operating at large scale on the public Internet (such as Google Talk), have spurred service providers to look for ways to disrupt the status quo.

17.2 History

In October 2012, an industry specifications group, “Network Functions Virtualisation”, [2] published a white paper at a conference in Darmstadt, Germany on software-defined networking and OpenFlow. [3] The group, part of the European Telecommunications Standards Institute (ETSI), was made up of representatives from the telecommunications industry from both Europe and beyond. [4] [5]

Since the publication of the white paper, the group has produced several more in-depth materials, including a standard terminology definition [6] and use cases for NFV that act as references for vendors and operators considering implementing NFV.

17.3 NFV Framework

The NFV framework consists of three main components: [7]

1. **Virtualized Network Functions (VNF)** are software implementations of network functions that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

2. **NFV Infrastructure (NFVI)** is the totality of all hardware and software components which build up the environment in which VNFs are deployed. The NFV-Infrastructure can span across several locations. The network providing connectivity between these locations is regarded to be part of the NFV-Infrastructure.
3. **Network Functions Virtualization Management and Orchestration Architectural Framework (NFV-MANO Architectural Framework)** is the collection of all functional blocks, data repositories used by these functional blocks, and reference points and interfaces through which these functional blocks exchange information for the purpose of managing and orchestrating NFVI and VNFs.

The building block for both the NFVI and the NFV-MANO is the NFV platform. In the NFVI role, it consists of both virtual and physical compute and storage resources, and virtualization software. In its NFV-MANO role it consists of VNF and NFVI managers and virtualization software operating on a hardware controller. The NFV platform implements carrier-grade features used to manage and monitor the platform components, recover from failures and provide effective security - all required for the public carrier network.

17.4 Practical aspects

A service provider who follows the NFV design will implement one or more virtualized network functions, or *VNFs*. A VNF by itself does not automatically provide a usable product or service to the provider's customers. To build more complex services, the notion of *service chaining* is used, where multiple VNFs are used in sequence to deliver a service.

Another aspect of implementing NFV is the *orchestration* process. In order to build highly reliable and scalable services, NFV requires that the network be able to instantiate VNF instances, monitor them, repair them, and (most importantly for a service provider business) bill for the services rendered. These attributes, referred to as Carrier-Grade* [8] features, are allocated to an orchestration layer in order to achieve high availability and security, and low operations and maintenance costs. Importantly, the orchestration layer must be able to manage VNFs irrespective of what the underlying technology within the VNF is. For example, an orchestration layer must be able to manage an SBC VNF from vendor X running on VMware vSphere just as well as an IMS VNF from vendor Y running on KVM (Kernel-based Virtual Machine).

17.5 Distributed NFV

The initial perception of NFV was that virtualized capability should be implemented in data centers. This approach works in many – but not all – cases. NFV presumes and emphasizes the widest possible flexibility as to the physical location of the virtualized functions.

Ideally, therefore, virtualized functions should be located where they will be the most effective and least expensive. That means a service provider should be free to locate NFV in all possible locations, from the data center to the network node to the customer premises. This approach, known as Distributed NFV, has been emphasized from the beginning as NFV was being developed and standardized, and is prominent in the recently released NFV ISG documents.* [9]

For some cases there are clear advantages for a service provider to locate this virtualized functionality at the customer premises. These advantages range from economics to performance to the feasibility of the functions being virtualized.* [10]

17.6 NFV modularity benefits

When designing and developing the software that provides the VNFs, vendors may structure that software into software components (implementation view of a software architecture) and package those components into one or more images (deployment view of a software architecture). These vendor-defined software components are called VNF Components (VNFCs). VNFs are implemented with one or more VNFCs and it is assumed, without loss of generality, that VNFCs map 1:1 to VM Images. Scale Up and scale Out enablement: By being able to allocate flexible (virtual) CPU to each of the VNFC instance, the network management layer can scale UP & down VNFC resource to follow up with the throughput/performance and scalability expectations over a single system or a single platform.

Similarly, the network management layer can scale OUT & down the VNFC instances over multiple platforms and therefore reach out to the performance and architecture specifications whilst not compromising the other VNFC function stabilities.

Early adopters of such architecture blueprints have already implemented the NFV modularity principles.* [11]

17.7 Relationship to SDN

SDN, or **software-defined networking**, is a concept related to NFV, but they refer to different domains.

In essence, Software-defined networking (SDN) is an approach to building data networking equipment and software that separates and abstracts elements of these systems. It does this by decoupling the control plane and data plane from each other, such that the control plane resides centrally and the forwarding components remain distributed. The control plane interacts both northbound and southbound. In the northbound direction the control plane provides a common abstracted view of the network to higher-level applications and programs using APIs. In the southbound direction the control plane programs the forwarding behavior, using device level APIs, of the physical network equipment distributed around the network.

Thus, NFV is not dependent on SDN or SDN concepts. It is entirely possible to implement a virtualized network function (VNF) as a standalone entity using existing networking and orchestration paradigms. However, there are inherent benefits in leveraging SDN concepts to implement and manage an NFV infrastructure, particularly when looking at the management and orchestration of VNFs, and that's why multivendor platforms are being defined that incorporate SDN and NFV in concerted ecosystems.* [12]

An NFV infrastructure is more than simply adapting existing network applications to run on x86 technology. It needs a central orchestration and management system that takes operator requests associated with a VNF, translates them into the appropriate compute, storage and network configuration needed to bring the VNF into operation. Once in operation, the VNF potentially needs to be monitored for capacity and utilization and adapted if necessary.

All these functions can be accomplished using SDN concepts and NFV could be considered one of the primary SDN use cases in service provider environments. It is also apparent that many SDN use-cases could incorporate concepts introduced in the NFV initiative. Examples include where the centralized controller is controlling a distributed forwarding function that could in fact be also visualized on existing compute or routing equipment.

17.8 Industry impact

NFV has proven a popular standard even in its infancy. Its immediate applications are numerous, such as virtualization of Mobile base stations, Platform as a Service, Content Delivery Networks (CDN), fixed access and home environments.* [13] The potential benefits of NFV is anticipated to be significant. Virtualization of network functions deployed on general purpose standardized hardware is expected to reduce capital and operational expenditures, and service and product introduction times.* [14]* [15] Many major network equipment vendors have announced support for NFV.* [16] This has coincided with NFV announcements from major software suppliers who provide the NFV platforms used by equipment suppliers to build their NFV products.* [17]* [18]

However, to realize the expected benefits, the network equipment vendors need to improve IT virtualization technology to incorporate carrier-grade attributes required by Cloud Service Providers (CSPs). They include fault, performance, security, operations and administration management capabilities.* [8]

17.9 MANO - Management and Orchestration

ETSI has already indicated that an important part of controlling the NFV environment should be done through automation and orchestration. There is a separate stream MANO within NFV outlining how flexibility should be controlled. [Mano at network-functions-virtualization.com](http://network-functions-virtualization.com)

17.10 See also

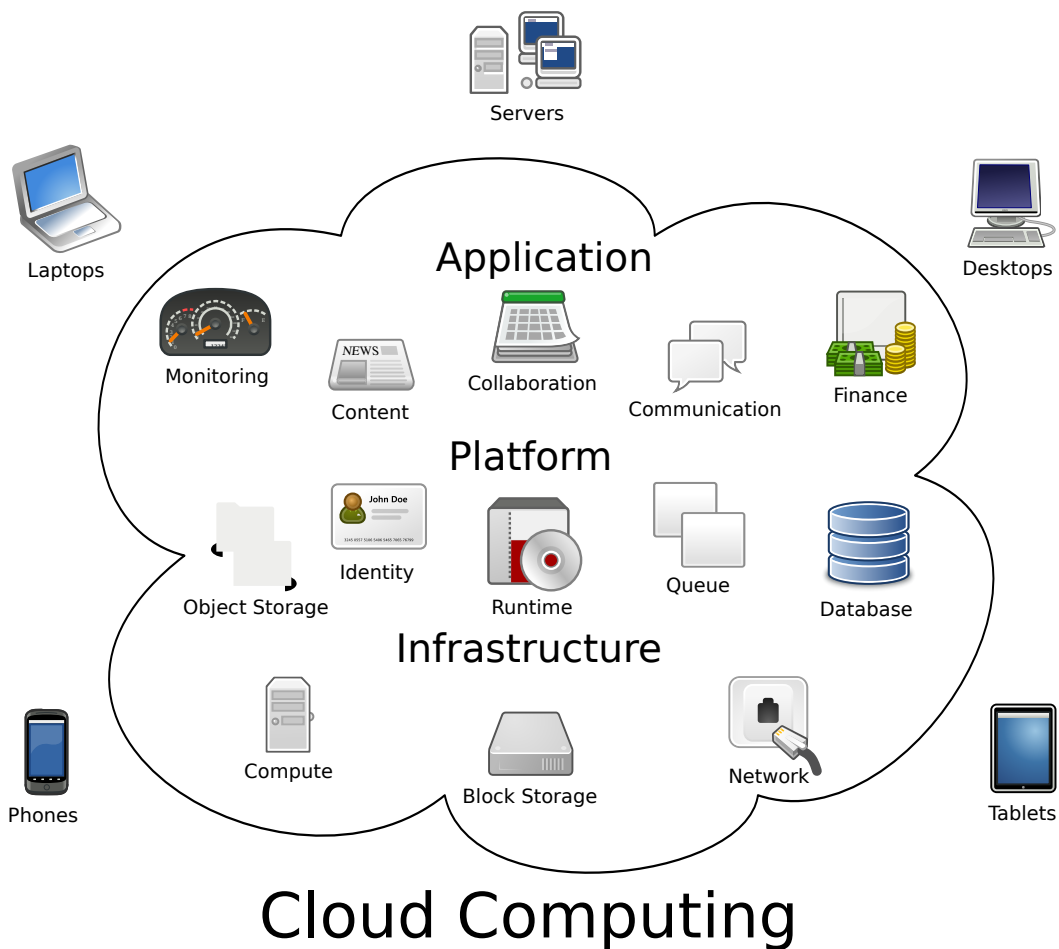
- Hardware virtualization
- Software-defined networking
- Network virtualization
- Network management
- Shortest Path Bridging

17.11 References

- [1] “Network Functions Virtualisation (NFV); Use Cases” . Retrieved 6 June 2014.
- [2] “Network Functions Virtualisation” . *ISG web portal*. Retrieved 20 June 2013.
- [3] “Network Functions Virtualisation—Introductory White Paper” . ETSI. 22 October 2012. Retrieved 20 June 2013.
- [4] Ray Le Maistre (22 October 2012). “Tier 1 Carriers Tackle Telco SDN” . *Light Reading*. Retrieved 20 June 2013.
- [5] “Latest Agenda at SDN & OpenFlow World Congress” . Layer123.com. Archived from the original on October 14, 2012. Retrieved 20 June 2013.
- [6] Mulligan, Ultan. “ETSI Publishes First Specifications for Network Functions Virtualisation” . Retrieved 5 December 2013.
- [7] Network Functions Virtualization (NFV) Proofs of Concept; Framework, GS NFV-PER 002 v1.1.1 (2013-10),
- [8] Don’ t Confuse ‘High Availability’ with Carrier Grade, Embedded Community, Charlie Ashton, April, 2014
- [9] Tom Nolle (18 September 2013). “Is “Distributed NFV” Teaching Us Something?”. *CIMI Corporation’s Public Blog*. Retrieved 2 January 2014.
- [10] Carol Wilson (3 October 2013). “RAD Rolls Out Distributed NFV Strategy” . *Light Reading*. Retrieved 2 January 2014.
- [11] TMCnet News (26 June 2014). “Qosmos Awarded a 2014 INTERNET TELEPHONY NFV Pioneer Award” . *TMC*. Retrieved 26 June 2014.
- [12] Platform to Multivendor Virtual and Physical Infrastructure
- [13] Network Functions Virtualization (NFV) Use Cases, ETSI GS NFV 001 v1.1.1 (2013-10)
- [14] What’ s NFV – Network Functions Virtualization?, SDN Central
- [15] Carrier Network Virtualization, ETSI news
- [16] “Openwave Exec Discusses the Benefits, Challenges of NFV & SDN” . *Article*. 12 November 2013. Retrieved 22 November 2013.
- [17] Middleware for the NFV Generation, Service, Lee Doyle
- [18] Wind River Launches NFV Ecosystem Program with Five Industry Leaders, PCC Mobile Broadband, Ray Sharma

Chapter 18

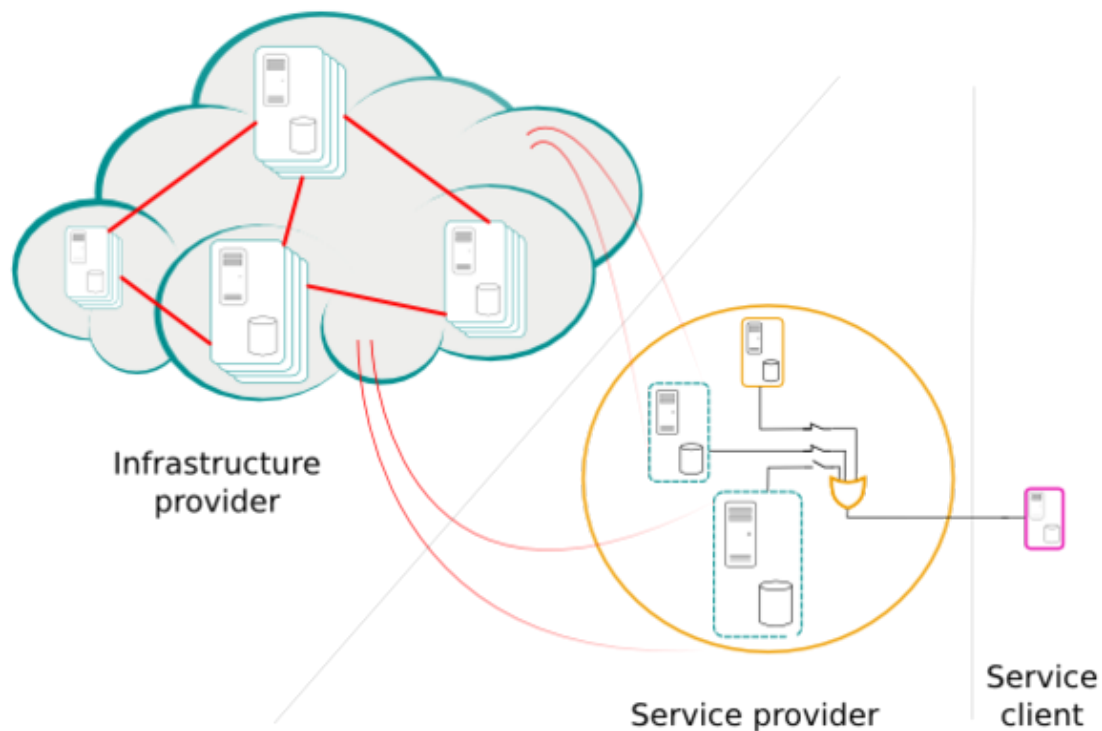
Cloud computing



Cloud Computing

Cloud computing metaphor: For a user, the network elements representing the provider-rendered services are invisible, as if obscured by a cloud.

Cloud computing is a recently evolved computing terminology or metaphor based on utility and consumption of computing resources. Cloud computing involves deploying groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources. Clouds can be classified as public, private or hybrid.* [1]* [2]



Another representation of the cloud computing model. Here, the emphasis is on the delivery of a service.

18.1 Overview

Cloud computing^{*} [3] relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network.^{*} [2] At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

Cloud computing, or in simpler shorthand just “the cloud”, also focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users. For example, a cloud computer facility that serves European users during European business hours with a specific application (e.g., email) may reallocate the same resources to serve North American users during North America's business hours with a different application (e.g., a web server). This approach should maximize the use of computing power thus reducing environmental damage as well since less power, air conditioning, rack space, etc. are required for a variety of functions. With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different applications.

The term “moving to cloud” also refers to an organization moving away from a traditional CAPEX model (buy the dedicated hardware and depreciate it over a period of time) to the OPEX model (use a shared cloud infrastructure and pay as one uses it).

Proponents claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure.^{*} [4] Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.^{*} [4]^{*} [5]^{*} [6] Cloud providers typically use a “pay as you go” model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model.^{*} [7]

The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing.^{*} [8]^{*} [9]^{*} [10]

Cloud vendors are experiencing growth rates of 50% per annum.^{*} [11]

18.2 History

18.2.1 Origin of the term



The origin of the term *cloud computing* is unclear. The expression *cloud* is commonly used in science to describe a large agglomeration of objects that visually appear from a distance as a cloud and describes any set of things whose details are not inspected further in a given context.*[12]

In analogy to above usage the word *cloud* was used as a metaphor for the Internet and a standardized cloud-like shape was used to denote a network on telephony schematics and later to depict the Internet in **computer network diagrams**. With this simplification, the implication is that the specifics of how the end points of a network are connected are not relevant for the purposes of understanding the diagram. The cloud symbol was used to represent the Internet as early as 1994,*[13]*[14] in which servers were then shown connected to, but external to, the cloud.

References to cloud computing in its modern sense appeared early as 1996, with the earliest known mention in a Compaq internal document.*[15]

The popularization of the term can be traced to 2006 when Amazon.com introduced the Elastic Compute Cloud.*[16]

18.2.2 The 1950s

The underlying concept of cloud computing dates to the 1950s, when large-scale **mainframe computers** were seen as the future of computing, and became available in academia and corporations, accessible via thin clients/**terminal computers**, often referred to as "**dumb terminals**", because they were used for communications but had no internal processing capacities. To make more efficient use of costly mainframes, a practice evolved that allowed multiple users to share both the physical access to the computer from multiple terminals as well as the CPU time. This eliminated periods of inactivity on the mainframe and allowed for a greater return on the investment. The practice of sharing CPU time on a mainframe became known in the industry as **time-sharing**.*[17] During the mid 70s, time-sharing was popularly known as RJE (Remote Job Entry); this nomenclature was mostly associated with large vendors such as IBM and DEC.

18.2.3 The 1990s

In the 1990s, telecommunications companies, who previously offered primarily dedicated point-to-point data circuits, began offering **virtual private network (VPN)** services with comparable quality of service, but at a lower cost. By switching traffic as they saw fit to balance server use, they could use overall network bandwidth more effectively. They began to use the cloud symbol to denote the demarcation point between what the provider was responsible for and what users were responsible for. Cloud computing extends this boundary to cover all servers as well as the network infrastructure.* [18]

As computers became more prevalent, scientists and technologists explored ways to make large-scale computing power available to more users through time-sharing. They experimented with algorithms to optimize the infrastructure, platform, and applications to prioritize CPUs and increase efficiency for end users.* [19]

18.2.4 Since 2000

In early 2008, **Eucalyptus** became the first open-source, AWS API-compatible platform for deploying private clouds. In early 2008, **OpenNebula**, enhanced in the RESERVOIR European Commission-funded project, became the first open-source software for deploying private and hybrid clouds, and for the federation of clouds.* [20] In the same year, efforts were focused on providing **quality of service** guarantees (as required by real-time interactive applications) to cloud-based infrastructures, in the framework of the IRMOS European Commission-funded project, resulting in a real-time cloud environment.* [21] By mid-2008, Gartner saw an opportunity for cloud computing “to shape the relationship among consumers of IT services, those who use IT services and those who sell them” * [22] and observed that “organizations are switching from company-owned hardware and software assets to per-use service-based models” so that the “projected shift to computing ... will result in dramatic growth in IT products in some areas and significant reductions in other areas.” * [23]

In July 2010, **Rackspace Hosting** and **NASA** jointly launched an open-source cloud-software initiative known as **OpenStack**. The OpenStack project intended to help organizations offer cloud-computing services running on standard hardware. The early code came from NASA's **Nebula platform** as well as from **Rackspace's Cloud Files platform**.* [24]

On March 1, 2011, IBM announced the **IBM SmartCloud** framework to support **Smarter Planet**.* [25] Among the various components of the **Smarter Computing** foundation, cloud computing is a critical piece.

On June 7, 2012, Oracle announced the **Oracle Cloud**.* [26] While aspects of the Oracle Cloud are still in development, this cloud offering is posed to be the first to provide users with access to an integrated set of IT solutions, including the Applications (SaaS), Platform (PaaS), and Infrastructure (IaaS) layers.* [27]* [28]* [29]

18.3 Similar concepts

Cloud computing is the result of evolution and adoption of existing technologies and paradigms. The goal of cloud computing is to allow users to take benefit from all of these technologies, without the need for deep knowledge about or expertise with each one of them. The cloud aims to cut costs, and helps the users focus on their core business instead of being impeded by IT obstacles.* [30]

The main enabling technology for cloud computing is **virtualization**. Virtualization software separates a physical computing device into one or more “virtual” devices, each of which can be easily used and managed to perform computing tasks. With **operating system-level virtualization** essentially creating a scalable system of multiple independent computing devices, idle computing resources can be allocated and used more efficiently. Virtualization provides the agility required to speed up IT operations, and reduces cost by increasing infrastructure utilization. Autonomic computing automates the process through which the user can provision resources **on-demand**. By minimizing user involvement, automation speeds up the process, reduces labor costs and reduces the possibility of human errors.* [30]

Users routinely face difficult business problems. Cloud computing adopts concepts from **Service-oriented Architecture (SOA)** that can help the user break these problems into **services** that can be integrated to provide a solution. Cloud computing provides all of its resources as services, and makes use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way.

Cloud computing also leverages concepts from utility computing to provide metrics for the services used. Such

metrics are at the core of the public cloud pay-per-use models. In addition, measured services are an essential part of the feedback loop in autonomic computing, allowing services to scale on-demand and to perform automatic failure recovery.

Cloud computing is a kind of **grid computing**; it has evolved by addressing the QoS (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional **parallel computing** techniques. <ref name=HAM2012/

Cloud computing shares characteristics with:

- **Client-server model** — *Client-server computing* refers broadly to any **distributed application** that distinguishes between service providers (servers) and service requestors (clients). * [31]
- **Grid computing** — “A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a **cluster of networked, loosely coupled computers** acting in concert to perform very large tasks.”
- **Mainframe computer** — Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as: **census**; industry and consumer statistics; police and secret intelligence services; **enterprise resource planning**; and **financial transaction processing**.
- **Utility computing** — The “**packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity.**” * [32] * [33]
- **Peer-to-peer** — A distributed architecture without the need for central coordination. Participants are both suppliers and consumers of resources (in contrast to the traditional client-server model).

18.4 Characteristics

Cloud computing exhibits the following key characteristics:

- **Agility** improves with users' ability to re-provision technological infrastructure resources.
- **Application programming interface** (API) accessibility to software that enables machines to interact with cloud software in the same way that a traditional user interface (e.g., a computer desktop) facilitates interaction between humans and computers. Cloud computing systems typically use Representational State Transfer (REST)-based APIs.
- **Cost** reductions claimed by cloud providers. A public-cloud delivery model converts capital expenditure to **operational expenditure**. * [34] This purportedly lowers **barriers to entry**, as infrastructure is typically provided by a third party and does not need to be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is fine-grained, with usage-based options and fewer IT skills are required for implementation (in-house). * [35] The e-FISCAL project's state-of-the-art repository * [36] contains several articles looking into cost aspects in more detail, most of them concluding that costs savings depend on the type of activities supported and the type of infrastructure available in-house.
- **Device and location independence** * [37] enable users to access systems using a web browser regardless of their location or what device they use (e.g., PC, mobile phone). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect from anywhere. * [35]
- **Maintenance** of cloud computing applications is easier, because they do not need to be installed on each user's computer and can be accessed from different places.
- **Multitenancy** enables sharing of resources and costs across a large pool of users thus allowing for:
 - **centralization** of infrastructure in locations with lower costs (such as real estate, electricity, etc.)
 - **peak-load capacity** increases (users need not engineer for highest possible load-levels)
 - **utilisation and efficiency** improvements for systems that are often only 10–20% utilised. * [38] * [39]
- **Performance** is monitored, and consistent and loosely coupled architectures are constructed using **web services** as the system interface. * [35] * [40] * [41]

- **Productivity** may be increased when multiple users can work on the same data simultaneously, rather than waiting for it to be saved and emailed. Time may be saved as information does not need to be re-entered when fields are matched, nor do users need to install application software upgrades to their computer. * [42]
- **Reliability** improves with the use of multiple redundant sites, which makes well-designed cloud computing suitable for business continuity and disaster recovery. * [43]
- **Scalability and elasticity** via dynamic (“on-demand”) provisioning of resources on a fine-grained, self-service basis in near real-time * [44] * [45] (Note, the VM startup time varies by VM type, location, OS and cloud providers * [44]), without users having to engineer for peak loads. * [46] * [47] * [48]
- **Security** can improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. * [49] Security is often as good as or better than other traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford to tackle. * [50] However, the complexity of security is greatly increased when data is distributed over a wider area or over a greater number of devices, as well as in multi-tenant systems shared by unrelated users. In addition, user access to security **audit logs** may be difficult or impossible. Private cloud installations are in part motivated by users' desire to retain control over the infrastructure and avoid losing control of information security.

The National Institute of Standards and Technology's definition of cloud computing identifies “five essential characteristics”:

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

—National Institute of Standards and Technology * [2]

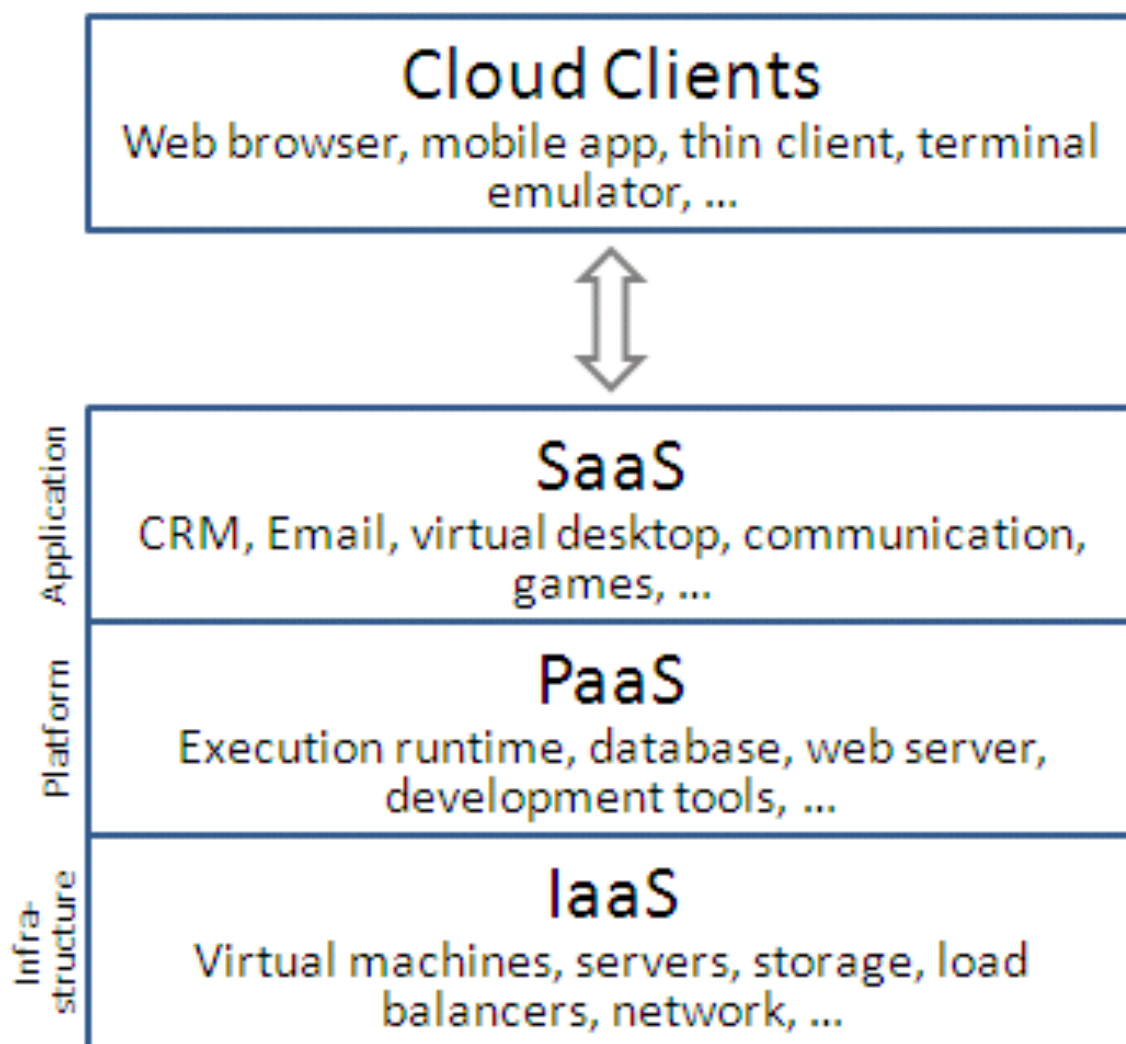
18.5 Service models

Cloud computing providers offer their services according to several fundamental models: * [2] * [51]

18.5.1 Infrastructure as a service (IaaS)

See also: Category:Cloud infrastructure and Software-defined application delivery

In the most basic cloud-service model & according to the IETF (Internet Engineering Task Force), providers of IaaS offer computers – physical or (more often) virtual machines – and other resources. (A hypervisor, such as Xen, Oracle VirtualBox, KVM, VMware ESX/ESXi, or Hyper-V runs the virtual machines as guests. Pools of hypervisors within the cloud operational support-system can support large numbers of virtual machines and the ability to scale services up and down according to customers' varying requirements.) IaaS clouds often offer additional resources such as a virtual-machine disk image library, raw block storage, and file or object storage, firewalls, load balancers,



IP addresses, virtual local area networks (VLANs), and software bundles.*[52] IaaS-cloud providers supply these resources on-demand from their large pools installed in data centers. For wide-area connectivity, customers can use either the Internet or carrier clouds (dedicated virtual private networks).

To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software. Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed.*[53]*[54]*[55]

18.5.2 Platform as a service (PaaS)

Main article: Platform as a service
See also: Category:Cloud platforms

In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. With some PaaS offers like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually. The latter has also been proposed by an architecture aiming to facilitate real-time in cloud environments.*[56]

18.5.3 Software as a service (SaaS)

Main article: [Software as a service](#)

In the **business model** using software as a service (SaaS), users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as “on-demand software” and is usually priced on a pay-per-use basis or using a subscription fee.

In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and support. Cloud applications are different from other applications in their scalability—which can be achieved by cloning tasks onto multiple **virtual machines** at run-time to meet changing work demand.*[57] **Load balancers** distribute the work over the set of virtual machines. This process is transparent to the cloud user, who sees only a single access point. To accommodate a large number of cloud users, cloud applications can be *multitenant*, that is, any machine serves more than one cloud user organization.

The pricing model for SaaS applications is typically a monthly or yearly flat fee per user,* [58] so price is scalable and adjustable if users are added or removed at any point.*[59]

Proponents claim SaaS allows a business the potential to reduce IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and personnel expenses, towards meeting other goals. In addition, with applications hosted centrally, updates can be released without the need for users to install new software. One drawback of SaaS is that the users' data are stored on the cloud provider's server. As a result, there could be unauthorized access to the data. For this reason, users are increasingly adopting intelligent third-party key management systems to help secure their data.

18.5.4 Unified Communications as a Service

Main article: [Unified communications](#)

In the UCaaS model, multi-platform communications over the network are packaged by the service provider. The services could be in different devices, such as computers and mobile devices. Services may include IP telephony, unified messaging, video conferencing and mobile extension*[60] etc.

18.6 Cloud clients

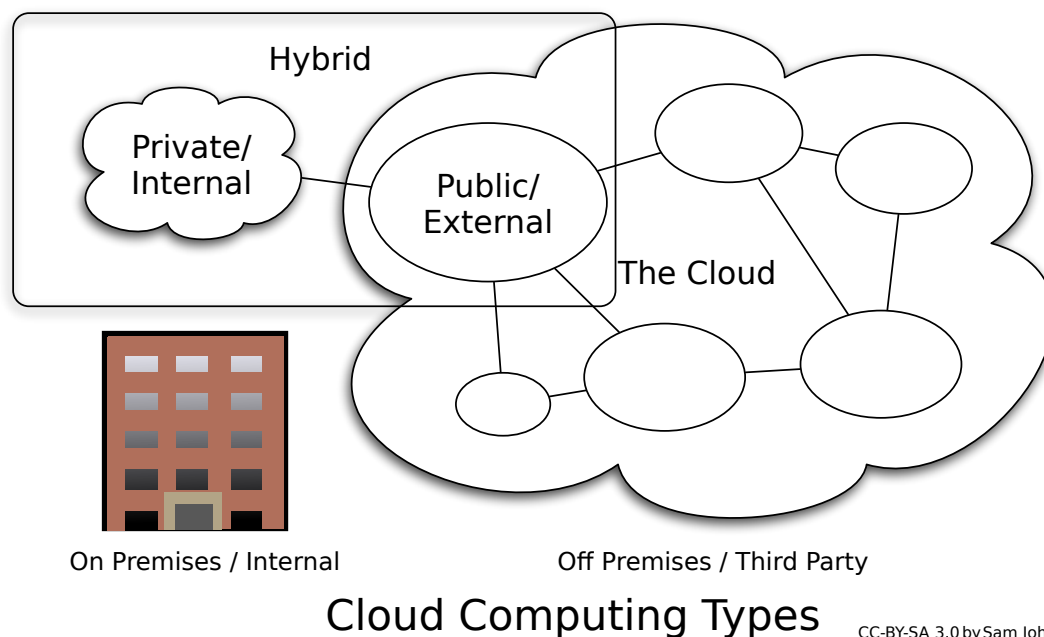
See also: [Category:Cloud clients](#)

Users access cloud computing using networked client devices, such as **desktop computers**, **laptops**, **tablets** and **smartphones**. Some of these devices – *cloud clients* – rely on cloud computing for all or a majority of their applications so as to be essentially useless without it. Examples are **thin clients** and the browser-based **Chromebook**. Many cloud applications do not require specific software on the client and instead use a web browser to interact with the cloud application. With **Ajax** and **HTML5** these **Web user interfaces** can achieve a similar, or even better, **look and feel** to native applications. Some cloud applications, however, support specific client software dedicated to these applications (e.g., **virtual desktop** clients and most email clients). Some legacy applications (line of business applications that until now have been prevalent in thin client computing) are delivered via a screen-sharing technology.

18.7 Deployment models

18.7.1 Private cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally.* [2] Undertaking a private cloud project requires a significant level and



Cloud computing types

degree of engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. When done right, it can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities.*[61] Self-run data centers*[62] are generally capital intensive. They have a significant physical footprint, requiring allocations of space, hardware, and environmental controls. These assets have to be refreshed periodically, resulting in additional capital expenditures. They have attracted criticism because users “still have to buy, build, and manage them” and thus do not benefit from less hands-on management,*[63] essentially “[lacking] the economic model that makes cloud computing such an intriguing concept” .*[64]*[65]

18.7.2 Public cloud

A cloud is called a “public cloud” when the services are rendered over a network that is open for public use. Public cloud services may be free or offered on a pay-per-usage model.*[66] Technically there may be little or no difference between public and private cloud architecture, however, security consideration may be substantially different for services (applications, storage, and other resources) that are made available by a service provider for a public audience and when communication is effected over a non-trusted network. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure at their data center and access is generally via the Internet. AWS and Microsoft also offer direct connect services called “AWS Direct Connect” and “Azure ExpressRoute” respectively, such connections require customers to purchase or lease a private connection to a peering point offered by the cloud provider.*[35]

18.7.3 Hybrid cloud

Hybrid cloud is a composition of two or more clouds (private, community or public) that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources.*[2]

Gartner, Inc. defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers.*[67] A hybrid cloud service crosses isolation and provider boundaries so that it can’t be simply put in one category of private, public, or community cloud service. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.

Varied use cases for hybrid cloud composition exist. For example, an organization may store sensitive client data in house on a private cloud application, but interconnect that application to a business intelligence application provided on a public cloud as a software service.*[68] This example of hybrid cloud extends the capabilities of the enterprise to deliver a specific business service through the addition of externally available public cloud services.

Another example of hybrid cloud is one where IT organizations use public cloud computing resources to meet temporary capacity needs that can not be met by the private cloud.*[69] This capability enables hybrid clouds to employ cloud bursting for scaling across clouds.*[2] Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and “bursts” to a public cloud when the demand for computing capacity increases. A primary advantage of cloud bursting and a hybrid cloud model is that an organization only pays for extra compute resources when they are needed.*[70] Cloud bursting enables data centers to create an in-house IT infrastructure that supports average workloads, and use cloud resources from public or private clouds, during spikes in processing demands.*[71]

18.7.4 Others

Community cloud

Community cloud shares infrastructure between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party, and either hosted internally or externally. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized.*[2]

Distributed cloud

Cloud computing can also be provided by a distributed set of machines that are running at different locations, while still connected to a single network or hub service. Examples of this include distributed computing platforms such as BOINC and Folding@Home. An interesting attempt in such direction is Cloud@Home, aiming at implementing cloud computing provisioning model on top of voluntarily shared resources * [72]

Intercloud

Main article: [Intercloud](#)

The Intercloud*[73] is an interconnected global “cloud of clouds”*[74]*[75] and an extension of the Internet “network of networks” on which it is based. The focus is on direct interoperability between public cloud service providers, more so than between providers and consumers (as is the case for hybrid- and multi-cloud).*[76]*[77]*[78]

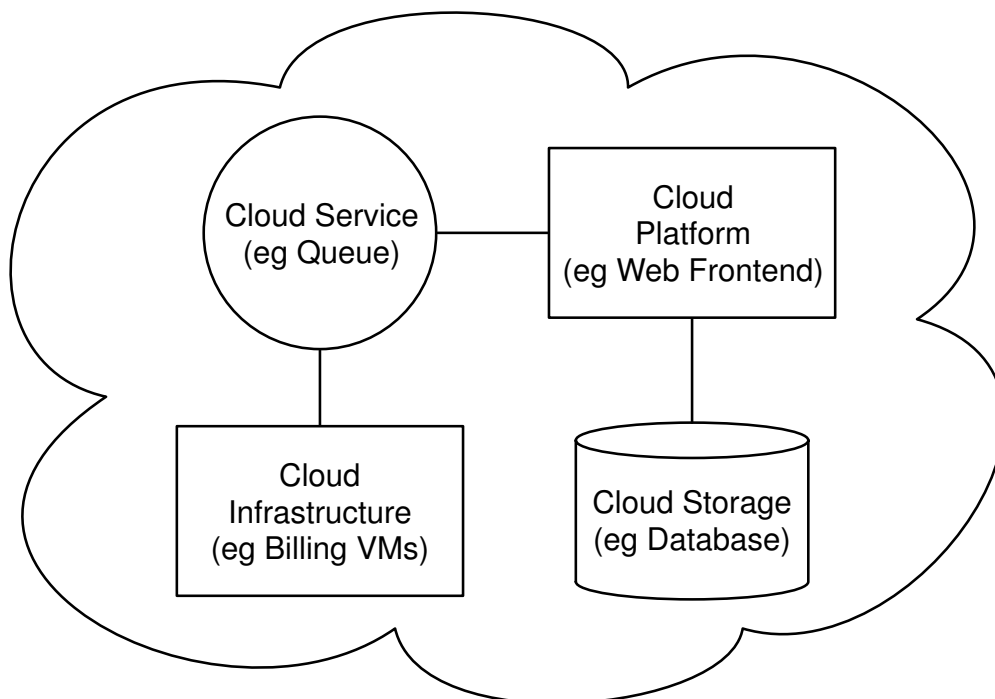
Multicloud

Main article: [Multicloud](#)

Multicloud is the use of multiple cloud computing services in a single heterogeneous architecture to reduce reliance on single vendors, increase flexibility through choice, mitigate against disasters, etc. It differs from hybrid cloud in that it refers to multiple cloud services, rather than multiple deployment modes (public, private, legacy).*[79]*[80]

18.8 Architecture

Cloud architecture,*[81] the systems architecture of the software systems involved in the delivery of cloud computing, typically involves multiple *cloud components* communicating with each other over a loose coupling mechanism such as a messaging queue. Elastic provision implies intelligence in the use of tight or loose coupling as applied to mechanisms such as these and others.



Cloud computing sample architecture

18.8.1 Cloud engineering

Cloud engineering is the application of engineering disciplines to cloud computing. It brings a systematic approach to the high-level concerns of commercialization, standardization, and governance in conceiving, developing, operating and maintaining cloud computing systems. It is a multidisciplinary method encompassing contributions from diverse areas such as systems, software, web, performance, information, security, platform, risk, and quality engineering.

18.9 Security and privacy

Main article: [Cloud computing issues](#)

Cloud computing poses privacy concerns because the service provider can access the data that is on the cloud at any time. It could accidentally or deliberately alter or even delete information.*[82] Many cloud providers can share information with third parties if necessary for purposes of law and order even without a warrant. That is permitted in their privacy policies which users have to agree to before they start using cloud services.*[83] Solutions to privacy include policy and legislation as well as end users' choices for how data is stored.*[84] Users can encrypt data that is processed or stored within the cloud to prevent unauthorized access.*[84]

There is the problem of legal ownership of the data (If a user stores some data in the cloud, can the cloud provider profit from it?). Many Terms of Service agreements are silent on the question of ownership.*[85]

Physical control of the computer equipment (private cloud) is more secure than having the equipment off site and under someone else's control (public cloud). This delivers great incentive to public cloud computing service providers to prioritize building and maintaining strong management of secure services.*[86] Some small businesses that don't have expertise in IT security could find that it's more secure for them to use a public cloud.

There is the risk that end users don't understand the issues involved when signing on to a cloud service (persons sometimes don't read the many pages of the terms of service agreement, and just click "Accept" without reading). This is important now that cloud computing is becoming popular and required for some services to work, for example for an intelligent personal assistant (Apple's Siri or Google Now).

Fundamentally private cloud is seen as more secure with higher levels of control for the owner, however public cloud is seen to be more flexible and requires less time and money investment from the user. * [87]

18.10 Cloud Marketplaces

Cloud providers and other third-party brokers offer to end-users and developers cloud solutions through online marketplace environments. * [88] These e-commerce sites have as primary objective to simplify the consumers' access to the various cloud products and services and optimize and facilitate their management across the cloud stack layers. The cloud marketplaces operate either as part of the cloud provider's ecosystem or as independent business entities which sell cloud-based solutions using resources and services from public clouds. * [89] Through the cloud marketplaces, software, technologies and cloud resources are tailored together and merchandised as bundles, exploiting techniques for automated instantiation, deployment and management, and allowing for centralized monitoring, accounting and billing. * [90] These bundles include application servers, databases, software libraries and virtual machines but practically any type of software and technology can be attached using orchestration and configuration tools such as Chef, Puppet and Juju.

Cloud marketplaces also incorporate the typical features of the modern online marketplaces for rating and feedback on the various offerings from the consumers, settlement, payments and analytics, which are adapted for cloud-based software and service products. Additionally, cloud marketplaces may also provide mechanisms for SLA Negotiation and Management, * [91] analysis of application requirements and dependencies, * [92] trust management, * [93] flexible pricing (e.g. spot instances) * [94] and cost calculation. In that sense, the consumers can focus on the application features and its business aspects, using the marketplace as the single point of access for the various technologies and cloud resources required for the delivery of the application.

18.11 Other services that require it

Other services require the use of Cloud Computing:

- Intelligent personal assistant : The “intelligent” part of it is usually done in the cloud for being computationally expensive (For example for Apple's Siri * [95] * [96]).

18.12 The future

According to Gartner's Hype cycle, cloud computing has reached a maturity that leads it into a productive phase. This means that most of the main issues with cloud computing have been addressed to a degree that clouds have become interesting for full commercial exploitation. This however does not mean that all the problems listed above have actually been solved, only that the according risks can be tolerated to a certain degree. * [97] Cloud computing is therefore still as much a research topic, as it is a market offering. * [98] What is clear through the evolution of Cloud Computing services is that the CTO is a major driving force behind Cloud adoption. * [99] The major Cloud technology developers continue to invest billions a year in Cloud R&D, in 2011 Microsoft for example committed 90% of its \$9.6bn R&D budget to Cloud * [100]

18.13 See also

- Category:Cloud computing providers
- Category:Cloud platforms
- Cloud computing comparison
- Cloud management
- Cloud research
- Cloud storage

- Edge computing
- Fog computing
- Grid computing
- iCloud
- Mobile cloud computing
- Personal cloud
- Synaptop
- Ubiquitous computing
- Web computing

18.14 References

- [1] Hassan, Qusay (2011). “Demystifying Cloud Computing” . *The Journal of Defense Software Engineering (CrossTalk)* **2011** (Jan/Feb): 16–21. Retrieved 11 December 2014.
- [2] “The NIST Definition of Cloud Computing” . National Institute of Standards and Technology. Retrieved 24 July 2011.
- [3] “Know Why Cloud Computing Technology is the New Revolution” . By Fonebell. Retrieved 8 January 2015.
- [4] “What is Cloud Computing?”. *Amazon Web Services*. 2013-03-19. Retrieved 2013-03-20.
- [5] “Baburajan, Rajani, “The Rising Cloud Storage Market Opportunity Strengthens Vendors,” infoTECH, August 24, 2011” . It.tmcnet.com. 2011-08-24. Retrieved 2011-12-02.
- [6] Oestreich, Ken, (2010-11-15). “Converged Infrastructure” . *CTO Forum*. Thectoforum.com. Retrieved 2011-12-02.
- [7] “Where’s The Rub: Cloud Computing’s Hidden Costs” . 2014-02-27. Retrieved 2014-07-14.
- [8] “Cloud Computing: Clash of the clouds” . *The Economist*. 2009-10-15. Retrieved 2009-11-03.
- [9] “Gartner Says Cloud Computing Will Be As Influential As E-business” . Gartner. Retrieved 2010-08-22.
- [10] Gruman, Galen (2008-04-07). “What cloud computing really means” . *InfoWorld*. Retrieved 2009-06-02.
- [11] “The economy is flat so why are financials Cloud vendors growing at more than 90 percent per annum?”. FSN. March 5, 2013.
- [12] Liu, [edited by] Hongji Yang, Xiaodong (2012). “9” . *Software reuse in the emerging cloud computing era*. Hershey, PA: Information Science Reference. pp. 204–227. ISBN 9781466608979. Retrieved 11 December 2014.
- [13] Figure 8, “A network 70 is shown schematically as a cloud” , US Patent 5,485,455, column 17, line 22, filed Jan 28, 1994
- [14] Figure 1, “the cloud indicated at 49 in Fig. 1.” , US Patent 5,790,548, column 5 line 56–57, filed April 18, 1996
- [15] Antonio Regalado (31 October 2011). “Who Coined 'Cloud Computing'?”. *Technology Review (MIT)*. Retrieved 31 July 2013.
- [16] “Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta” . Amazon.com. 2006-08-24. Retrieved 2014-05-31.
- [17] Strachey, Christopher (June 1959). “Time Sharing in Large Fast Computers” . *Proceedings of the International Conference on Information processing, UNESCO*. paper B.2.19: 336–341.
- [18] “July, 1993 meeting report from the IP over ATM working group of the IETF” . CH: Switch. Retrieved 2010-08-22.
- [19] Corbató, Fernando J. “An Experimental Time-Sharing System” . *SJCC Proceedings*. MIT. Retrieved 3 July 2012.
- [20] Rochwerger, B.; Breitgand, D.; Levy, E.; Galis, A.; Nagin, K.; Llorente, I. M.; Montero, R.; Wolfsthal, Y.; Elmroth, E.; Caceres, J.; Ben-Yehuda, M.; Emmerich, W.; Galan, F. “The Reservoir model and architecture for open federated cloud computing” . *IBM Journal of Research and Development* **53** (4): 4:1–4:11. doi:10.1147/JRD.2009.5429058.

- [21] Kyriazis, D; A Menychtas; G Kousiouris; K Oberle; T Voith; M Boniface; E Oliveros; T Cucinotta; S Berger (November 2010). “A Real-time Service Oriented Infrastructure” . *International Conference on Real-Time and Embedded Systems (RTES 2010)* (Singapore).
- [22] Keep an eye on cloud computing, Amy Schurr, Network World, 2008-07-08, citing the Gartner report, “Cloud Computing Confusion Leads to Opportunity” . Retrieved 2009-09-11.
- [23] Gartner (2008-08-18). “Gartner Says Worldwide IT Spending On Pace to Surpass Trillion in 2008” .
- [24] “OpenStack History” .
- [25] “Launch of IBM Smarter Computing” . Retrieved 1 March 2011.
- [26] “Launch of Oracle Cloud” . Retrieved 28 February 2014.
- [27] “Oracle Cloud, Enterprise-Grade Cloud Solutions: SaaS, PaaS, and IaaS” . Retrieved 12 October 2014.
- [28] “Larry Ellison Doesn’t Get the Cloud: The Dumbest Idea of 2013” . Forbes.com. Retrieved 12 October 2014.
- [29] “Oracle Disrupts Cloud Industry with End-to-End Approach” . Forbes.com. Retrieved 12 October 2014.
- [30] HAMDAQA, Mohammad (2012). *Cloud Computing Uncovered: A Research Landscape*. Elsevier Press. pp. 41–85. ISBN 0-12-396535-7.
- [31] “Distributed Application Architecture” . Sun Microsystem. Retrieved 2009-06-16.
- [32] “It’s probable that you’ve misunderstood ‘Cloud Computing’ until now” . TechPluto. Retrieved 2010-09-14.
- [33] Danielson, Krissi (2008-03-26). “Distinguishing Cloud Computing from Utility Computing” . Ebizq.net. Retrieved 2010-08-22.
- [34] “Recession Is Good For Cloud Computing – Microsoft Agrees” . CloudAve. Retrieved 2010-08-22.
- [35] “Defining ‘Cloud Services’ and “Cloud Computing””. IDC. 2008-09-23. Retrieved 2010-08-22.
- [36] “e-FISCAL project state of the art repository” .
- [37] Farber, Dan (2008-06-25). “The new geek chic: Data centers” . CNET News. Retrieved 2010-08-22.
- [38] “Jeff Bezos’ Risky Bet” . *Business Week*.
- [39] He, Sijin; L. Guo, Y. Guo, M. Ghanem,. “Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models” . 2012 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 574–581. doi:10.1109/CLOUD.2012.66. ISBN 978-1-4673-2892-0.
- [40] He, Qiang, et al. “Formulating Cost-Effective Monitoring Strategies for Service-based Systems.” (2013): 1-1.
- [41] A Self-adaptive hierarchical monitoring mechanism for Clouds Elsevier.com
- [42] Heather Smith (23 May 2013). *Xero For Dummies*. John Wiley & Sons. pp. 37–. ISBN 978-1-118-57252-8.
- [43] King, Rachael (2008-08-04). “Cloud Computing: Small Companies Take Flight” . *Bloomberg BusinessWeek*. Retrieved 2010-08-22.
- [44] Mao, Ming; M. Humphrey (2012). “A Performance Study on the VM Startup Time in the Cloud” . *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing (Cloud2012)*: 423. doi:10.1109/CLOUD.2012.103. ISBN 978-1-4673-2892-0.
- [45] Dario Bruneo, Salvatore Distefano, Francesco Longo, Antonio Puliafito, Marco Scarpa: Workload-Based Software Rejuvenation in Cloud Systems. *IEEE Trans. Computers* 62(6): 1072-1085 (2013)
- [46] “Defining and Measuring Cloud Elasticity” . KIT Software Quality Departement. Retrieved 13 August 2011.
- [47] “Economies of Cloud Scale Infrastructure” . Cloud Slam 2011. Retrieved 13 May 2011.
- [48] He, Sijin; L. Guo; Y. Guo; C. Wu; M. Ghanem; R. Han. “Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning” . 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), pp. 15–22. doi:10.1109/AINA.2012.74. ISBN 978-1-4673-0714-7.
- [49] “Encrypted Storage and Key Management for the cloud” . Cryptoclarity.com. 2009-07-30. Retrieved 2010-08-22.
- [50] Mills, Elinor (2009-01-27). “Cloud computing security forecast: Clear skies” . CNET News. Retrieved 2010-08-22.

- [51] Voorsluys, William; Broberg, James; Buyya, Rajkumar (February 2011). "Introduction to Cloud Computing" . In R. Buyya, J. Broberg, A.Goscinski. *Cloud Computing: Principles and Paradigms*. New York, USA: Wiley Press. pp. 1–44. ISBN 978-0-470-88799-8.
- [52] Amies, Alex; Sluiman, Harm; Tong, Qiang Guo; Liu, Guo Ning (July 2012). "*Infrastructure as a Service Cloud Concepts. Developing and Hosting Applications on the Cloud*". IBM Press. ISBN 978-0-13-306684-5.
- [53] "Amazon EC2 Pricing" . Retrieved 7 July 2014.
- [54] "Compute Engine Pricing" . Retrieved 7 July 2014.
- [55] "Microsoft Azure Virtual Machines Pricing Details" . Retrieved 7 July 2014.
- [56] Boniface, M. et al. (2010), *Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds*, 5th International Conference on Internet and Web Applications and Services (ICIW), Barcelona, Spain: IEEE, pp. 155–160, doi:10.1109/ICIW.2010.91
- [57] Hamdaqa, Mohammad. *A Reference Model for Developing Cloud Applications*.
- [58] Chou, Timothy. *Introduction to Cloud Computing: Business & Technology*.
- [59] "HVD: the cloud's silver lining" . Intrinsic Technology. Retrieved 30 August 2012.
- [60] http://www.csc.com/workplace_services/offerings/87346/87570-ucaas_unified_communications_as_a_service
- [61] "Is The Private Cloud Really More Secure?". CloudAndCompute.com. Retrieved 12 October 2014.
- [62] "Self-Run Private Cloud Computing Solution - GovConnection" . *govconnection.com*. 2014. Retrieved April 15, 2014.
- [63] Foley, John. "Private Clouds Take Shape" . *InformationWeek*. Retrieved 2010-08-22.
- [64] Haff, Gordon (2009-01-27). "Just don't call them private clouds" . CNET News. Retrieved 2010-08-22.
- [65] "There's No Such Thing As A Private Cloud" . *InformationWeek*. 2010-06-30. Retrieved 2010-08-22.
- [66] Rouse, Margaret. "What is public cloud?". Definition from Whatis.com. Retrieved 12 October 2014.
- [67] http://blogs.gartner.com/thomas_bittman/2012/09/24/mind-the-gap-here-comes-hybrid-cloud/
- [68] "Business Intelligence Takes to Cloud for Small Businesses" . CIO.com. 2014-06-04. Retrieved 2014-06-04.
- [69] Metzler, Jim; Taylor, Steve. (2010-08-23) "Cloud computing: Reality vs. fiction," Network World.
- [70] Rouse, Margaret. "Definition: Cloudbursting," May 2011. SearchCloudComputing.com.
- [71] Vizard, Michael. "How Cloudbursting 'Rightsizes' the Data Center" , (2012-06-21). Slashdot.
- [72] Vincenzo D. Cunsolo, Salvatore Distefano, Antonio Puliafito, Marco Scarpa: Volunteer Computing and Desktop Cloud: The Cloud@Home Paradigm. IEEE International Symposium on Network Computing and Applications, NCA 2009, pp 134-139
- [73] Bernstein, David; Ludvigson, Erik; Sankar, Krishna; Diamond, Steve; Morrow, Monique (2009-05-24). "Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability" . IEEE Computer Society. pp. 328–336. doi:10.1109/ICIW.2009.55. ISBN 978-1-4244-3851-8. |chapter= ignored (help)
- [74] "Kevin Kelly: A Cloudbook for the Cloud" . Kk.org. Retrieved 2010-08-22.
- [75] "Intercloud is a global cloud of clouds" . Samj.net. 2009-06-22. Retrieved 2010-08-22.
- [76] "Vint Cerf: Despite Its Age, The Internet is Still Filled with Problems" . Readwriteweb.com. Retrieved 2010-08-22.
- [77] "SP360: Service Provider: From India to Intercloud" . Blogs.cisco.com. Retrieved 2010-08-22.
- [78] Canada (2007-11-29). "Head iaaan the clouds? Welcome to the future" . *The Globe and Mail* (Toronto). Retrieved 2010-08-22.
- [79] Rouse, Margaret. "What is a multi-cloud strategy" . SearchCloudApplications. Retrieved 3 July 2014.
- [80] King, Rachel. "Pivotal's head of products: We're moving to a multi-cloud world" . ZDnet. Retrieved 3 July 2014.
- [81] "Building GrepTheWeb in the Cloud, Part 1: Cloud Architectures" . Developer.amazonwebservices.com. Retrieved 2010-08-22.

- [82] “Cloud Computing Privacy Concerns on Our Doorstep” .
- [83] “Sharing information without a warrant” . Retrieved 2014-12-05.
- [84]
- [85] Maltais, Michelle (26 April 2012). “Who owns your stuff in the cloud?”. *Los Angeles Times*. Retrieved 2012-12-14.
- [86] “Security of virtualization, cloud computing divides IT and security pros” . Network World. 2010-02-22. Retrieved 2010-08-22.
- [87] “The Bumpy Road to Private Clouds” . Retrieved 2014-10-08.
- [88] “Cloud Marketplace definition” . November 2013. Retrieved 29 December 2014.
- [89] *Cloud-Marketplace: New paradigm for e-marketplaces*, Proceedings of the tenth ACM international conference on Embedded software, ACM, 2010, pp. 1 – 8, doi:10.1145/1879021.1879022
- [90] *4CaaS marketplace: An advanced business environment for trading cloud services*, Future Generation Computer Systems **41**, Elsevier, 2014, pp. 104 – 120, doi:10.1016/j.future.2014.02.020
- [91] *Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets*, Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, ACM/IEEE, 2012, pp. 85 – 94, doi:10.1109/Grid.2012.18
- [92] *A Business Resolution Engine for Cloud Marketplaces*, Third International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2011, pp. 462 – 469, doi:10.1109/CloudCom.2011.68
- [93] *Towards a Trust Management System for Cloud Computing*, 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2011, pp. 933 – 939, doi:10.1109/DASC.2011.38
- [94] *Dynamic Resource Allocation for Spot Markets in Cloud Computing Environments*, Fourth IEEE International Conference on Utility and Cloud Computing (UCC), IEEE, 2011, pp. 178 – 185, doi:10.1109/UCC.2011.33
- [95] <http://www.petri.com/cloud-improving-siri-in-ios6.htm>
- [96] <http://www.macrumors.com/2013/04/19/anonymized-siri-voice-clips-stored-by-apple-for-up-to-two-years/>
- [97] <http://blog.kaseya.com/blog/2014/10/06/realistic-look-cloud-computing/>
- [98] Smith, David Mitchell. “Hype Cycle for Cloud Computing, 2013” . Gartner. Retrieved 3 July 2014.
- [99] <http://www.hello-cirro.co.uk/evolution-of-cloud-computing/>
- [100] <http://cloudtimes.org/2011/04/12/microsoft-says-to-spend-90-of-rd-on-cloud-strategy/>

18.15 External links

Chapter 19

Elasticity (cloud computing)

In **cloud computing**, **elasticity** is defined as the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. ^[1] It is a defining characteristic, that differentiates it from previously proposed computing paradigms, such as **grid computing**. This dynamic variation in the use of computer resources to meet a varying workload is called “elastic computing”. ^[2] ^[3] In general an elastic cloud application or process has three elasticity dimensions, ^[4] *Cost*, *Quality*, and *Resources*, enabling it to increase and decrease its cost, quality, or available resources, as to accommodate specific requirements.

19.1 Example

Let us illustrate elasticity through a simple example of a service provider who wants to run a **website** on an **IaaS** cloud. At moment t_0 , the website is unpopular and a single machine (most commonly a **virtual machine**) is sufficient to serve all web users. At moment t_1 , the website suddenly becomes popular, for example, as a result of a **flash crowd**, and a single machine is no longer sufficient to serve all users. Based on the amount of web users simultaneously accessing the website and the resource requirements of the **web server**, it might be that ten machines are needed. An elastic system should immediately detect this condition and provision nine additional machines from the cloud, so as to serve all web users responsively.

At time t_2 , the website becomes unpopular again. The ten machines that are currently allocated to the website are mostly idle and a single machine would be sufficient to serve the few users who are accessing the website. An elastic system should immediately detect this condition and deprovision nine machines and release them to the cloud.

19.2 Purpose

Elasticity aims at matching the amount of resources allocated to a service with the amount of resources it actually requires, avoiding over- or under-provisioning. **Over-provisioning**, i.e., allocating more resources than required, should be avoided as the service provider often has to pay for the resources that are allocated to the service. For example, at the time of this writing, **Amazon EC2** charges \$0.480/hour for an “extra large” virtual machine. If a service is allocated two virtual machines, instead of one required, the service provider wastes \$4,205 every year. Hence, the service provider's **expenses** are higher than optimal and the **profit** is reduced.

Under-provisioning, i.e., allocating fewer resources than required, must be avoided, otherwise the service cannot serve its users with a good service. In the above example, under-provisioning the website may make it seem slow or unreachable. Web users eventually give up on accessing it, thus, the service provider loses customers. On the long term, the provider's **income** will decrease, which also reduces the profit.

19.3 Problems

19.3.1 Resources provisioning time

One potential problem is that elasticity takes time. A cloud virtual machine (VM) can be acquired at any time by the user, however, it may take up to several minutes for the acquired VM to be ready to use. The VM startup time is dependent on factors, such as image size, VM type, data center location, number of VMs, etc.* [5] Cloud providers have different VM startup performance. This implies any control mechanism designed for elastic applications must consider in its decision process the time needed for the elasticity actions to take effect,* [6] such as provisioning another VM for a specific application component.

19.3.2 Monitoring elastic applications

Elastic applications can allocate and deallocate resources (such as VMs) on demand for specific application components. This makes cloud resources volatile, and traditional monitoring tools which associate monitoring data with a particular resource (i.e. VM), such as Ganglia or Nagios, are no longer suitable for monitoring the *behavior* of elastic applications. For example, during its lifetime, a data storage tier of an elastic application might add and remove data storage VMs due to cost and performance requirements, varying the number of used VMs. Thus, additional information is needed in monitoring elastic applications, such as associating the logical application structure over the underlying virtual infrastructure.* [7] This in turn generates other problems, such as how to aggregate data from multiple VMs towards extracting the behavior of the application component running on top of those VMs, as different metrics might need to be aggregated differently (e.g., cpu usage could be averaged, network transfer might be summed up).

19.3.3 Elasticity requirements

When deploying applications in cloud infrastructures (IaaS/PaaS), requirements of the stakeholder need to be considered in order to ensure proper elasticity behavior. Even though traditionally one would try to find the optimal trade-off between cost and quality or performance, for real world cloud users requirements regarding the behavior are more complex and target multiple dimensions of elasticity (e.g., SYBL* [8]).

19.3.4 Multiple levels of control

Cloud applications can be of varying types and complexities, with multiple levels of artifacts deployed in layers. Controlling such structures must take into consideration a variety of issues, an approach in this sense being rSYBL.* [9] For multi-level control, control systems need to consider the impact lower level control has upon higher level ones and vice versa (e.g., controlling virtual machines, web containers, or web services in the same time), as well as conflicts which may appear between various control strategies from various levels.* [10]

19.4 See also

- Amazon Elastic Compute Cloud
- CELAR: Cloud ELAsticity pRovisining

19.5 References

- [1] Herbst, Nikolas Roman; Samuel Kounev; Ralf Reussner (2012). “Elasticity in Cloud Computing: What It Is, and What It Is Not” . *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24–28*.
- [2] *Cloud Computing Principles and Paradigms*, John Wiley and Sons, 2011, ISBN 978-0-470-88799-8
- [3] Perez et al, *Responsive Elastic Computing*, ISBN 978-1-60558-578-9
- [4] Schahram, Dustdar; Yike Guo, Benjamin Satzger, Hong-Linh Truong (2011). “Principles of Elastic Processes” . *IEEE Internet Computing*, vol. 15, no. 5, pp. 66-71, September/October., doi:10.1109/MIC.2011.121.

- [5] Mao, Ming; M. Humphrey (2012). “A Performance Study on the VM Startup Time in the Cloud” . *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing (Cloud2012)*: 423. doi:10.1109/CLOUD.2012.103. ISBN 978-1-4673-2892-0.
- [6] Gambi, Alessio; Daniel Moldovan, Georgiana Copil,Hong-Linh Truong, Schahram Dustdar (2013). “On estimating actuation delays in elastic computing systems” . *Proceedings of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. doi:10.1109/SEAMS.2013.6595490.
- [7] Moldovan, Daniel; Georgiana Copil,Hong-Linh Truong, Schahram Dustdar (2013). “MELA: Monitoring and Analyzing Elasticity of Cloud Services” . *Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013)*. doi:10.1109/CloudCom.2013.18.
- [8] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, “SYBL: an Extensible Language for Controlling Elasticity in Cloud Applications” , *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 14–16, 2013, Delft, the Netherlands
- [9] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, “Specifying, Monitoring, and Controlling Elasticity of Cloud Services” , *Proceedings of the 11th International Conference on Service Oriented Computing*. Berlin, Germany, 2–5 December 2013. doi=10.1007/978-3-642-45005-1_31
- [10] Kranas, Pavlos (2012). “ElaaS: An Innovative Elasticity as a Service Framework for Dynamic Management across the Cloud Stack Layers” . *Proceedings of Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS) 4–6 July 2012 (IEEE)*. doi:10.1109/CISIS.2012.117.

19.6 External links

- The NIST Definition of Cloud Computing. Peter Mell and Timothy Grance, NIST Special Publication 800-145 (September 2011). National Institute of Standards and Technology, U.S. Department of Commerce.
- Buyya, Rajkumar; et al (June 2009). “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility” . *Future Generation Computer Systems* **25** (6).

Chapter 20

Platform as a service

Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run and manage Web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. [1][2][3] PaaS can be delivered in two ways: as a public cloud service from a provider, where the consumer controls software deployment and configuration settings, and the provider provides the networks, servers, storage and other services to host the consumer's application; or as software installed in private data centers or public infrastructure as a service and managed by internal IT departments. [4][5] The two primary programming languages for PaaS are Java and .NET, [1] according to Gartner. [6]

20.1 Development and uses

The idea of PaaS was pioneered on the Web by Amazon Web Services (AWS) and Salesforce.com. In April 2008, Google launched App Engine, with a free trial version limited to 10,000 developers. [7] This was said to have “turned the Internet cloud computing space into a fully-fledged industry virtually overnight.” [8]

The original intent of PaaS was to simplify the code-writing process for developers, with the infrastructure and operations handled by the PaaS provider. Originally, all PaaSes were in the public cloud. Because many companies did not want to have everything in the public cloud, private and hybrid PaaS options (managed by internal IT departments) were created. [9]

PaaS provides an environment for developers and companies to create, host and deploy applications, saving developers from the complexities of the infrastructure side (setting up, configuring and managing elements such as servers and databases). PaaS can improve the speed of developing an app, and allow the consumer to focus on the application itself. With PaaS, the consumer manages applications and data, while the provider (in public PaaS) or IT department (in private PaaS) manages runtime, middleware, operating system, virtualization, servers, storage and networking. [4][10] Development tools provided by the vendor are customized according to the needs of the user. [11] The user can choose to maintain the software, or have the vendor maintain it. [4]

PaaS offerings may also include facilities for application design, application development, testing and deployment, as well as services such as team collaboration, web service integration, and marshalling, database integration, security, scalability, storage, persistence, state management, application versioning, application instrumentation, and developer community facilitation. Besides the service engineering aspects, PaaS offerings include mechanisms for service management, such as monitoring, workflow management, discovery and reservation. [12][13]

20.2 Advantages and disadvantages

The advantages to PaaS are primarily that it allows for higher-level programming with dramatically reduced complexity; the overall development of the application can be more effective, as it has built-in infrastructure; and maintenance and enhancement of the application is easier. [3] It can also be useful in situations where multiple developers are working on a single project involving parties who are not located nearby. [2]

The primary disadvantage would be the possibility of being locked in to a certain platform. However, most PaaSes are relatively lock-in free. [8] Other possible disadvantages, according to *Forbes*, include the relative youth of the

cloud service model, the lack of support for .NET by many providers, and that the value and definition of PaaS has been at times misunderstood by those working in IT.*[9]

20.3 Types

20.3.1 Public, private and hybrid

There are several types of PaaS, including public, private and hybrid.*[9] PaaS was originally intended as an application solution in the public cloud, before expanding to include private and hybrid options.*[9]

Public PaaS is derived from *software as a service* (SaaS),*[7] and is situated in cloud computing between SaaS and *infrastructure as a service* (IaaS).*[1] SaaS is software that is hosted in the cloud, so that it doesn't take up hard drive or server space. IaaS provides virtual storage from a provider with adjustable scalability.*[1] With IaaS, the user still has to manage the server, whereas with PaaS the server management is done by the provider.*[14]

A private PaaS can typically be downloaded and installed either on a company's on-premises infrastructure, or in a public cloud. Once the software is installed on one or more machines, the private PaaS arranges the application and database components into a single hosting platform.*[15] Private PaaS vendors include *Apprenda*, which started out on the *Microsoft .NET* platform before rolling out a Java PaaS; *Red Hat's OpenShift* and *Pivotal Cloud Foundry*.*[16] *Apprenda* and *Microsoft* are considered to be two of the only PaaSes that provide superior .NET support.*[9]

20.3.2 Mobile PaaS

Initiated in 2012, mobile PaaS (mPaaS) provides development capabilities for mobile app designers and developers.*[17] The *Yankee Group* identified mPaaS as one of its themes for 2014, naming a number of providers including *Kinvey*, *CloudMine*, *AnyPresence*, *FeedHenry*, *FatFractal* and *Point.io*.*[18]*[19]

20.3.3 Open PaaS

Open PaaS does not include hosting, but rather it provides open source software allowing a PaaS provider to run applications in an open source environment. For example, *AppScale* allows a user to deploy some applications written for *Google App Engine* to their own servers, providing datastore access from a standard *SQL* or *NoSQL* database. Some open platforms let the developer use any programming language, database, operating system or server to deploy their applications.*[20]*[21]

20.3.4 PaaS for Rapid Development

In 2014, *Forrester Research* defined Enterprise Public Cloud Platforms for Rapid Developers as an emerging trend, naming a number of providers including *OutSystems*, *Mendix*, *Salesforce.com*, and *Acquia*.*[22]

20.3.5 System types

PaaS is found on the following types of systems:

Add-on development facilities These facilities customization of existing SaaS applications, often requiring PaaS developers and their users to purchase subscriptions to the add-on SaaS application.*[23]

Stand alone environments Stand-alone PaaS environments do not include technical, licensing or financial dependencies on specific SaaS applications or web services, and are intended to provide a generalized development environment.*[23]

Application delivery-only environments Delivery-only PaaS offerings generally focus on hosting services, such as security and on-demand scalability. The service does not include development, debugging and test capabilities, though they may be supplied offline (via an *Eclipse* plugin, for example).*[23]

20.4 Providers

There are various types of PaaS providers. All offer application hosting and a deployment environment, along with various integrated services. Services offer varying levels of scalability and maintenance.* [8] Developers can write an application and upload it to a PaaS that supports their software language of choice, and the application runs on that PaaS.* [24]

Public PaaS providers and private PaaS vendors include:

- [AppearIQ](#)
- [Apprenda](#)
- [AppScale](#)
- [AWS Elastic Beanstalk](#)
- [Cloud Foundry](#)
- [CloudControl](#)
- [Cloudera](#)
- [Engine Yard](#)
- [Google App Engine](#)
- [Heroku](#)
- [IBM Bluemix](#)
- [Microsoft Azure Web Sites](#)
- [Mendix](#)
- [Nodejitsu](#)
- [OpenShift](#)
- [OutSystems](#)
- [QlikView](#)
- [Salesforce](#)
- [WaveMaker](#)

20.5 See also

- [Network as a service](#)
- [Software as a service](#)
- [Infrastructure as a service](#)
- [Mobile Backend as a service](#)

20.6 References

- [1] Brandon Butler, “PaaS Primer: What is platform as a service and why does it matter?” *Network World*, February 11, 2013.
- [2] “Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS,” Rackspace, October 22, 2013.
- [3] William Y. Chang, Hosame Abu-Amara, Jessica Feng Sanford, *Transforming Enterprise Cloud Services*, London: Springer, 2010, pp. 55-56.
- [4] Judith Hurwitz, Marcia Kaufman, Fern Halper and Dan Kirsh, “What Is Platform as a Service (PaaS) in Cloud Computing?” *Hybrid Cloud For Dummies*, Hoboken, NJ: John Wiley & Sons, 2012.
- [5] “The NIST Definition of Cloud Computing” . National Institute of Science and Technology. Retrieved 24 July 2011.
- [6] Mark Driver, *Java and .NET: You Can't Pick a Favorite Child*, Gartner, p. 2.
- [7] Jack Schofield, “Google angles for business users with ‘platform as a service’ ,” *The Guardian*, April 16, 2008.
- [8] Dion Hinchcliffe, “Comparing Amazon’s and Google’s Platform-as-a-Service (PaaS) Offerings,” ZDNet, April 11, 2008.
- [9] Mike Kavis, “Top 8 Reasons Why Enterprises Are Passing On PaaS,” *Forbes*, September 15, 2014.
- [10] Sean Ludwig, “An ugly duckling no more: Why Platform-as-a-Service is poised for huge growth,” *VentureBeat*, October 8, 2012.
- [11] Andrea Peiro, “Keep Your Head in the Cloud,” *Inc.*, January 2009.
- [12] M. Boniface, “Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds,” *ieee.org*, May 2010.
- [13] Chen, Tse-Shih, et al. “Platform-as-a-Service Architecture for Parallel Video Analysis in Clouds.” *Advances in Intelligent Systems and Applications-Volume 2*. Springer Berlin Heidelberg, 2013. 619-626.
- [14] Andrew C. Oliver, “Which freaking PaaS should I use?” *InfoWorld*, October 8, 2012.
- [15] Jason Brooks, “Apprenda 3.0 Brings Private PaaS to .NET Developers,” *eWeek*, January 6, 2012.
- [16] Ben Kepes, “Apprenda Extends Its PaaS And Aims A Kick In The Direction of Red Hat,” *Forbes*, October 6, 2014.
- [17] Anthony Wing Kosner, “Developers in Demand: Platform As A Service Is Key to Growth of Mobile Cloud Computing,” *Forbes*, June 8, 2012.
- [18] Yankee 2014 Predictions Mobility hits a tipping point, Yankee Group, 2014.
- [19] Christina Warren, “How to Pick a Server for Your App,” *Mashable*, November 16, 2011.
- [20] “Interview With Brian Sullivan – Open Platform As A Service,” *openplatformasaservice.com*, 2010.
- [21] “The Top 20 Platform as a Service (PaaS) Vendors,” *clouds360.com*. Accessed January 23, 2015.
- [22] “Forrester Wave: Enterprise Public Cloud Platforms,” Q4 2014.
- [23] “Platform as a Service (PaaS),” *Quality Testing*, December 22, 2010.
- [24] Nancy Gohring, “Platform as a service heats up,” *Computerworld*, July 8, 2013.

Chapter 21

Software as a service

Software as a service (SaaS; pronounced /sæs/ or /sɑ:s/* [1]) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. * [2] * [3] It is sometimes referred to as “on-demand software”. * [4] SaaS is typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications, including office & messaging software, payroll processing software, DBMS software, management software, CAD software, development software, gamification, virtualization, * [4] accounting, collaboration, customer relationship management (CRM), management information systems (MIS), enterprise resource planning (ERP), invoicing, human resource management (HRM), content management (CM) and service desk management. * [5] SaaS has been incorporated into the strategy of all leading enterprise software companies. One of the biggest selling points for these companies is the potential to reduce IT support costs by outsourcing hardware and software maintenance and support to the SaaS provider. * [6]

According to a Gartner Group estimate, * [7] SaaS sales in 2010 reached \$10 billion, and were projected to increase to \$12.1bn in 2011, up 20.7% from 2010. Gartner Group estimates that SaaS revenue will be more than double its 2010 numbers by 2015 and reach a projected \$21.3bn. Customer relationship management (CRM) continues to be the largest market for SaaS. SaaS revenue within the CRM market was forecast to reach \$3.8bn in 2011, up from \$3.2bn in 2010. * [8]

The term “software as a service” (SaaS) is considered to be part of the nomenclature of cloud computing, along with infrastructure as a service (IaaS), platform as a service (PaaS), desktop as a service (DaaS), backend as a service (BaaS), and information technology management as a service (ITMaaS). * [9]

21.1 History

Centralized hosting of business applications dates back to the 1960s. Starting in that decade, IBM and other mainframe providers conducted a service bureau business, often referred to as time-sharing or utility computing. Such services included offering computing power and database storage to banks and other large organizations from their worldwide data centers.

The expansion of the Internet during the 1990s brought about a new class of centralized computing, called Application Service Providers (ASP). ASPs provided businesses with the service of hosting and managing specialized business applications, with the goal of reducing costs through central administration and through the solution provider's specialization in a particular business application. Two of the world's pioneers and largest ASPs were USI, which was headquartered in the Washington, D.C. area, and Futurelink Corporation, headquartered in Orange County California.

Software as a service essentially extends the idea of the ASP model. The term *Software as a Service (SaaS)*, however, is commonly used in more specific settings:

- Whereas most initial ASPs focused on managing and hosting third-party independent software vendors' software, as of 2012 SaaS vendors typically develop and manage their own software.
- Whereas many initial ASPs offered more traditional client-server applications, which require installation of software on users' personal computers, SaaS solutions of today rely predominantly on the Web and only require a web browser to use.

- Whereas the **software architecture** used by most initial ASPs mandated maintaining a separate instance of the application for each business, as of 2012 SaaS solutions normally utilize a **multi-tenant** architecture, in which the application serves multiple businesses and users, and partitions its data accordingly.

The SAAS acronym allegedly first appeared in an article called “Strategic Backgrounder: Software As A Service,” internally published in February 2001 by the Software & Information Industry Association’s (SIIA) eBusiness Division.* [10]

DbaaS (Database as a Service) has emerged as a sub-variety of SaaS.* [11]

21.2 Distribution

The Cloud (or SaaS) model has no physical need for indirect distribution since it is not distributed physically and is deployed almost instantaneously. The first wave of SaaS companies built their own economic model without including partner remuneration in their pricing structure (except when there were certain existing affiliations). It has not been easy for traditional software publishers to enter into the SaaS model. Firstly, because the SaaS model does not bring them the same income structure, secondly, because continuing to work with a distribution network was decreasing their profit margins and was damaging to the competitiveness of their product pricing. Today a landscape is taking shape with SaaS and managed service players who combine the indirect sales model with their own existing business model, and those who seek to redefine their role within the 3.0 IT economy.* [12]

21.3 Pricing

Unlike traditional software which is conventionally sold as a perpetual license with an up-front cost (and an optional ongoing support fee), SaaS providers generally price applications using a subscription fee, most commonly a monthly fee or an annual fee.* [13] Consequently, the initial setup cost for SaaS is typically lower than the equivalent enterprise software. SaaS vendors typically price their applications based on some usage parameters, such as the number of users using the application. However, because in a SaaS environment customers’ data reside with the SaaS vendor, opportunities also exist to charge per transaction, event, or other unit of value.* [14]

The relatively low cost for user provisioning (i.e., setting up a new customer) in a multi-tenant environment enables some SaaS vendors to offer applications using the freemium model.* [14] In this model, a free service is made available with limited functionality or scope, and fees are charged for enhanced functionality or larger scope.* [14] Some other SaaS applications are completely free to users, with revenue being derived from alternate sources such as advertising.

A key driver of SaaS growth is SaaS vendors’ ability to provide a price that is competitive with on-premises software. This is consistent with the traditional rationale for outsourcing IT systems, which involves applying **economies of scale** to application operation, i.e., an outside service provider may be able to offer better, cheaper, more reliable applications..

21.4 Architecture

The vast majority of SaaS solutions are based on a **multi-tenant** architecture. With this model, a single **version** of the application, with a single **configuration** (**hardware**, **network**, **operating system**), is used for all customers (“**tenants**”). To support scalability, the application is installed on multiple machines (called **horizontal scaling**). In some cases, a second version of the application is set up to offer a select group of customers with access to pre-release versions of the applications (e.g., a **beta version**) for **testing** purposes. This is contrasted with traditional software, where multiple physical copies of the software —each potentially of a different version, with a potentially different configuration, and often customized —are installed across various customer sites.

While an exception rather than the norm, some SaaS solutions do not use multi-tenancy, or use other mechanisms —such as **virtualization**—to cost-effectively manage a large number of customers in place of multi-tenancy.* [15] Whether multi-tenancy is a necessary component for software-as-a-service is a topic of controversy.* [16]

21.5 Characteristics

While not all software-as-a-service applications share all traits, the characteristics below are common among many SaaS applications:

21.5.1 Configuration and customization

SaaS applications similarly support what is traditionally known as application *customization*. In other words, like traditional enterprise software, a single customer can alter the set of configuration options (a.k.a., *parameters*) that affect its functionality and *look-and-feel*. Each customer may have its own settings (or: parameter values) for the configuration options. The application can be customized to the degree it was designed for based on a set of predefined configuration options.

For example: to support customers' common need to change an application's look-and-feel so that the application appears to be having the customer's *brand* (or—if so desired—*co-branded*), many SaaS applications let customers provide (through a *self service* interface or by working with application provider staff) a custom logo and sometimes a set of custom colors. The customer cannot, however, change the *page layout* unless such an option was designed for.

21.5.2 Accelerated feature delivery

SaaS applications are often updated more frequently than traditional software,* [17] in many cases on a weekly or monthly basis. This is enabled by several factors:

- *The application is hosted centrally, so an update is decided and executed by the provider, not by customers.*
- The application only has a single configuration, making development testing faster.
- The application vendor has access to all customer data, expediting design and regression testing.
- The solution provider has access to user behavior within the application (usually via *web analytics*), making it easier to identify areas worthy of improvement.

Accelerated feature delivery is further enabled by *agile software development* methodologies.* [18] Such methodologies, which have evolved in the mid-1990s, provide a set of *software development tools* and practices to support frequent software releases.

21.5.3 Open integration protocols

Since SaaS applications cannot access a company's internal systems (databases or internal services), they predominantly offer integration protocols and *application programming interfaces (APIs)* that operate over a *wide area network*. Typically, these are protocols based on *HTTP*, *REST*, *SOAP* and *JSON*.

The ubiquity of SaaS applications and other Internet services and the standardization of their API technology has spawned development of *mashups*, which are lightweight applications that combine data, presentation and functionality from multiple services, creating a compound service. Mashups further differentiate SaaS applications from on-premises software as the latter cannot be easily integrated outside a company's firewall.

21.5.4 Collaborative (and “social”) functionality

Inspired by the success of *online social networks* and other so-called *web 2.0* functionality, many SaaS applications offer features that let its users *collaborate* and *share information*.

For example, many *project management* applications delivered in the SaaS model offer—in addition to traditional project planning functionality—collaboration features letting users comment on tasks and plans and share documents within and outside an organization. Several other SaaS applications let users vote on and offer new feature ideas.

While some collaboration-related functionality is also integrated into on-premises software, (implicit or explicit) collaboration between users or different customers is only possible with centrally hosted software.

21.6 Adoption drivers

Several important changes to the software market and technology landscape have facilitated acceptance and growth of SaaS solutions:

- The growing use of web-based **user interfaces** by applications, along with the proliferation of associated practices (e.g., **web design**), continuously decreased the need for traditional client-server applications. Consequently, traditional software vendor's investment in software based on **fat clients** has become a disadvantage (mandating ongoing support), opening the door for new software vendors offering a **user experience** perceived as more "modern" .
- The standardization of web page technologies (**HTML**, **JavaScript**, **CSS**), the increasing popularity of **web development** as a practice, and the introduction and ubiquity of **web application frameworks** like **Ruby on Rails** or languages like **PHP** gradually reduced the cost of developing new SaaS solutions, and enabled new solution providers to come up with competitive solutions, challenging traditional vendors.
- The increasing penetration of **broadband Internet access** enabled remote centrally hosted applications to offer speed comparable to on-premises software.
- The standardization of the **HTTPS** protocol as part of the web stack provided universally available **lightweight security** that is sufficient for most everyday applications.
- The introduction and wide acceptance of **lightweight integration protocols** such as **REST** and **SOAP** enabled affordable integration between SaaS applications (residing in the cloud) with internal applications over wide area networks and with other SaaS applications.

21.7 Adoption challenges

Some limitations slow down the acceptance of SaaS and prohibit it from being used in some cases:

- Since data are being stored on the vendor' s servers, data security becomes an issue.*[19]
- **SaaS** applications are hosted in the cloud, far away from the application users. This introduces latency into the environment; so, for example, the SaaS model is not suitable for applications that demand response times in the milliseconds.
- Multi-tenant architectures, which drive cost efficiency for SaaS solution providers, limit customization of applications for large clients, inhibiting such applications from being used in scenarios (applicable mostly to large enterprises) for which such customization is necessary.
- Some business applications require access to or integration with customer's current data. When such data are large in volume or sensitive (e.g., end users' personal information), integrating them with remotely hosted software can be costly or risky, or can conflict with data governance regulations.
- Constitutional search/seizure warrant laws do not protect all forms of SaaS dynamically stored data. The end result is that a link is added to the chain of security where access to the data, and, by extension, misuse of these data, are limited only by the assumed honesty of 3rd parties or government agencies able to access the data on their own recognizance.*[20]*[21]*[22]*[23]
- Switching SaaS vendors may involve the slow and difficult task of transferring very large data files over the Internet.
- Organizations that adopt SaaS may find they are forced into adopting new versions, which might result in unforeseen training costs or an increase in probability that a user might make an error.
- Relying on an Internet connection means that data are transferred to and from a SaaS firm at Internet speeds, rather than the potentially higher speeds of a firm' s internal network.*[24]

The standard model also has limitations:

- Compatibility with hardware, other software, and operating systems.* [25]
- Licensing and compliance problems (unauthorized copies of the software program putting the organization at risk of fines or litigation).
- Maintenance, support, and patch revision processes.

21.8 Emerging trends

As a result of widespread fragmentation in the SaaS provider space,* [26] there is an emerging trend towards the development of SaaS Integration Platforms (SIP).* [27] These SIPs allow subscribers to access multiple SaaS applications through a common platform. They also offer new application developers an opportunity to quickly develop and deploy new applications. This trend is being referred to as the “third wave” in software adoption - where SaaS moves beyond standalone applications to become a comprehensive platform. Zoho and SutiSoft are two companies that offer comprehensive SIPs today. Several other industry players, including Salesforce, Microsoft, and Oracle are aggressively developing similar integration platforms. Another trend deals with the rise of software products that combine functions for human resource management, payroll accounting, and expense management as an all-in-one solution in promoting collaboration between an employer and an employee. This supplements the ongoing effort of many businesses to create Employee Self-Service tools for their workforce.

21.9 Data escrow

Software as a service data escrow is the process of keeping a copy of critical software-as-a-service application data with an independent third party. Similar to *source code escrow*, where critical software source code is stored with an independent third party, SaaS data escrow is the same logic applied to the data within a SaaS application. It allows companies to protect and insure all the data that resides within SaaS applications, protecting against data loss.* [28]

There are many and varied reasons for considering SaaS data escrow including concerns about vendor *bankruptcy*, unplanned service outages and potential *data loss* or corruption. Many businesses are also keen to ensure that they are complying with their own *data governance* standards or want improved reporting and *business analytics* against their SaaS data. A research conducted by Clearpace Software Ltd. into the growth of SaaS showed that 85 percent of the participants wanted to take a copy of their SaaS data. A third of these participants wanted a copy on a daily basis.* [29]

21.10 Criticism

One notable criticism of SaaS comes from Richard Stallman of the Free Software Foundation referring to it as Service as a Software Substitute (SaaS).* [30] He considers the use of SaaS to be a violation of the principles of *free software*.* [31] According to Stallman,

With SaaS, the users do not have a copy of the executable file: it is on the server, where the users can't see or touch it. Thus it is impossible for them to ascertain what it really does, and impossible to change it. SaaS inherently gives the server operator the power to change the software in use, or the users' data being operated on.

This criticism does not apply to all SaaS products. In 2010, *Forbes* contributor Dan Woods noted that Drupal Gardens, a free web hosting platform based on the open source Drupal content management system, is a “new open source model for SaaS”. He added, “Open source provides the escape hatch. In Drupal Gardens, users will be able to press a button and get a source code version of the Drupal code that runs their site along with the data from the database. Then, you can take that code, put it up at one of the hosting companies, and you can do anything that you would like to do.”* [32]

Andrew Hoppin, a former Chief Information Officer for the New York State Senate, refers to this combination of SaaS and open source software as OpenSaaS and points to WordPress as another successful example of an OpenSaaS software delivery model that gives customers “the best of both worlds, and more options. The fact that it is open

source means that they can start building their websites by self-hosting WordPress and customizing their website to their heart's content. Concurrently, the fact that WordPress is SaaS means that they don't have to manage the website at all -- they can simply pay WordPress.com to host it."*[33]

21.11 See also

- Servicizing
- Cloud-based integration
- Application service provider

21.12 References

- [1] Panker, Jon; Lewis, Mark; Fahey, Evan; Vasquez, Melvin Jafet (August 2007). "How do you pronounce IT?". TechTarget. Retrieved 24 May 2012.
- [2] Paul, Gil. "What Is 'SaaS' (Software as a Service)?" . *About.com*.
- [3] "Definition of:SaaS". *PC Magazine Encyclopedia*. Ziff Davis. Retrieved 14 May 2014.
- [4] "IT Channel Glossary" . compuBase. March 2013. Retrieved 13 February 2013.
- [5] "Software as a Service (SaaS)". *Cloud Taxonomy*. Retrieved 24 April 2011.
- [6] "SaaS Implementation" . simplilearn.com.
- [7] McHall, Tom (7 July 2011). "Gartner Says Worldwide Software as a Service Revenue Is Forecast to Grow 21 Percent in 2011" . *Gartner.com*. Gartner. Retrieved 28 July 2011.
- [8] Barret, Larry (27 July 2010). "SaaS Market Growing by Leaps and Bounds: Gartner" . *Datamation*. QuinStreet, Inc.
- [9] Anderson, Tim (5 May 2011). "full form of SaaS" . *The Register*.
- [10] "Software As A Service: Strategic Backgrounder" . Washington, D.C.: Software & Information Industry Association. 28 February 2001. Retrieved 24 April 2011.
- [11] Ferrari, Elena (2010). *Access Control in Data Management Systems*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers. p. 77. ISBN 978-1-60845-375-7. Retrieved 2012-02-13. [...] a new emerging option is represented by the Database as a Service (DbaaS) paradigm [...]. DbaaS is regulated by the same principles as Software as a Service (SaaS) and basically means the delivery of the typical functionalities of a database management system in the cloud.
- [12] "The year of Cloud adoption by the Channel" . compuBase. March 2013. Retrieved 13 February. Check date values in: `laccessdate= (help)`
- [13] http://www.cio.com/article/109704/Software_as_a_Service_SaaS_Definition_and_Solutions
- [14] Byron Deeter and Ryan Jung (July 2013). "Software as a Service Pricing Strategies" .
- [15] Wainwright, Phil (19 October 2007). "Workstream prefers virtualization to multi-tenancy" . *ZDNet*. CBS Interactive. Retrieved 24 April 2011.
- [16] Carraro, Gianpaolo (21 June 2008). "I can't believe we are still talking about whether saas == multi-tenancy..." . *Gianpaolo's blog*. Microsoft Corporation. Retrieved 24 April 2011.
- [17] Creese, Guy (18 May 2010). "SaaS vs. Software: The Release Cycle for SaaS Is Usually (Not Always) Faster" . *Gartner blog*. Gartner, Inc. Retrieved 24 April 2011.
- [18] "Jumping to SaaS? Take Agile Software Development Along with You" . *DevX.com*. QuinStreet Inc. 8 January 2008. Retrieved 24 April 2011. `|first1= missing |last1= in Authors list (help)`
- [19] Lisserman, Miroslaw (20 December 2010). "SaaS And The Everlasting Security Concerns" . Forrester Research. Retrieved 24 April 2011.
- [20] Arthur, Charles (2010-12-14). "Google's ChromeOS means losing control of the data, warns GNU founder Richard Stallman | Technology | guardian.co.uk" . Guardian. Retrieved 2012-02-16.

- [21] Adhikari, Richard. "Why Richard Stallman Takes No Shine to Chrome." *LinuxInsider*, 15 December 2010.
- [22] Stallman, Richard (2011-09-20). "Who does that server really serve?". GNU, Boston Review. Retrieved 15 January 2012.
- [23] Examples: Hill, Benjamin Mako (1 Feb 2009). "Show Me the Code" . *Revealing Errors*. Retrieved 15 January 2012. Assange, Julian (April 9, 2011). (Interview). RT. London <http://rt.com/news/wikileaks-revelations-assange-interview/>. Retrieved 15 January 2012. Facebook, Google, Yahoo – all these major US organizations have built-in interfaces for US intelligence. It's not a matter of serving a subpoena. They have an interface that they have developed for US intelligence to use. Missing or empty title= (help)
- [24] Gallagher, John. "Information Systems: A Manager's Guide to Harnessing Technology" . Flat World Knowledge. Retrieved 2012-04-21.
- [25] "Cloud Software as a Service (SaaS) in Cloud Computing. This is not rightServices" . *CloudComputingSec*. 2011. Retrieved 2011-12-15.
- [26] Garofalo, Josh. "Why SaaS is Broken (and how we're going to fix it)". *blitzen.com*.
- [27] <http://saugatucktechnology.com/research/latest-research/1595-255str-saas-integration-platforms-the-looming-saas-deployment-and-support.html>
- [28] Wilson, Deborah R; BonaPart, Alexa (7 August 7, 2009). "Develop a Framework for SaaS Application Business Continuity Risk Mitigation" . *Gartner.com*. Gartner, Inc. Retrieved 24 April 2011. Check date values in: |date= (help)
- [29] "SaaS Data Escrow International Report" (PDF). *RainStor* (Gloucester, England: Clearpace Software Ltd). 14 December 2009. Retrieved 24 April 2011. Only 15 per cent of those who currently use, or plan to use, SaaS have no inclination to keep a copy of their data.
- [30] <https://www.gnu.org/philosophy/who-does-that-server-really-serve.html>
- [31] Stallman, Richard (18 March 2010). "Who Does That Server Really Serve?". Boston Review. Retrieved 6 July 2013.
- [32] Woods, Dan (9 November 2010). "A New Open-Source Model For SaaS" . Forbes. Retrieved 21 September 2014.
- [33] Hoppin, Andrew (9 January 2014). "OpenSaaS and the future of government innovation" . *OpenSource.com*. Retrieved 21 September 2014.

Chapter 22

Cloud computing issues

Cloud Computing has become a social phenomenon being used by most people everyday. As with every important social phenomenon there are issues that limit its widespread adoption.

Most issues start from the fact that the user loses control of his or her data, because it is stored on someone else's computer (the cloud provider). This happens when the owner of the remote servers is a person or organisation other than the user, as their interests may point in different directions, for example, the user may wish that his or her information is kept private, but the owner of the remote servers may want to take advantage of it for their own business.

There are many issues relating to cloud computing, some of which are discussed here:

22.1 Threats and opportunities of the cloud

Critical voices including GNU project initiator Richard Stallman and Oracle founder Larry Ellison warned that the whole concept is rife with privacy and ownership concerns and constitute merely a fad.*[1]

However, cloud computing continues to gain steam*[2] with 56% of the major European technology decision-makers estimate that the cloud is a priority in 2013 and 2014, and the cloud budget may reach 30% of the overall IT budget.*[3]

According to the *TechInsights Report 2013: Cloud Succeeds* based on a survey, the cloud implementations generally meets or exceeds expectations across major service models, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).*[4]

Several deterrents to the widespread adoption of cloud computing remain. Among them are reliability, availability of services and data, security, complexity, costs, regulations and legal issues, performance, migration, reversion, the lack of standards, limited customization and issues of privacy. The *cloud* offers many strong points: infrastructure flexibility, faster deployment of applications and data, cost control, adaptation of cloud resources to real needs, improved productivity, etc. The early 2010s cloud market is dominated by software and services in SaaS mode and IaaS (infrastructure), especially the private cloud. PaaS and the public cloud are further back.

22.2 Privacy

The increased use of cloud computing services such as Gmail and Google Docs has pressed the issue of privacy concerns of cloud computing services to the utmost importance.*[5] The provider of such services lie in a position such that with the greater use of cloud computing services has given access to a plethora of data.*[5] This access has the immense risk of data being disclosed either accidentally or deliberately.*[5] Privacy advocates have criticized the cloud model for giving hosting companies' greater ease to control—and thus, to monitor at will—communication between host company and end user, and access user data (with or without permission). Instances such as the *secret NSA program*, working with AT&T, and Verizon, which recorded over 10 million telephone calls between American citizens, causes uncertainty among privacy advocates, and the greater powers it gives to telecommunication companies to monitor user activity.*[6]*[7] A cloud service provider (CSP) can complicate data privacy because of the extent of

virtualization (virtual machines) and **cloud storage** used to implement cloud service.* [8] CSP operations, customer or tenant data may not remain on the same system, or in the same data center or even within the same provider's cloud; this can lead to legal concerns over jurisdiction. While there have been efforts (such as **US-EU Safe Harbor**) to “harmonise” the legal environment, providers such as Amazon still cater to major markets (typically to the United States and the **European Union**) by deploying local infrastructure and allowing customers to select “regions and availability zones” .*[9] Cloud computing poses privacy concerns because the service provider can access the data that is on the cloud at any time. It could accidentally or deliberately alter or even delete information.*[10] This becomes a major concern as these service providers, who employ administrators which can leave room for potential unwanted disclosure of information on the cloud.*[5]

22.2.1 Sharing information without a warrant

Many cloud providers can share information with third parties if necessary for purposes of law and order even without a warrant. That is permitted in their privacy policies which users have to agree to before they start using cloud services.

There are life-threatening situations in which there is no time to wait for the police to issue a warrant. Many cloud providers can share information immediately to the police in such situations.

Example of a Privacy Policy that allows this

The Dropbox Privacy policy states that *[11]

We may share information as discussed below ...

Law & Order. We may disclose your information to third parties if we determine that such disclosure is reasonably necessary to (a) comply with the law; (b) protect any person from death or serious bodily injury; (c) prevent fraud or abuse of Dropbox or our users; or (d) protect Dropbox's property rights.

Previous situation about this

The Sydney Morning Herald reported about the **Mosman bomb hoax**, which was a life threatening situation, that: *[12]

As to whether NSW Police needed a warrant to access the information it was likely to have, Byrne said it depended on the process taken. “Gmail does set out in their process in terms of their legal disclosure guidelines [that] it can be done by a search warrant ... but there are exceptions that can apply in different parts of the world and different service providers. For example, Facebook generally provides an exception for emergency life threatening situations that are signed off by law enforcement.”

Another computer forensic expert at iT4ensics, which works for large corporations dealing with matters like internal fraud, Scott Lasak, said that police “would just contact Google” and “being of a police or FBI background Google would assist them” .

“Whether or not they need to go through warrants or that sort of thing I'm not sure. But even for just an IP address they might not even need a warrant for something like that being of a police background.

...

NSW Police would not comment on whether it had received help from Google. The search giant also declined to comment, instead offering a standard statement on how it cooperated with law enforcement.

A spokesman for the online users' lobby group Electronic Frontiers Australia, Stephen Collins, said Google was likely to have handed over the need information on the basis of “probable cause or a warrant” , which he said was “perfectly legitimate” . He also said “It happens with relative frequency. ...Such things are rarely used in Australia for trivial or malevolent purposes.”

22.3 Privacy solutions

Solutions to privacy in cloud computing include policy and legislation as well as end users' choices for how data is stored.*[5] The cloud service provider needs to establish clear and relevant policies that describe how the data of each cloud user will be accessed and used.*[5] Cloud service users can encrypt data that is processed or stored within the cloud to prevent unauthorized access.*[5]

22.4 Compliance

To comply with regulations including FISMA, HIPAA, and SOX in the United States, the Data Protection Directive in the EU and the credit card industry's PCI DSS, users may have to adopt *community* or *hybrid* deployment modes that are typically more expensive and may offer restricted benefits. This is how Google is able to “manage and meet additional government policy requirements beyond FISMA” * [13]* [14] and Rackspace Cloud or QubeSpace are able to claim PCI compliance.* [15]

Many providers also obtain a SAS 70 Type II audit, but this has been criticised on the grounds that the hand-picked set of goals and standards determined by the auditor and the auditee are often not disclosed and can vary widely.* [16] Providers typically make this information available on request, under *non-disclosure agreement*.* [17]* [18]

Customers in the EU contracting with cloud providers outside the EU/EEA have to adhere to the EU regulations on export of personal data.* [19]

U.S. Federal Agencies have been directed by the Office of Management and Budget to use a process called FedRAMP (Federal Risk and Authorization Management Program) to assess and authorize cloud products and services. Federal CIO Steven VanRoekel issued a memorandum to federal agency Chief Information Officers on December 8, 2011 defining how federal agencies should use FedRAMP. FedRAMP consists of a subset of NIST Special Publication 800-53 security controls specifically selected to provide protection in cloud environments. A subset has been defined for the FIPS 199 low categorization and the FIPS 199 moderate categorization. The FedRAMP program has also established a Joint Accreditation Board (JAB) consisting of Chief Information Officers from DoD, DHS, and GSA. The JAB is responsible for establishing accreditation standards for 3rd party organizations who perform the assessments of cloud solutions. The JAB also reviews authorization packages, and may grant provisional authorization (to operate). The federal agency consuming the service still has final responsibility for final authority to operate.* [20]

A multitude of laws and regulations have forced specific compliance requirements onto many companies that collect, generate or store data. These policies may dictate a wide array of data storage policies, such as how long information must be retained, the process used for deleting data, and even certain recovery plans. Below are some examples of compliance laws or regulations.

- United States, the Health Insurance Portability and Accountability Act (HIPAA) requires a contingency plan that includes, data backups, data recovery, and data access during emergencies.
- The privacy laws of Switzerland demand that private data, including emails, be physically stored in Switzerland.
- In the United Kingdom, the Civil Contingencies Act of 2004 sets forth guidance for a business contingency plan that includes policies for data storage.

In a virtualized cloud computing environment, customers may never know exactly where their data is stored. In fact, data may be stored across multiple data centers in an effort to improve reliability, increase performance, and provide redundancies. This geographic dispersion may make it more difficult to ascertain legal jurisdiction if disputes arise.* [21]

22.5 Legal

As with other changes in the landscape of computing, certain legal issues arise with cloud computing, including trademark infringement, security concerns and sharing of proprietary data resources.

The Electronic Frontier Foundation has criticized the United States government during the Megaupload seizure process for considering that people lose *property rights* by storing data on a cloud computing service.* [22]

One important but not often mentioned problem with cloud computing is the problem of who is in “possession” of the data. If a cloud company is the possessor of the data, the possessor has certain legal rights. If the cloud company is the “custodian” of the data, then a different set of rights would apply. The next problem in the legalities of cloud computing is the problem of legal ownership of the data. Many Terms of Service agreements are silent on the question of ownership.* [23]

These legal issues are not confined to the time period in which the cloud-based application is actively being used. There must also be consideration for what happens when the provider-customer relationship ends. In most cases, this event will be addressed before an application is deployed to the cloud. However, in the case of provider insolvencies or bankruptcy the state of the data may become blurred.* [21]

22.6 Vendor lock-in

Because cloud computing is still relatively new, standards are still being developed.*[24] Many cloud platforms and services are proprietary, meaning that they are built on the specific standards, tools and protocols developed by a particular vendor for its particular cloud offering.*[24] This can make migrating off a proprietary cloud platform prohibitively complicated and expensive.*[24]

Three types of vendor lock-in can occur with cloud computing:*[25]

- Platform lock-in: cloud services tend to be built on one of several possible virtualization platforms, for example VMWare or Xen. Migrating from a cloud provider using one platform to a cloud provider using a different platform could be very complicated.
- Data lock-in: since the cloud is still new, standards of ownership, i.e. who actually owns the data once it lives on a cloud platform, are not yet developed, which could make it complicated if cloud computing users ever decide to move data off of a cloud vendor's platform.
- Tools lock-in: if tools built to manage a cloud environment are not compatible with different kinds of both virtual and physical infrastructure, those tools will only be able to manage data or apps that live in the vendor's particular cloud environment.

Heterogeneous cloud computing is described as a type of cloud environment that prevents vendor lock-in, and aligns with enterprise data centers that are operating hybrid cloud models.*[26] The absence of vendor lock-in lets cloud administrators select his or her choice of hypervisors for specific tasks, or to deploy virtualized infrastructures to other enterprises without the need to consider the flavor of hypervisor in the other enterprise.*[27]

A heterogeneous cloud is considered one that includes on-premise private clouds, public clouds and software-as-a-service clouds. Heterogeneous clouds can work with environments that are not virtualized, such as traditional data centers.*[28] Heterogeneous clouds also allow for the use of piece parts, such as hypervisors, servers, and storage, from multiple vendors.*[29]

Cloud piece parts, such as cloud storage systems, offer APIs but they are often incompatible with each other.*[30] The result is complicated migration between backends, and makes it difficult to integrate data spread across various locations.*[30] This has been described as a problem of vendor lock-in.*[30] The solution to this is for clouds to adopt common standards.*[30]

Heterogeneous cloud computing differs from homogeneous clouds, which have been described as those using consistent building blocks supplied by a single vendor.*[31] Intel General Manager of high-density computing, Jason Waxman, is quoted as saying that a homogeneous system of 15,000 servers would cost \$6 million more in capital expenditure and use 1 megawatt of power.*[31]

22.7 Open source

See also: [Category:Free software for cloud computing](#)

Open-source software has provided the foundation for many cloud computing implementations, prominent examples being the Hadoop framework*[32] and VMware's Cloud Foundry.*[33] In November 2007, the Free Software Foundation released the Affero General Public License, a version of GPLv3 intended to close a perceived legal loophole associated with free software designed to run over a network.*[34]

22.8 Open standards

See also: [Category:Cloud standards](#)

Most cloud providers expose APIs that are typically well documented (often under a Creative Commons license*[35]) but also unique to their implementation and thus not interoperable. Some vendors have adopted others' APIs and there are a number of open standards under development, with a view to delivering interoperability and portability.*[36]

As of November 2012, the Open Standard with broadest industry support is probably OpenStack, founded in 2010 by NASA and Rackspace, and now governed by the OpenStack Foundation. [37] OpenStack supporters include AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP, IBM, Yahoo and now VMware. [38]

22.9 Security

Main article: Cloud computing security

As cloud computing is achieving increased popularity, concerns are being voiced about the security issues introduced through adoption of this new model. [39] [40] The effectiveness and efficiency of traditional protection mechanisms are being reconsidered as the characteristics of this innovative deployment model can differ widely from those of traditional architectures. [41] An alternative perspective on the topic of cloud security is that this is but another, although quite broad, case of “applied security” and that similar security principles that apply in shared multi-user mainframe security models apply with cloud security. [42]

The relative security of cloud computing services is a contentious issue that may be delaying its adoption. [43] Physical control of the Private Cloud equipment is more secure than having the equipment off site and under someone else's control. Physical control and the ability to visually inspect data links and access ports is required in order to ensure data links are not compromised. Issues barring the adoption of cloud computing are due in large part to the private and public sectors' unease surrounding the external management of security-based services. It is the very nature of cloud computing-based services, private or public, that promote external management of provided services. This delivers great incentive to cloud computing service providers to prioritize building and maintaining strong management of secure services. [44] Security issues have been categorised into sensitive data access, data segregation, privacy, bug exploitation, recovery, accountability, malicious insiders, management console security, account control, and multi-tenancy issues. Solutions to various cloud security issues vary, from cryptography, particularly public key infrastructure (PKI), to use of multiple cloud providers, standardisation of APIs, and improving virtual machine support and legal support. [41] [45] [46]

Cloud computing offers many benefits, but is vulnerable to threats. As cloud computing uses increase, it is likely that more criminals find new ways to exploit system vulnerabilities. Many underlying challenges and risks in cloud computing increase the threat of data compromise. To mitigate the threat, cloud computing stakeholders should invest heavily in risk assessment to ensure that the system encrypts to protect data, establishes trusted foundation to secure the platform and infrastructure, and builds higher assurance into auditing to strengthen compliance. Security concerns must be addressed to maintain trust in cloud computing technology. [39]

Data breach is a big concern in cloud computing. A compromised server could significantly harm the users as well as cloud providers. A variety of information could be stolen. These include credit card and social security numbers, addresses, and personal messages. The U.S. now requires cloud providers to notify customers of breaches. Once notified, customers now have to worry about identity theft and fraud, while providers have to deal with federal investigations, lawsuits and reputational damage. Customer lawsuits and settlements have resulted in over \$1 billion in losses to cloud providers. [47]

22.10 Sustainability

Although cloud computing is often assumed to be a form of *green computing*, there is currently no way to measure how “green” computers are. [48]

The primary environmental problem associated with the cloud is energy use. Phil Radford of Greenpeace said “we are concerned that this new explosion in electricity use could lock us into old, polluting energy sources instead of the clean energy available today.” [49] Greenpeace ranks the energy usage of the top ten big brands in cloud computing, and successfully urged several companies to switch to clean energy. On Thursday, December 15, 2011, Greenpeace and Facebook announced together that Facebook would shift to use clean and renewable energy to power its own operations. [50] [51] Soon thereafter, Apple agreed to make all of its data centers ‘coal free’ by the end of 2013 and doubled the amount of solar energy powering its Maiden, NC data center. [52] Following suit, Salesforce agreed to shift to 100% clean energy by 2020. [53]

Citing the servers' effects on the environmental effects of cloud computing, in areas where climate favors natural cooling and renewable electricity is readily available, the environmental effects will be more moderate. (The same

holds true for “traditional” data centers.) Thus countries with favorable conditions, such as Finland,*[54] Sweden and Switzerland,*[55] are trying to attract cloud computing data centers. Energy efficiency in cloud computing can result from energy-aware scheduling and server consolidation.*[56] However, in the case of distributed clouds over data centers with different sources of energy including renewable energy, the use of energy efficiency reduction could result in a significant carbon footprint reduction.*[57]

22.11 Abuse

As with privately purchased hardware, customers can purchase the services of cloud computing for nefarious purposes. This includes password cracking and launching attacks using the purchased services.*[58] In 2009, a banking trojan illegally used the popular Amazon service as a command and control channel that issued software updates and malicious instructions to PCs that were infected by the malware.*[59]

22.12 IT governance

Main article: Corporate governance of information technology

The introduction of cloud computing requires an appropriate IT governance model to ensure a secured computing environment and to comply with all relevant organizational information technology policies.*[60]*[61] As such, organizations need a set of capabilities that are essential when effectively implementing and managing cloud services, including demand management, relationship management, data security management, application lifecycle management, risk and compliance management.*[62] A danger lies with the explosion of companies joining the growth in cloud computing by becoming providers. However, many of the infrastructural and logistical concerns regarding the operation of cloud computing businesses are still unknown. This over-saturation may have ramifications for the industry as a whole.*[63]

22.13 Consumer end storage

The increased use of cloud computing could lead to a reduction in demand for high storage capacity consumer end devices, due to cheaper low storage devices that stream all content via the cloud becoming more popular. In a Wired article, Jake Gardner explains that while unregulated usage is beneficial for IT and tech moguls like Amazon, the anonymous nature of the cost of consumption of cloud usage makes it difficult for business to evaluate and incorporate it into their business plans.*[63]

22.14 Ambiguity of terminology

Outside of the information technology and software industry, the term “cloud” can be found to reference a wide range of services, some of which fall under the category of cloud computing, while others do not. The cloud is often used to refer to a product or service that is discovered, accessed and paid for over the Internet, but is not necessarily a computing resource. Examples of service that are sometimes referred to as “the cloud” include, but are not limited to, crowd sourcing, cloud printing, crowd funding, cloud manufacturing.*[64]*[65]

22.15 Performance interference and noisy neighbors

Due to its multi-tenant nature and resource sharing, cloud computing must also deal with the “noisy neighbor” effect. This effect in essence indicates that in a shared infrastructure, the activity of a virtual machine on a neighboring core on the same physical host may lead to increased performance degradation of the VMs in the same physical host, due to issues such as e.g. cache contamination. Due to the fact that the neighboring VMs may be activated or deactivated at arbitrary times, the result is an increased variation in the actual performance of cloud resources. This effect seems to be dependent on the nature of the applications that run inside the VMs but also other factors such as scheduling

parameters and the careful selection may lead to optimized assignment in order to minimize the phenomenon. This has also led to difficulties in comparing various cloud providers on cost and performance using traditional benchmarks for service and application performance, as the time period and location in which the benchmark is performed can result in widely varied results.*[66] This observation has led in turn to research efforts to make cloud computing applications intrinsically aware of changes in the infrastructure so that the application can automatically adapt to avoid failure.*[67]

22.16 Monopolies and privatization of cyberspace

Philosopher Slavoj Žižek points out that, although cloud computing enhances content accessibility, this access is “increasingly grounded in the virtually monopolistic privatization of the cloud which provides this access”. According to him, this access, necessarily mediated through a handful of companies, ensures a progressive privatization of global cyberspace. Žižek criticises the argument purported by supporters of cloud computing that this phenomenon is part of the “natural evolution” of the Internet, sustaining that the quasi-monopolies “set prices at will but also filter the software they provide to give its “universality” a particular twist depending on commercial and ideological interests.”*[68]

22.17 See also

- Cloud computing

22.18 References

- [1] Bobby Johnston. Cloud computing is a trap, warns GNU founder Richard Stallman. *The Guardian*, 29 September 2008.
- [2] http://www.morganstanley.com/views/perspectives/cloud_computing.pdf
- [3] *Challenges & Opportunities for IT partners when transforming or creating a business in the Cloud*. compuBase consulting. 2012. p. 77.
- [4] Cloud Computing Grows Up: Benefits Exceed Expectations According to Report. Press Release, May 21, 2013.
- [5] ryan, mark (January 2011). “Cloud Computing Privacy Concerns on Our Doorstep.” . *ACM*.
- [6] Cauley, Leslie (2006-05-11). “NSA has massive database of Americans' phone calls” . *USA Today*. Retrieved 2010-08-22.
- [7] “NSA taps in to user data of Facebook, Google and others, secret files reveal” . 2013-06-07. Retrieved 2013-06-07.
- [8] Winkler, Vic (2011). *Securing the Cloud: Cloud Computer Security Techniques and Tactics*. Waltham, Massachusetts: Elsevier. p. 60. ISBN 978-1-59749-592-9.
- [9] “Regions and Availability Zones” . Amazon Web Services Documentation. Retrieved 2014-01-24.
- [10] “Cloud Computing Privacy Concerns on Our Doorstep” .
- [11] “Dropbox Privacy Policy” . Retrieved 2014-12-04.
- [12] “'Every touch leaves a trace': how Google helped track bomb hoax suspect” . Retrieved 2014-12-04.
- [13] “FISMA compliance for federal cloud computing on the horizon in 2010” . SearchCompliance.com. Retrieved 2010-08-22.
- [14] “Google Apps and Government” . Official Google Enterprise Blog. 2009-09-15. Retrieved 2010-08-22.
- [15] “Cloud Hosting is Secure for Take-off: Mosso Enables The Spreadsheet Store, an Online Merchant, to become PCI Compliant” . Rackspace. 2009-03-14. Retrieved 2010-08-22.
- [16] “Amazon gets SAS 70 Type II audit stamp, but analysts not satisfied” . SearchCloudComputing.com. 2009-11-17. Retrieved 2010-08-22.
- [17] “Assessing Cloud Computing Agreements and Controls” . WTN News. Retrieved 2010-08-22.

- [18] “Cloud Certification From Compliance Mandate to Competitive Differentiator” . Cloudcor. Retrieved 2011-09-20.
- [19] “How the New EU Rules on Data Export Affect Companies in and Outside the EU | Dr. Thomas Helbing – Kanzlei für Datenschutz-, Online- und IT-Recht” . Dr. Thomas Helbing. Retrieved 2010-08-22.
- [20] “FedRAMP” . *U.S. General Services Administration*. 2012-06-13. Retrieved 2012-06-17.
- [21] Chambers, Don (July 2010). “Windows Azure: Using Windows Azure’ s Service Bus to Solve Data Security Issues” . *Rebus Technologies*. Retrieved 2012-12-14.
- [22] Cohn, Cindy; Samuels, Julie (31 October 2012). “Megupload and the Government's Attack on Cloud Computing”]. *Electronic Frontier Foundation*. Retrieved 2012-12-14.
- [23] Maltais, Michelle (26 April 2012). “Who owns your stuff in the cloud?”. *Los Angeles Times*. Retrieved 2012-12-14.
- [24] McKendrick, Joe. (2011-11-20) “Cloud Computing's Vendor Lock-In Problem: Why the Industry is Taking a Step Backward,” *Forbes.com*
- [25] Hinkle, Mark. (2010-6-9) “Three cloud lock-in considerations” , *Zenoss Blog*
- [26] Staten, James (2012-07-23). “Gelsinger brings the 'H' word to VMware” . *ZDNet*.
- [27] Vada, Eirik T. (2012-06-11) “Creating Flexible Heterogeneous Cloud Environments” , page 5, *Network and System Administration, Oslo University College*
- [28] Geada, Dave. (June 2, 2011) “The case for the heterogeneous cloud,” *Cloud Computing Journal*
- [29] Burns, Paul (2012-01-02). “Cloud Computing in 2012: What's Already Happening” . *Neovise*.
- [30] Livenson, Ilja. Laure, Erwin. (2011) “Towards transparent integration of heterogeneous cloud storage platforms” , pages 27–34, *KTH Royal Institute of Technology, Stockholm, Sweden*.
- [31] Gannes, Liz. GigaOm, “Structure 2010: Intel vs. the Homogeneous Cloud,” June 24, 2010.
- [32] Jon Brodtkin (July 28, 2008). “Open source fuels growth of cloud computing, software-as-a-service” . *Network World*. Retrieved 2012-12-14.
- [33] “VMware Launches Open Source PaaS Cloud Foundry” . 2011-04-21. Retrieved 2012-12-14.
- [34] “AGPL: Open Source Licensing in a Networked Age” . *Redmonk.com*. 2009-04-15. Retrieved 2010-08-22.
- [35] *GoGrid Moves API Specification to Creative Commons*
- [36] “Eucalyptus Completes Amazon Web Services Specs with Latest Release” . *Ostatic.com*. Retrieved 2010-08-22.
- [37] “OpenStack Foundation launches” . *Infoworld.com*. 2012-09-19. Retrieved 2012-11-17.
- [38] “Did OpenStack Let VMware Into The Henhouse?”. *InformationWeek*. 2012-10-19. Retrieved 2012-11-17.
- [39] Mariana Carroll, Paula Kotzé, Alta van der Merwe (2012). “Securing Virtual and Cloud Environments” . In I. Ivanov et al. *Cloud Computing and Services Science, Service Science: Research and Innovations in the Service Economy*. Springer Science+Business Media. doi:10.1007/978-1-4614-2326-3.
- [40] M Carroll, P Kotzé, Alta van der Merwe (2011), *Secure virtualization: benefits, risks and constraints*, 1st International Conference on Cloud Computing and Services Science, Noordwijkerhout, The Netherlands, 7–9 May 2011
- [41] Zisis, Dimitrios; Lekkas (2010). “Addressing cloud computing security issues” . *Future Generation Computer Systems* **28** (3): 583. doi:10.1016/j.future.2010.12.006.
- [42] Winkler, Vic (2011). *Securing the Cloud: Cloud Computer Security Techniques and Tactics*. Waltham, MA USA: Syngress. pp. 187, 189. ISBN 978-1-59749-592-9.
- [43] “Are security issues delaying adoption of cloud computing?”. *Network World*. Retrieved 2010-08-22.
- [44] “Security of virtualization, cloud computing divides IT and security pros” . *Network World*. 2010-02-22. Retrieved 2010-08-22.
- [45] Armbrust, M; Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Zaharia, (2010). “A view of cloud computing” . *Communication of the ACM* **53** (4): 50–58. doi:10.1145/1721654.1721672.
- [46] Anthens, G (2010). “Security in the cloud” . *Communications of the ACM* **53** (11): 16. doi:10.1145/1839676.1839683.

- [47] Bowen, Janine Anthony. (2011). Cloud Computing: Issues in Data Privacy/Security and Commercial Considerations. *Computer and Internet Lawyer Trade Journal*. 28 (8), 8.
- [48] James Urquhart (January 7, 2010). "Cloud computing's green paradox" . CNET News. Retrieved March 12, 2010. ... there is some significant evidence that the cloud is encouraging more compute consumption
- [49] "Dirty Data Report Card" . Greenpeace. Retrieved 2013-08-22.
- [50] "Facebook and Greenpeace settle Clean Energy Feud" . Techcrunch. Retrieved 2013-08-22.
- [51] "Facebook Commits to Clean Energy Future" . Greenpeace. Retrieved 2013-08-22.
- [52] "Apple is leaving Microsoft and Amazon in 'dust' for its clean internet efforts – Greenpeace" . Greenpeace. Retrieved 2013-08-22.
- [53] "Salesforce Announces Commitment to a Cloud Powered by 100% Renewable Energy" . Greenpeace. Retrieved 2013-08-22.
- [54] Finland – First Choice for Siting Your Cloud Computing Data Center.. Retrieved 4 August 2010.
- [55] Swiss Carbon-Neutral Servers Hit the Cloud.. Retrieved 4 August 2010.
- [56] Berl, Andreas, et al., Energy-Efficient Cloud Computing, *The Computer Journal*, 2010.
- [57] Farrahi Moghaddam, Fereydoun, et al., Low Carbon Virtual Private Clouds, *IEEE Cloud 2011*.
- [58] Alpeyev, Pavel (2011-05-14). "Amazon.com Server Said to Have Been Used in Sony Attack" . Bloomberg. Retrieved 2011-08-20.
- [59] Goodin, Dan (2011-05-14). "PlayStation Network hack launched from Amazon EC2" . *The Register*. Retrieved 2012-05-18.
- [60] Hsu, Wen-Hsi L., "Conceptual Framework of Cloud Computing Governance Model – An Education Perspective" , *IEEE Technology and Engineering Education (ITEE)*, Vol 7, No 2 (2012)
- [61] Stackpole, Beth, "Governance Meets Cloud: Top Misconceptions" , *InformationWeek*, 7 May 2012
- [62] Joha, A and M. Janssen (2012) "Transformation to Cloud Services Sourcing: Required IT Governance Capabilities" , *ICST Transactions on e-Business* 12(7-9)
- [63] Gardner, Jake (2013-03-28). "Beware: 7 Sins of Cloud Computing" . *Wired*. Retrieved 2013-06-20.
- [64] S. Stonham and S. Nahalkova (2012) "What is the Cloud and how can it help my business?"
- [65] S. Stonham and S. Nahalkova (2012), Whitepaper "Tomorrow Belongs to the Agile (PDF)"
- [66] George Kousiouris, Tommaso Cucinotta, Theodora Varvarigou, "The Effects of Scheduling, Workload Type and Consolidation Scenarios on Virtual Machine Performance and their Prediction through Optimized Artificial Neural Networks" , *The Journal of Systems and Software* (2011), Volume 84, Issue 8, August 2011, pp. 1270–1291, Elsevier, doi:10.1016/j.jss.2011.04.013.
- [67] <http://www.cloudwave-f7.eu>
- [68] Slavoj Žižek (May 2, 2011). "Corporate Rule of Cyberspace" . *Inside Higher Ed*. Retrieved July 10, 2013.

Chapter 23

Cloud computing comparison

The following is a comparison of some cloud computing software and providers. A more comprehensive list can be found at the foot of this page.

23.1 General information

23.2 Supported Hosts

(what the cloud software runs on)

23.3 Supported Clients

(what the cloud software will run as a virtual instance)

23.4 Providers

PaaS providers which can run on IaaS providers (“itself” means the provider is both PaaS and IaaS):

PaaS pricing comparison of hosting services:

23.5 Features

- Failover - supports automatic handling of hardware failures (Failover) (partial indicates failovers of controller nodes are not supported)
- OCCI - supports Open Cloud Computing Interface
- vCloud - supports vCloud migration
- S3 - supports Amazon S3 volumes

23.6 See also

23.7 References

- [1] Urquhart, James (June 22, 2009). “The new generation of cloud-development platforms.” CNET News. Accessed November 2011.
- [2] Harris, Derrick Harris (October 22, 2010). “Microsoft Joins OpenStack to Add Hyper-V Support.” Gigaom.com. Accessed November 2011.
- [3] Prickett Timothy M. (May 10, 2011). “Ubuntu eats OpenStack for clouds - Eucalyptus leaves.” *The Register*. Accessed November 2011.
- [4] Info-Tech Research Group (October 24, 2012). “Vendor Landscape: Cloud Management.” . Accessed January 2013.
- [5] European Commission Expert Group Report (January 26, 2010). “The Future of Cloud Computing”
- [6] " OpenQRM Enterprise Architecture (February 24, 2010)
- [7] Schabell, Eric (December 2012). *OpenShift Primer “Ebook”* . Developer.Press.
- [8] Pepple, Ken (August 2011). *Deploying OpenStack*. O'Reilly Media. ISBN 1-4493-1105-9.
- [9] Poul Weiss " Youtube.com install instruction for node cluster." Youtube Video.12 Oct 2012.
- [10] “Apache CloudStack Features - Wide Range Guest VM OS Support” .

23.8 Text and image sources, contributors, and licenses

23.8.1 Text

- Virtualization** *Source:* <http://en.wikipedia.org/wiki/Virtualization?oldid=642075507> *Contributors:* Breakpoint, Expatrik, Furrykef, Jeffq, Bearcat, Pengo, Tobias Bergemann, Kak, Uzume, Ency, Velella, Versageek, Mahanga, Woohookitty, Pol098, Siddhant, Wavelength, Bhny, Jengelh, SmackBot, Gribeco, Vermorel, Thumperward, Nick Levine, Warren, 16@r, Kvng, Quaeler, TerryE, Colonel Warden, UncleDouggy, Artemgy, FleetCommand, Sir Lothar, Yukoba, Gogo Dodo, Kubanczyk, Widefox, VictorAnyakin, JAnDbot, NapoliRoma, Kitdaddio, Dekimasu, GermanX, In Transit, Funandrvl, VolkovBot, Wrev, Zhenqinli, Heiser, JerzyTarasiuk, Svick, StaticGull, Farrahi, CiudadanoGlobal, Lms2007, RafaAzevedo, Pointillist, Swtechwr, Vdmeraj, SF007, XLinkBot, Nepenthes, Dsimic, Bazj, Addbot, Ghet-toblaster, KitchM, MrOllie, Jasper Deng, Jarble, Luckas-bot, Pcap, Wadamja, Jkeogh, AnomieBOT, MaterialsScientist, M.amjad.saeed, Nakakapagpabagabag, Zero Thrust, Geek2003, Wisewave, Boleyn3, Umawera, Pinethicket, HRoestBot, MHPSM, Yunshui, Wo.lawen, Rakarlin, Ripchip Bot, Smbat.petrosyan, Virtimo, Dbastro, EmausBot, Sliceofmiami, Bathawes, GoingBatty, Icantthinkofaname4msn, L Kensington, Sunil256, PetrB, ClueBot NG, MelbourneStar, Ardahal.nitw, Cntras, Widr, Opensystemsmedia, Blewisopensystemsme-dia, Bslewis1, Texasfight2010, Wasbeer, Titizebioutifoul, Michael Vick PHI, JMJeditor, Collinusc, Glacialfox, Anshuln95, Pdelvecchio, Darylgolden, Softwareqa, Calsoft123, ChrisGualtieri, Dwright7861, SimonBramfitt, Giso6150, Codename Lisa, Webclient101, Jamesy1, Lugia2453, Stefmorrow, Dave Braunschweig, DBhavsar709, Ruby Murray, Ckolias, Brk0 0, ScotXW, Wills64, Monkbot, Shrutib6182 and Anonymous: 134
- Hardware virtualization** *Source:* <http://en.wikipedia.org/wiki/Hardware%20virtualization?oldid=643479034> *Contributors:* Pnm, Ronz, Nurg, Rsduhamel, Dratman, Cmprince, Pol098, Bilbo1507, Rjwilmsi, Bernard van der Wees, Chobot, Wavelength, SmackBot, DagEr-lingSmørgrav, Dl2000, Mrozlog, UncleDouggy, Sir Lothar, Mblumber, JAnDbot, Honette, Kwesterh, Cucinotta, Billinghurst, Heiser, Ooiww, Jerryobject, Svick, Niceguyedc, Rrpr, Jusdafax, Zodon, Dsimic, Addbot, Mortense, Thecarpy, MrOllie, Jasper Deng, Yobot, Pcap, Becky Sayles, KamikazeBot, AnomieBOT, Xqbot, Tayloj, Animist, Anna Frodesiak, FrescoBot, Skatesmen, William Surya Permana, A.wasylewski, Jesse V., Jfmantis, Jsdustin, Manasprakash79, Shencypeter, Robbiemorrison, John Aplesed, ClueBot NG, Jfalkent, Harihs, Ctopham, Codename Lisa, Bbartlomiej, Rotlink, YiFeiBot, Monkbot, ComsciStudent and Anonymous: 52
- Full virtualization** *Source:* <http://en.wikipedia.org/wiki/Full%20virtualization?oldid=619103932> *Contributors:* Uzume, Beland, EagleOne, Odalcet, GregorB, Sega381, Rjwilmsi, Chobot, Froth, Modify, Elonka, Gribeco, Jerome Charles Potts, Colonies Chris, Frap, Spinality, Quaeler, UncleDouggy, Mwarren us, Bradmkjr, Tedickey, Whiteglitter79, CommonsDelinker, Lilac Soul, Wrev, Biscuittin, EwokiWiki, Altzinn, ClueBot, Hidro, SF007, Dsimic, Addbot, Elsendero, MrOllie, Juggler2005, AnomieBOT, Woodyriddle, CXCVC, Smredo, Llukomsk, Expertour, Shshme, Weltmeister1990, Virtimo, Angrytoast, W163, ClueBot NG, Codename Lisa, Someone not using his real name and Anonymous: 15
- Paravirtualization** *Source:* <http://en.wikipedia.org/wiki/Paravirtualization?oldid=612470011> *Contributors:* Damian Yerrick, Leandro, Rl, Pabouk, Lupin, JimD, AlistairMcMillan, Khalid hassani, ConradPino, Kutulu, Imroy, Rich Farmbrough, Ylai, Linuxlad, Interiot, Surruena, Sunbiz3000, GregorB, Reisis, Pmc, Intgr, WikiWikiPhil, Wasted Time R, Pcoulombeau, Zigamorph, Daniel Bonniot de Ruis-selet, Svenstaro, Pegship, DrHok, Chip Zero, Banus, Mark hermeling, Amux, Thumperward, Frap, Adamantios, Tompsci, Warren, JzG, Nagle, Quaeler, Riordanmr, UncleDouggy, Paul Foxworthy, Xlynx, Bradmkjr, Hlsu, VolkovBot, Xanderdad, SieBot, Isthistingworking, Чръный человек, Addbot, Dawynn, Faltschuler, Elsendero, Yobot, Houselifter, PabloCastellano, Nav-wiki, Alfons.crespo, FrescoBot, Garandel, Virtimo, Cf. Hay, Khajaea, DavidLeighEllis and Anonymous: 64
- Hypervisor** *Source:* <http://en.wikipedia.org/wiki/Hypervisor?oldid=635631285> *Contributors:* AxelBoldt, Ghakko, Dcoetzee, Havardk, Visorstuff, Earthsound, Nurg, Brouhaha, Levin, Dratman, Dav4is, Mboverload, Uzume, ConradPino, Pbannister, TheObtuseAngleOf-Doom, Moxfyre, Thorwald, Praxis, Grstain, RossPatterson, Sbb, ChrisJ, Bender235, PaulMEdwards, Csabo, Andrewpmk, Snowolf, Velella, Ronark, DanShearer, Rothgar, Pauli133, Ringbang, Linas, JFG, GregorB, Mark Williamson, Marudubshinki, Rjwilmsi, Mfwills, PHenry, Pmc, FlaBot, Pruefer, Quuxplusone, Eugene Esterly III, Bgwhite, FrankTobia, YurikBot, Pcoulombeau, Jeffhoy, Bovineone, Роман Беккер, Dwarfpower, Cleared as filed, Voidxor, Scope creep, Drdavis, ThinkingInBinary, Zzuuzz, JLaTondre, ViperSnake151, Belandorsqueaks, Finell, Mark hermeling, SmackBot, Mmernex, Henriok, Omniuni, Gilliam, Brianski, Hmains, Chris the speller, Thumperward, Kevin Ryde, Tonyhinkle, Liontooth, Frap, Eliyahu S, Jacob Poon, Dweaver, Adamantios, UU, Meson537, Schultmc, Warren, Rudolph04, Spinality, Rmohns, Martinkop, Soumyasch, Nagle, Korejora, Smswigart, UncleDouggy, Thewebdruid, Raysonho, Cydebot, PamD, Thijs'bot, Kubanczyk, Nick Number, Aplemeister, Changlinn, Dougher, Skarkkai, JAnDbot, Magioladitis, Appraiser, Bswinnerton, Ehdr, Seashorewiki, Symbolt, Gwern, Delucardenal, Hlsu, Panarchy, Cyprien44, ItsProgrammable, Dave Tuttle, RenniePet, Juliancolton, GCFreak2, Milnivlek, MaxBrowne, Mezzaluna, Mwils024, DRady, Kbrose, Gambler132, Heiser, SieBot, EwokiWiki, Noveltyghost, XMNboy, AlanUS, Edifyyo, CP2002, Echo95, HumanBeing01, CiudadanoGlobal, Chaosfreak69, Glenikins, RanceDeLong, RobChafer, Niceguyedc, DragonBot, Socrates2008, Aeolian145, Anon lynx, Bennopia, Paul Lizer, Ne0922, SF007, Atxbyea, Dsimic, Addbot, Xionglifeng, Eamei, Elsendero, Ratokeshi, MrOllie, Enormator, Favonian, Tide rolls, Lightbot, Jarble, Ben Ben, Yobot, Bunny-hop11, Pcap, KamikazeBot, Peter Flass, Kehuston, Fultheim, AnomieBOT, Aneah, PabloCastellano, Smredo, LarryStevens, Kicekpicke, Technopedian, Terrivism, Nasa-verve, RibotBOT, Locobot, Alfons.crespo, Breadtk, Baishuwei, FrescoBot, Llukomsk, ChrixH, Winterst, Skyerise, Jandalhandler, Alexey Izbyshv, A.wasylewski, Linuxpundit, Hanky27, Phlegat, Dbastro, EmausBot, John of Reading, Barnet.Herts, LiamWestley, Slashusrbin, Bbalban, Bollyjeff, Kiwi128, Singaporian, Demonkoryu, OnePt618, Erget2005, Main-Frame, Voomoo, Crowdthink, Jeff Song, Bbruce32, IAmTheCandyman, Khajaea, Raghu Udiyar, Geekbychoice, Sumercip, GGShinobi, Snow Blizzard, Meabandit, Vxassist, Vicentenicolaugalleo, SimonBramfitt, Codename Lisa, Jnargus, Debrell, Wanderenvy, Bouncysbot, ScotXW, Aswirthm, Monkbot, Vmdocker and Anonymous: 224
- Hardware-assisted virtualization** *Source:* <http://en.wikipedia.org/wiki/Hardware-assisted%20virtualization?oldid=608150636> *Contributors:* Kku, Rchandra, Uzume, Supergloom, Woohookitty, Pol098, Bovineone, Elkman, Rwxrwxrwx, SmackBot, Henriok, Gribeco, Anthony Liguori, Kneufeld, Jerome Charles Potts, Spinality, Arkrishna, Quaeler, UncleDouggy, CmdrObot, Uconnbill, Kubanczyk, Davidhorman, Bradmkjr, Ptery, Tedickey, To Serve Man, Vanpedia, Andy Dingley, Jojalozzo, Gerardohc, Treekids, Some jerk on the Internet, Elsendero, Download, Yobot, Pcap, Woodyriddle, FrescoBot, Loodi, Shshme, Jandalhandler, Akolyth, Xzyzyavatar, Voomoo, Peter James, Iswantoumy, SimonBramfitt, Codename Lisa, Mark viking, Monkbot and Anonymous: 27
- Emulator** *Source:* <http://en.wikipedia.org/wiki/Emulator?oldid=634340488> *Contributors:* Damian Yerrick, Lee Daniel Crocker, CYD, Uriyan, Bryan Derksen, Robert Merkel, PierreAbbat, SimonP, Ben-Zin, Michael Hardy, Lxor, Pnm, Wapcaplet, Ixf64, TakuyaMura-rata, Yaronf, Dcoetzee, Tediuz Zanarukando, Andrewman327, Furrykef, Grendelkhan, Wernher, Jni, Robbot, Fredrik, Altenmann, Pingveno, Wikibot, David Gerard, McDutchie, Marc Venot, Ds13, Everyking, Siroxo, Khalid hassani, VampWillow, Edcolins, Mackeriv,

LiDaobing, Billposer, Dweddit, CesarFelipe, Urhixidur, Joyous!, Mschlindwein, M1sslontomars2k4, Zondor, Adashiel, DmitryKo, Oskar Sigvardsson, Mormegil, Freakofnurture, Sysy, Discospinster, Hcs, Evicce, Ht1848, Uli, Ylee, CanisRufus, Gen0cide, Matteh, Foobaz, Minghong, HasharBot, Alansohn, Liao, Jtalledo, Malo, Tony Sidaway, Mrja84, Gortu, SteinbDJ, Kay Dekker, Forteblast, Simmetrical, Qnonsense, Woohookitty, RHaworth, Trevor Andersen, ThomasHarte, Graham87, Qwertyus, Kbdank71, Yurik, JIP, Rjwilmsi, Mbutts, LjL, Ravuya, Xmoogole, Gurch, Viznut, SkiDragon, Kri, Chobot, Jpfagerbot, RobotE, Isuldor, Petiatil, Hede2000, Cpc464, Bovineone, Claulnia, Anomalocaris, Tim baker, Bota47, Caspian, Zunaid, Lt-wiki-bot, Icedog, Arthur Rubin, Reyk, GraemeL, GrinBot, One, Luk, Erik Sandberg, SmackBot, Mmernex, Mira, Maelwys, Darklock, Canthusus, Aksi great, Slo-mo, Chris the speller, Thumperward, Cornflake pirate, Darth Panda, Can't sleep, clown will eat me, Furby100, Addshore, RedHillian, Shadow, Nakon, Tom9729, SashatoBot, Njomany, Kuru, 041744, Thesoccerjunky4, Andyandy.UK, Rock4arolla, Arkrishna, Watchsmart, Mattabat, Heitzer, Manifestation, Wisegoldfish, Jestix, Tawkerbot2, Emote, ScottW, Randalllin, R9tfgokunks, Moonknightus, AndrewHowse, Dancter, HitroMilanese, Chrisk02, Slordak, Daniel Olsen, Thijs!bot, Wikid77, John254, JustAGal, TangentCube, Silver Edge, LachlanA, AntiVandalBot, Widefox, Jhsounds, Realko, Kethinov, Mark Grant, Malignant Manor, Greensburger, Magioladitis, Bongwarrior, VoABot II, Leftblank, Robotman1974, Trisar, Cander0000, Stephenchou0722, Moggie2002, Arjun01, Mmendezs, Haffner, Pharaoh of the Wizards, Mange01, Eufox69, SpigotMap, Emuguru, Rbakker99, DrunkSolo, Tygrrr, GCfreak2, Castlevaniamaster1, Jcea, VolkovBot, TXiKiBoT, Rei-bot, Gforce75, Commodorekid, Slysplace, LeaveSleeves, Master Bigode, WikiLoco, Malus Catulus, SheffieldSteel, Barroids, Synthebot, Helaynehag, Neparis, SieBot, Paul20070, YonaBot, X-Fi6, Yintan, Hac13, Flyer22, Oxymoron83, Techman224, StaticGull, ClueBot, Czarkoff, Mild Bill Hiccup, System86, Dominar Rygel XVI, WalterGR, Dekisugi, AbJ32, EMUForza, DerBorg, Childsrevolt, I created one, SF007, Darkicebot, Neuralwarp, Roxy the dog, Rror, IngerAlHaosului, Dsimic, Addbot, Harviecz, MrOllie, 3193wct, Juno677, Brutuslvr33, Aap3030, Ovpa, Jarble, Pabloignacio 02, Yobot, PMLawrence, THEN WHO WAS PHONE?, AnomieBOT, Law, Materialscientist, Xqbot, 417.417, Michaemi, Jeffwang, Amaury, Charvest, Chatul, Chaheel Riens, Catter46, FrescoBot, Pdebonte, Pinethicket, Adlerbot, RedBot, MastiBot, Unrulyevil, MAXXX-309, DASHBot, EmausBot, WikitanvirBot, AvicAWB, Ebrambot, Coasterlover1994, Romalias, Rare Akuma, ClueBot NG, Aurelio0014, Titodutta, PhomPencil, Isacdaavid, BattyBot, Mquinson, Codename Lisa, Pavel Tkachuk, Mogism, ABunnell, Jodosma, Reda cosi, Oranjelo100, Monkbot, Abc 123 def 456, AsHaFLoki, Prataps26 and Anonymous: 275

- **Snapshot (computer storage)** *Source:* [http://en.wikipedia.org/wiki/Snapshot%20\(computer%20storage\)?oldid=640025235](http://en.wikipedia.org/wiki/Snapshot%20(computer%20storage)?oldid=640025235) *Contributors:* Ghakko, Pnm, Glenn, Uzume, YUL89YYZ, Danhash, PdDemeter, Woohookitty, Lost.goblin, FlaBot, GreyCat, Chobot, RussBot, מרדכי רפאלי, Saper, Voidxor, SmackBot, Reedy, Hmains, Mairibot, Thumperward, Jerome Charles Potts, Harryboyles, NJA, Lee Carre, Cydebot, Ian Macintosh, Thijs!bot, Nick Number, LenzGr, Computer.Pers, Gwern, Stephenchou0722, Klaus Wuestefeld, Doc aberdeen, Baz whyte, Omc, Signalhead, Cjuans, Jcea, Fsmoura, Martarius, ClueBot, PixelBot, Jimmy Fleischer, MystBot, Dsimic, Addbot, RonGrevink, Luckas-bot, Yobot, FrescoBot, FGrose, Stefan Weil, Hajecate, MastiBot, EmausBot, Fairwin99, ClueBot NG, Ramaksoud2000, ChrisGualtieri, Lagoset, HaifaAlmshari, Monkbot, TerryAlex, Benmathat, Josephinebitangcor and Anonymous: 49
- **Migration (virtualization)** *Source:* [http://en.wikipedia.org/wiki/Migration%20\(virtualization\)?oldid=636167927](http://en.wikipedia.org/wiki/Migration%20(virtualization)?oldid=636167927) *Contributors:* Breakpoint, Bearcat, Mgiganteus1, YhnMzw, Dsimic, Helpful Pixie Bot, BattyBot and Anonymous: 2
- **Operating-system-level virtualization** *Source:* <http://en.wikipedia.org/wiki/Operating-system-level%20virtualization?oldid=644214561> *Contributors:* Rp, Pratyeka, Joy, Lproven, Wmahan, Eroux, RossPatterson, Demitsu, Haxwell, Ghoseb, TheParanoidOne, Jhertel, Suruena, Woohookitty, Camw, Reisio, Intr, Roboto de Ajvol, Gardar Rurak, Gaius Cornelius, Bovineone, ZacBowling, MikhailGusarov, SmackBot, OdileB, Brianski, Bluebot, Thumperward, Jsavit, Warren, A5b, Soumyasch, Leksey, LuisIbanez, Dicklyon, Jack Waugh, Quaele, UncleDouggie, Raysonho, PuerExMachina, Dream of Goats, Thijs!bot, Zapacky, K001, Electron9, JeffV, NapoliRoma, Bradmkjr, Bertl, Logснаath, Lengyeltom, Doc aberdeen, R'n'B, Nono64, Andareed, Silas S. Brown, Carla Safigan, Trasz, 2strokewool, Xanderdad, Scheibenzahl, Freevps, Blueblizzy, EwokiWiki, Echo95, Brian Geppert, The Thing That Should Not Be, Niceguyedc, SF007, Delt01, Dsimic, Addbot, KitchM, MrOllie, Simbir, Yobot, Edoe, Jlgrock, AnomieBOT, LilHelpa, Bbenson217, Terrivsm, ManolRoujinov, LucienBOT, WaldirBot, Qfissler, Expertour, Safinaskar, Divinity76, Jandalhandler, Tremaster, Timtempleton, Roadburns, Mysterytre, KumarDSukhani, MusikAnimal, Relient92, Ai148961, Camusensei, SimonBramfitt, Comp.arch, Systembelle, Alxfarr, Dough34, ScotXW, Machorbital, CaptTofo, Nelsonkam and Anonymus: 94
- **Application virtualization** *Source:* <http://en.wikipedia.org/wiki/Application%20virtualization?oldid=644001211> *Contributors:* Michael Hardy, Anders Feder, Glenn, Rchandra, Wmahan, Kevin B12, Yonkeltron, Enric Naval, Sam Korn, Musiphil, Anthony Appleyard, Rjwilmsi, Utuado, Butros, Wavelength, Bhny, Dclapp, Bovineone, Pegship, Closedmouth, Darrel francis, SmackBot, Raywood, Gribeco, JGXenite, JonHarder, RomanSpa, TastyPoutine, UncleDouggie, Cheburashka, Mmccar, Phatom87, Yukoba, Salahx, Marc@agentos.net, Dawnseeker2000, John Ericson, Typochimp, Shigdon, Magioladitis, Waacstats, Drm310, Anandcv, Vpfaz, Joewski, Tsmitty31, STBotD, Egmason, TXiKiBoT, OlavN, Dkarofsky, Phobos11, DataSynapseInc, Denisarona, Jvlock527, ClueBot, Raj.emmanuel, EoGuy, Artem-Razin, Kathleen.wright5, Socrates2008, Muro Bot, Subash.chandran007, SF007, TravisAF, Dsimic, Novemberdobby, Addbot, Mortense, MrOllie, Jasper Deng, Chris Neville-Smith, Mayankjohri, Mjquinn id, Jarble, Otidenebotsenre, EthioRussian, Cloud2kz, Jerebin, AnomieBOT, ArthurBot, Dpernar, Rakesh india, Terrivsm, U1024, Njohns78, FrescoBot, Swsmith69, Winterst, Elvinasosa, Mikemonzon, Trappist the monk, Wo.luren, MoreNet, ChronoKinetic, Jesse V., Mtg1977, DASHBot, Timtempleton, Mmeijeri, Rocketbird79, Itspragee, Larsrein, Myohai, Cruiser 001, Loosecaboose2011, MelbourneStar, Shaddim, MerllwBot, Milworker, LeelanHowse, SimonBramfitt, Cheolsoo, Smartpacker, Dave Braunschweig, Epiegenius, Melonkelon, Cedric-vincent, BPLabsWiki, Hosteddesktopguru, Virtualsage, Virtual75, Giorgio.bonuccelli, Leonwalsh78, T3adke23, Ns sai and Anonymous: 92
- **Portable application** *Source:* <http://en.wikipedia.org/wiki/Portable%20application?oldid=634261877> *Contributors:* Damian Yerrick, Haakon, Indefatigable, EpiVictor, AlistairMcMillan, Dcandeto, Enric Naval, Darwinpolice, Guy Harris, Woohookitty, Jannex, Pol098, Jwoodger, Seidenstud, Mr.Unknown, Fragglet, Borgx, Cybercat, J. M., Jengelh, SpuriousQ, C777, Rwww, SmackBot, Rmccue, Gilliam, Chris the speller, Bluebot, Thumperward, EncMstr, Haidut, EdgeOfEpsilon, Zhinker, MatthewKarlsen, Pwjb, Aldaden, Spinality, Mojochan, WalksWithGrizzlies, Eliyak, Sjlewis, Koweja, UncleDouggie, Hiogui, Chovain, TMN, AndrewHowse, Mblumber, Cgand, CritterNYC, Paddles, Nuwewesco, Thijs!bot, Salahx, Louis Waweru, AntiVandalBot, Liquid-aim-bot, DizzyTech, Watkins, Infovarius, Jerome-Jerome, Numbo3, Shawn in Montreal, STBotD, VolkovBot, TXiKiBoT, OlavN, JhsBot, Jamelan, ToePeu.bot, Antonio Lopez, CP2002, Snarkosis, ClueBot, LionsPhil, Pascal.kotte, Kaspobot, Socrates2008, Ocarre, Colormagic, 842U, BOTarate, Porta123, Moonradar, SF007, Miami33139, DumZiBoT, XLinkBot, Sean Zhu, Addbot, Elsender, Livebird, NjardarBot, MrOllie, Dayofsweden, Tide rolls, Jarble, Pbtogourou, Okg2004, CrocodileMisfit, KamikazeBot, Jerebin, Tempodivalse, AnomieBOT, Rjanag, Kingpin13, Dr Lean, ArthurBot, Padenni, Obersachsebot, Thedon808, Pillbugsammy, Martnym, Omnipaedista, Michael93555, BenzolBot, Kalki101, Hellknowz, Skyerise, Taz789, Mutek Android, LinuxAngel, Asdfsf, Diamondland, ClueBot NG, Shaddim, Lopesmaio, Bshope, HMman, Jsantesmas, LeelanQubic, DarafshBot, Loganfalco, Myconix, Cedric-vincent, Lagoset, Portablesoftwarezone, T3adke23 and Anonymous: 142

- **Memory virtualization** *Source:* <http://en.wikipedia.org/wiki/Memory%20virtualization?oldid=595710038> *Contributors:* Expatrik, Josh Parris, Frap, Hu12, UncleDoggie, Unionhawk, Neustradamus, Kubanczyk, Shar1R, VolkovBot, JL-Bot, Carl von Blixen, SchreiberBike, Bunnyhop11, PlaysWithLife, FrescoBot, X7q, Tsipi, ClueBot NG, Khiladi 2010, Codename Lisa, Kaulgaurav84, Stefmorrow, Maykurutu and Anonymous: 12
- **Storage virtualization** *Source:* <http://en.wikipedia.org/wiki/Storage%20virtualization?oldid=631264281> *Contributors:* Pnm, Michael Devore, Chowbok, Am088, Icairms, Corti, IlyaHaykinson, Cbdorsett, Jorunn, Rjwilmsi, SchuminWeb, Intgr, Wavelength, Bhny, TheDrew, SteveLoughran, Mennis, SmackBot, Reedy, Tanzelmo, JonHarder, Nakon, JzG, Kuru, Soumyasch, Dicklyon, CmdrObot, Raysonho, LCP, Kubanczyk, Nick Number, DPdH, Krmadhukar, Austinmurphy, Swpb, Catgut, Rfellows, Annsilverthorn, Peter Chastain, Johnha, Baz whyte, Clerks, Provibe, Maderiaboy, GT1956, Kbrose, Mr.jeff.shelley, Aag6z, Richardglendenning, SoleraTec, Esthon, Feline Hymnic, Gpfwestie, Aleksd, Robertinfantino, Anticipation of a New Lover's Arrival, The, Addbot, Older and ... well older, Yobot, Ozgurdogan, Tombohannon, Materialschemist, Jobutae, Xqbot, J04n, Omnipaedista, Jinxynix, Sgt Floyd Pepper, FrescoBot, Betatester7, Ntoepfer, BenzolBot, Wikidude231, No1can, Mensch0815, Anandhm, Mheskett, Amillzie, John of Reading, Fairwin99, Vinaychitti, Marcfi, Music Sorter, Bomazi, Mirosław.frank, ClueBot NG, Matthiaspaul, Ninithekhhandelwal, CanadianBeatJunkie, Widr, Helpful Pixie Bot, AussieStorBlog, CityOfSilver, Aisteco, Camberleybates, Knieriemen, Nowf and Anonymous: 105
- **Network virtualization** *Source:* <http://en.wikipedia.org/wiki/Network%20virtualization?oldid=643678125> *Contributors:* The Anome, Pigsonthewing, Michael Devore, Rchandra, Discospinster, Firsfron, Rjwilmsi, Daderot, Finell, SmackBot, Unforgettableid, Zac67, Iridescent, UncleDoggie, CmdrObot, NapoliRoma, Curtbeckmann, Stefaniab, AlexGalís, Farrahi, Drmies, Boing! said Zebedee, Vitabrevissima, Addbot, Graham.Fountain, MrOllie, Fiftyquid, Luckas-bot, Yobot, Materialschemist, Xqbot, FrescoBot, Greenboite, Mike735150, Datacenterarchitect, Shawn xie, Mbrandwin, Awat66, Ckolias, Jianhui67, Flavio.esposito.wiki, Lrongzheni, Liang0428 and Anonymous: 34
- **Software-defined networking** *Source:* <http://en.wikipedia.org/wiki/Software-defined%20networking?oldid=643668694> *Contributors:* Disdero, Joy, Jeffq, Bearcat, Denwid, Rchandra, Khalid hassani, Glaucus, Wajlee, Planetneutral, Malcolm, Jpbowen, Moucher, Abune, Back ache, Scott Dial, Thumperward, Eewanco, Derek R Bullamore, Bushsf, Kvng, Patrickwooldridge, MeekMark, Jac16888, Andyj-smith, Nick Number, ErikHedberg, Dougher, Railrulez, Curtbeckmann, CommonsDelinker, Gillwil2000, Jcasman, Vjardin, Wdwd, Mild Bill Hiccup, Shaded0, Phileasson, XLinkBot, Dsimic, SlubGlub, Addbot, Luckas-bot, Yobot, Heading, AnomieBOT, Rohit talukdar, Rotellam1, Shulini, Mwehle, Awang82, Nameless23, FrescoBot, Llib xoc, Vmlaker, Nageh, W Nowicki, JNorman704, Gjcarter, JnRouvignac, V.podzimek, EmausBot, Dewritech, Alfredo ougaowen, BobGourley, MainFrame, ClueBot NG, Dfarrell07, Helpful Pixie Bot, Profjerome, BG19bot, Alain Pannetier, Nyavatar, Morning Sunshine, Martincasado, Justincheng12345-bot, Sminiman, M1sdata, Sardella2, Marmadukehussey, Epicgenius, Oneeighteen118, Marcella Wolfe, EvergreenFir, Sirnerd, K0zka, Someone not using his real name, Qtguy00, Feamster, Seankmccloskey44, Goldavberg, Rajeevijeyta, Writ large, Bnbaxani, Daveginsburg, SsGer0710, Hkpress, Rburke2, Noqued, Sardella26, RebeccaSalie1 and Anonymous: 71
- **Network Functions Virtualization** *Source:* <http://en.wikipedia.org/wiki/Network%20Functions%20Virtualization?oldid=644288448> *Contributors:* Andrewman327, ChuunenBaka, Blaxthos, Ithinktfiam, Kinawi, Zozoulia, Optikos, Dschradler, Dougher, Theroadslong, Nkant, Download, Zirion, Yobot, AnomieBOT, Hairhorn, Kakokalo, FrescoBot, W Nowicki, Jonesey95, Harv the, A.wasylewski, Robpater, Onel5969, Pinkbeast, Timtempleton, K6ka, Bollyjeff, Bomazi, BG19bot, SongO, BattyBot, ChrisGualtieri, Codename Lisa, Epicgenius, Ckolias, Ggaron, Nars2818, Gerardo.garciadeblas, JamieCaptainWizard, Peeps136, Lawal026, Thewiggler235, Dcinnovation, Rajeevijeyta, Igcarrillo, Raysharmapcc, Mschorer, Jterminator, Sardella26 and Anonymous: 34
- **Cloud computing** *Source:* <http://en.wikipedia.org/wiki/Cloud%20computing?oldid=644529964> *Contributors:* Zundark, ChangChienFu, Heron, Jose Icaza, Jdlh, Michael Hardy, Mahjongg, Rw2, Haakon, Ronz, Julesd, Andrewman327, DJ Clayworth, Tpradbury, Furrykef, Saltine, Fvw, Dbabbitt, Tkcoach, Rossumcapek, Chealer, Lapax, Rursus, SC, Jleedev, Superm401, Tobias Bergemann, Lysy, Martinwguy, Giftlite, Metapsyche, Smjg, Graeme Bartlett, Ryanrs, HangingCurve, Mckaysalisbury, DavidLam, Utcursch, SoWhy, Pgan002, SarekOfVulcan, Beland, Bumm13, Sfskett, Xinconnu, Axelangeli, Now3d, ShortBus, Chem1, Thorwald, Mike Rosoft, Slady, Discospinster, Rich Farmbrough, Hydrox, YUL89YYZ, Bender235, ESKog, Neko-chan, Syp, Shanes, Jpgordon, Shax, Sanjiv swarup, Richi, CKlunck, Justinc, Mdd, Alansohn, Gary, Csabo, Richard Harvey, Tobyeh, Free Bear, Kessler, Diego Moya, Andrewpmk, Ashley Pomeroy, Snowolf, Wtmitchell, Velella, Wtshymanski, Stephan Leeds, RubenSchade, LFaraone, Blaxthos, Richwales, Walshga, Oleg Alexandrov, Skrewler, Stuartyeates, Brianwc, Lordfaust, Firsfron, Woohookitty, Mindmatrix, RHaworth, TheNightFly, Ruud Koot, WadeSimMiser, Trödel, Cbdorsett, GregorB, Dogsbody, SPACEBAR, Littlewild, Mandarax, SqueakBox, BD2412, Pmj, Jorunn, Rjwilmsi, Nightscream, Wootoo, Salix alba, MZMcBride, Vegaswikian, Bodhran, ElKevbo, Bubba73, The wub, Nicolas1981, FayssalF, Makru, Windchaser, Jmc, Nogburt, Crazycomputers, Jacob1044, A.K.Karthikeyan, Intgr, David H Braun (1964), Ahunt, Imnotminkus, Chobot, DVdm, Guliolopez, Wavelength, RussBot, Bhny, Stephenb, Manop, SteveLoughran, Rsrikanth05, Bovineone, Tungsten, SamJohnston, LandoSr, Gram123, NawlinWiki, Dielectric, Grafen, Welsh, Hogne, Akropp, Dethomas, PhilipC, Moe Epsilon, Tony1, Jerome Kelly, Wizzard, Jeh, Sarathc, Bike90, Yonidebest, Rolf-Peter Wille, Zzuuzz, Sissyneck, Timwayne, E Wing, Juliano, JLaTondre, DoriSmith, Allens, Katieh5584, Snaxe920, Otto ter Haar, Bernd in Japan, Liujiang, Tom Morris, Victor falk, Kimdino, DanStern, Luk, Mcaffney, Palapa, SmackBot, Ashley thomas80, JoshDuffMan, McGeddon, Gigs, PhilJackson, CastAStone, C.Fred, Elminster Aumar, Davewild, WookieInHeat, Jab843, AnOddName, Lainagier, Yamaguchi 先生, Gilliam, Ohnoitsjamie, Skizzik, Samveen, Kawana, Rmosler2100, Chris the speller, Bidgee, Ebhakt, Thumperward, Siddii, RayAYang, Deli nk, Jerome Charles Potts, Dlohcierekim's sock, Letdorf, Nbarth, Colonies Chris, A. B., John Reaves, Scwlong, Wynand.winterbach, Nabeez, Mike hayes, Tped, Frap, StefanB sv, Jacob Poon, OSborn, Uozef, Billytkid, GVNayR, LuchoX, Abrahami, Speedplane, Valenciano, Preetesh.rao, Dreadstar, Drphilharmonic, DMacks, Shswanson, Vina-iwbot, Bejnar, Vasilij Faronov, Spiritia, KenCavallon, Acrooney, ArglebargleIV, AbdullahHaydar, Harryboyles, Gandalf44, JzG, Kuru, Oskilian, Tomhubbard, Gobonobo, Darktemplar, Robofish, JoshuaZ, Kashmiri, Minna Sora no Shita, IronGargoyle, Ckatz, Kompere, Beetstra, Mr Stephen, Ehheh, Larrymcp, Optakeover, Wagers, TastyPoutine, Dr.K., Kvng, Belfry, Keahapana, Hu12, Meitar, Quaeler, Sp0nman, Jonasalmeida, IvanLanin, UncleDoggie, Rnb, Mjboniface, Courcelles, Dlohcierekim, Chris55, Patrickwooldridge, FatalError, JForget, VoxLuna. Ourhistory153, Randhirreddy, Earthlyreason, Eric, JohnCD, Bill.albing, Kmmsgill, NaBUru38, Flood6, Sanspeur, WeisheitSuchen, Alexamies, Myasuda, Metatinaro, Jehfes, Rotiro, Yaris678, Cydebot, Mblumber, MC10, UncleBubba, Anthonycole, GRevolution824, Dancter, Clovis Sangrail, Christian75, Ameliorate!, Kozuch, Neustradamus, Casliber, Malleus Fatuorum, Thijs!bot, Epbr123, Kubanczyk, Dschradler, Wikid77, Vicwest, Shoabnz, Ugarit, Vondruska, Vertium, John254, James086, Edchi, EdJohnston, Nick Number, Bob.gourley@comcast.net, Heroeswithmetaphors, Tree Hugger, Dawnseeker2000, Escarbot, Porqin, MrMarmite, Seaphoto, Shirt58, Marokwitz, Smartse, Dinferno, Silver seren, MrKG, Lbecque, DaudSharif, Tangurena, Dougher, Barek, MER-C, Dsp13, Jldupont, MB1972, Mwarren us, Rms77, Ispabierto, Greensburger, East718, Ny156uk, Spojrznie, Magioladitis, Swikid, Bongwarrior, Lmbhull, JamesBWatson, Mathematrucker, GaryGo, Steven Walling, ForthOK, Jeffsnox, Hamiltonstone, Be-nice-), Pleft, Kibbled bits, Cpl Syx, Balaarjunan, SBunce, JaGa, Kgfleischmann, Philg88, Pikolas, Zevnik, Curtbeckmann, Pisapatis, Dezrtluver, CliffC, Iamthenewno2, Casieg, CitizenB, Parveson, Jack007, Xiler, Bus stop, Vermitt, Miguelcaldas, Alankc, Mariolina, Linuxbabu,

JonathonReinhart, Tgeairn, J.delanoy, PCock, Trusilver, Anandcv, Vpfaiz, Uncle Dick, Maurice Carbonaro, Jesant13, Ginsengbomb, Mathglot, Jarrad Lewis, Tsmitty31, Betswiki, Tonyshan, Staceyeschneider, NewEnglandYankee, Quantling, BostonRed, Biglovinb, Oleg-wiki, Bshende, KylieTastic, Raspalchima, HenryLarsen, Paulmmn, Songjin, Bonadea, Pegordon, Swollfs, Idioma-bot, Laurenced, Martin.ashcroft, Intiyazali4all, Bobwhitten, Obdurodon, Huygens 25, Vranak, 28bytes, VolkovBot, Jeff G., Dogbertwp, Edeskonline, Bkengland, Priyo123, FatUglyJo, Nylo, Philip Trueman, A.Ward, TXiKiBoT, Itangalo, Vipinhari, Technopat, Guillaume2303, Anonymous Dissident, Danielchalef, Markus95, Markfetherolf, GcSwRhIc, Monkey Bounce, Piperh, Rich Janis, Felipebm, Martin451, Broadbot, Willit63, Amog, Figureskatingfan, Everything counts, SpecMode, Johnpltsui, Andy Dingley, Fingall, Haseo9999, Lamro, Garima.rai30, The Seventh Taylor, Falcon8765, VanishedUserABC, Nelliejellynoona, Sapenov, LittleBenW, Jimmi Hugh, Logan, OsamaK, Biscuitin, SieBot, Skyrail, Moonriddengirl, EwokiWiki, Doctorfree, Sakaal, Dawn Bard, Navywings, Yintan, SuzanneIAM, Kpalsson, Jerryobject, Fishtron, Keilana, Chmyr, Heyitscory, Bentogoa, Happysailor, Flyer22, Jojalozzo, Nopetro, Snideology, Yerpo, Reservoirhill, OsamaBinLogin, Dominik92, Xe7al, North wiki, Techman224, Vykk, Rosiestep, Fuddle, StaticGull, Classivertsen, Bijoyr, WikiLaurent, Superbeecat, Laser813, Shinerunner, Denisarona, Motyka, Dlrhrer2003, Martarius, Simonmartin74, Elassist, ClueBot, GorillaWarfare, Wasami007, The Thing That Should Not Be, Cdhkmmaes, Nnemo, Czarkoff, Axorc, Jasapir, Drmies, VQuakr, Mild Bill Hiccup, Myokobill, Allenmwn, Enc1234, LizardJr8, Bob bobato, Darren uk, Esthon, Auntof6, 718 Bot, Pointillist, Jonathan.robie, Loadbang, Stuart.clayton.22, Ktr101, Excirial, Pumpmeup, Alexbot, Jusdafax, Sajeer50, Hfoxwell, Eeekster, Nasonmedia, Muhandes, SunnySide-OfStreet, Technobadger, 842U, Cmartell, M.O.X, Razorflame, Jinlye, SchreiberBike, Five-toed-sloth, Craig.Coward, Jakemoilanen, Vdmeraj, PCHS-NJROTC, Johnuniq, Vigilus, DumZiBoT, Jack Bauer00, Steveozone, Darkicebot, Beltman R., Lordduvengan, XLinkBot, AgnosticPreachersKid, Roxy the dog, Njkool, Stickee, Sponson, Feinoha, Chanakal, Bpgriner, C. A. Russell, Avoided, Fergus Cloughley, Imllorente, Skarebot, WikHead, Galzigler, Mifter, PcCoffee, Jbeans, Eleven even, Jht4060, NonNobisSolum, Richard.McGuire88, Sandipk singh, RealWorldExperience, Y212, Edepa, B Fizz, Dbrisinda, Deineka, Bazj, Addbot, American Eagle, TimFreeman701, Ramu50, Mortense, Realtimer, Sean R Fox, Mabdul, IXavier, VijayKrishnaPV, Fcalculators, Mkdonqui, Amore proprio, Tanhabot, Barmijo, TutterMouse, Fieldday-sunday, Scientus, Shakeelrashed, CanadianLinuxUser, Ethoslight, Kristiewells, Cst17, Mohamed Magdy, MrOllie, Download, Robert.Harker, Hatfields, Glane23, Mhdapp, Glass Sword, JimDelRossi, Favonian, Ohtatus, Stbrodie1, Terrillj, Numbo3-bot, Superkillball, Cybercool10, HandThatFeeds, Ashleymcneff, Tide rolls, אונו, Avono, NeD80, NonNobisSolum, Richard.McGuire88, Teles, Cloudcoder, Jarble, Mlavannis, Shri ram r, HerculeBot, Enaiburg, Gamber34, Legobot, Avlnet, Jerichochoang97, Luckas-bot, BaldPark, ZX81, Yobot, Evagarfer, Themfromspace, Dfxdeimos, Legobot II, Librsh, Jamalystic, Bruce404, Asieo, Indigokk, Reshadipoor, Washburnmav, Identity20, Adam Hauner, Imeson, Javaeu, Thesurfup, Achimew, Lerichard, Knoxi171, ByM4k5, Tiburondude, Aburrenson, Jean.julius, Sweerek, Peter Flass, Sql er2, WikiScrubber, Sivanesh, IANYL, Deicool, AnomieBOT, Momoricks, Dmichaud, Pgj1997, Iexec1, Cronos4d, ThaddeusB, Jim1138, IHSscj, JackieBot, Iamdavinci, CloudComputing, Yaraman, Mbblake, AdityaTandon, Csi-gabi, Felixchu, MaterialsScientist, RobertEves92, JamesLWilliams2010, The High Fin Sperm Whale, Citation bot, Jkelley, OllieFury, Shan.rajad23, ArthurBot, Quebec99, YoungManBlues, NW's Public Sock, PavelSolin, LemonairePaides, Mwmmaxey, Xqbot, L200817s, Alexlange, Lairdp, Avnralgom, Capricorn42, Rakesh india, Surajpandey10, Pontificalibus, Nfr-Maat, Nasmena, Poliverach, Gkronlad, Ohspite, Ramnathkc, Wlouth, Tatateme, Chadastrophic, Dbake, NFD9001, Emrekenci, Anna Frodesiak, Explorer09, Brian Wren, Peterduffell, Macholl, Anamika.search, EricTheRed20, Michael.owen4, MarkCPhinn, NocturneNoir, Miym, J04n, GrouchoBot, Onmytoes4eva, Frosted14, Popenose, ProtectionTaggingBot, Rsiddharth, Omnipaedista, Andyg511, DarrenOP, RibotBOT, Mattg82, TonyHagale, Jbekuykendall, Jw1801, Mathonius, Yodaspirine, Vmops, WootFTW, Liyf, Mobilecloud, Figaronline, BrennaElise, Shadowjams, E0steven, Chaheel Riens, Jef4444, Person1988, A.amitikumar, Dan6hell66, RetiredWikipedian789, Mmanic, FrescoBot, Intiyaz 81, Adlawlor, Yuchan0803, Zachnh, Manusnake, Blackguard SF, Cajetan da kid, Paj mcarthy, Ronen.hamias, Mark Renier, CloudBot, Sariman, Jakeburns99, Jesse.gibbs.elastra, W Nowicki, Estahl, Pottersson, FreddyMay, Recognizance, Nakakapagpabagabag, MichealH, Gratridge, Ashakeri3596, Ummelgroup, Sebastiangarth, HJ Mitchell, Pete maloney, Scott A Herbert, Zhanghaisu, Berny68, Wireless Keyboard, HamburgerRadio, Yinchunxiang, Lmp90, Rickyphyllis, Acandus, Vasq0123, Winterst, Monkeyfunforidiots, Pinethicket, I dream of horses, Elockid, HROestBot, Samuraiguy, Jonesy95, Eddiev11, AcuteSys, MJ94, PMstdnt, JLRdperson, Li Yue DePaul, Tinton5, Skyerise, Ggafraser, A8UDI, Dannyjohnson, Nimbus prof, RedBot, Janniscui, Manishkshawat, Bigfella1123, SpaceFlight89, Aneesh1981, Troy.frericks, Σ, Natishalom, Agemoi, Piandcompany, Noisalt, Cloudnit, Jandalhandler, Devinek22, Undiscovered778, Nolcan, Ras67, Maasmiley, Abligh, Reconsider the static, AstralWiki, Juliashapiro, SW3 5DL, Niri.M, Msileinad, Hughesjp, Skovigo, Kjohnthomas, Jburns2009, JanaGanesan, ConcernedVancouverite, Trappist the monk, Declan Clam, Iminvinciblekris, SchreyP, Rajeshkamboj, Avermapub, KotetsuKat, Sanmurugesan, Markus tauber, Burrows01, Lotje, Kieransimkin, EDC5370, Dinamik-bot, Vrenator, Danielrs, LilyKitty, Richramos, Clarkcj12, Robscheele, SeoMac, Miracle Pen, Çalıştay, Ansumang, Ycagen, Aoidh, Eco30, Reaper Eternal, Crysb, Whitehouseseo, Info20072009, Jeffrd10, Pmell, Innz730, Cemgurkok, Suffusion of Yellow, Taicl13, Tbhotch, Latha as, Colleenhaskett, Balvord, Hutch8700, MarshallWilensky, Nesjo, DARTH SIDIOUS 2, OmbudsTech, MidgleyC, Mean as custard, Sumdeus, RjwilmsiBot, Veerapatr.k, TjBot, DexDor, Alph Bot, Tdp610, Jon.ssss, Nasserjune, Amartya71, VernoWhitney, Darkyeffect, Chriswriting, Ypolavina, Beleg Täl, Ianmolynz, DuineSidhe, DSP-user, Learn2009, Aurifier, Jineshvara, Mmorinreliablenh, Bantelim, Adjectivore, Lipun4u, R39132, Nookncranny, J36miles, EmausBot, Editorfry, Understated1, Acather96, Pjposullivan, Deepalja, WikitanvirBot, Gozzilli, Rutger72, PaulQSalisbury, Logical Cowboy, Timtempleton, Eusbls, DragonflyReloaded, Macecanuck, Ajraddatz, Heracles31, Noloader, Jbadger88, Dewritech, Clutterpad, Ianlovesgolf, Gmm24, GoingBatty, RA0808, Snydeq, AoV2, Vanished user zq46pw21, Tisane, Sp33dyphil, Fzk chc, Luigi Baldassari, Solarra, Njcaus, Moswento, Wikipelli, 623des, K6ka, Tmguru, AsceticRose, Srastogis, Anirudh Emami, Zafar142003, Thecheesykid, Francis.w.usher, Hardduck, Vishwaraj.anand00, QuentinUK, Manasprakash79, Jtschoonhoven, BobGourley, Bongoramsey, Fæ, Josve05a, Deleob, Ppachwadkar, Stevenro, Aaditya025, Trinitade, Wackywace, 9Engine, TunaFreeDolphin, Tom6, Kronostar, Wtsao, Aavindraa, A930913, GZ-Bot, H3llBot, Chintan4u, Eniagrom, Amanisdude, Kyerussell, Machilde, Mr little irish, Tolly4bolly, Jay-Sebastos, Thexman20, Coasterlover1994, Scott.somohano, L Kensington, Ready, Mayur, Aflagg99, Donner60, Skumarvimal, Zfish118, MillinKilli, Puffin, Ego White Tray, Orange Suede Sofa, Rangoon11, Bill william compton, MainFrame, JohnnyJohnny, StartupMonkey, ClamDip, Kenny Strawn, MaxedoutnJJITWILL, Alex5678, Msfitzgibbonsaz, Shajulin, Nurnware, EmilyEgnyte, DASHBotAV, NatterJames, Jhodge88, JohnJamesWilson, 28bot, JonRichfield, Frozen Wind, Petr, RS-HKG, Dyepoy05, ClueBot NG, Sharktopus, Horoporo, Michaelmas1957, DellTechWebGuy, Jack Greenmaven, Slwri, VicoSystems1, Businesstecho, DrFurey, Dadomusic, Cloudcto, Amoebat, MelbourneStar, This lousy T-shirt, Qarakesek, CPieras, Satellizer, A520, Markqu, Bulldog73, Kkh03, Ethicalhackerin2010, Bped1985, Stickyboi, Candace Gillhoolley, Happyinmaine, Leifns, Hemapi, Lord Roem, Gxela, Nikoschance, The Master of Mayhem, Shawnconaway, Einasmadi, Qusayfadhel, Certitude1, Tylerskf, PJH4-NJITWILL, IDrivebackup, Lionheartf, O.Koslowski, Kevin Gorman, Dimos2k, ScottSteiner, Helotrydotorg, Alanmonteith, Digestor81, Widr, Scot-tonsocks, WikiPuppies, Gawali.jitesh, Andersoniooi, Rebuer, Chuahscy, Carl presscott, Qconcept, Keynoteworld, Fearlessafraid, Johananl, Helpful Pixie Bot, OAnimosity, Leoinospace, Iste Praetor, HMSSolent, Tastic007, Titodutta, Calabe1992, DBigXray, Lavanyalava, Aditya.smn, Whitehatpeople, Elauminri, Angrywikiuser, BG19bot, RLSnow, SocialRadiusOly, MikeGeldens, Oluropo, Krenair, Cloudx-tech, Sfteditor, ValStepanova, Joerajeev, Robert.ohaver, Cornelius383, Dpsd 02011, Jimsimwiki, Markoo3, Rijnatwiki, Bhargavee, Softdevusa, Northamerica1000, Nadeemamin, Jayvd, San2011, Shaysom09, Lee.kinsler, GhostModern, Om23, Panoramak, Avillaba,

Hallows AG, Wiki13, Stevictor134, MusikAnimal, Frze, Er.madnet, BDavis27, TylerFarell, Itzkishor, Mark Arsten, Compfreak7, Kiranani, EEldridge1, Leinsterboy, Dmcelroy1, Philopappos86, ThomasTrappler, StrategicBlue, Joydeep, Anne.naimoli, Jbucket, JM-Jeditor, Blvrao, Mychalmccabe, JDC321, Watal, Latticelattuce, Wretman, Frdfrm, Phil.gagner, Elasticprovisioner, DPL bot, Andromani, Tramen12, Torturedgenius, Phazoni, Chapmagr, Jmillerfamily, TJK4114, Subbumv, Dinhtaihoang, Charvoworld, Mpcirba, Smileyrainger, Kilidiplomus, Ssabihahmed, Achowat, Wannabemodel, ChambersDon, Fylbecatulous, BrianWo, Knodir, EricEnfermero, JoeBulsak, BattyBot, 21hattongardens, Eduardofeld, Kridjsss, Dlowenberger, 1337H4XX0R, Kunitakako, Elbarcino, ChannelFan, Haroldpolo, Cloudfest, Teammm, Xena77, Anhtrobote, Pratyya Ghosh, Tuwa, Mdann52, D rousslan, MPSTOR, Pea.hamilton, Crackers-peanut12, Mrt3366, Cloudfest, ChrisGualtieri, LarryEFast, Jackoboss, Valentina Ochoa, Beanilkumar, Mediran, EliyahuStern, Beer2beer, EuroCarGT, Prodirus, Fb2ts, SimonBramfitt, Xxyt65r, Mheikkurinen, Nithdaleman, Jags707, EagerToddler39, Davidgom, Padmaja cool, Zingophalitis, Zeeyanwiki, Mcsantacaterina, Shierro, Weterunni, Webclient101, Raushan Shahi, SJames1, Mogism, Gotocloud, Derekvicente, Nozomimous, Anderson, Cerabot, Chishtigarana, Lone boatman, Fabrice Florin (WMF), PonnagalMalar, Naturelover007, TwoTwoHello, Thewebartists013, TechyOne, Aloak1, Vikas gupta70, Arnavrox, Frosty, SFK2, MartinMichlmayr, Sk8trboi199, Os connect, Jamesj12345, Shubhi choudhary, Joe1689, Sriharsh1234, Viralshah0704, Millycylan, Lorenrb, Kevin12xd, Choir monster, Drjoseph7, BurritoBazooka, OSRules, Waynej6, Avacam, Khan.sharique1994, Amitgupta2792, Zimzaman, Faizan, Chiefspartan, Epicgenius, Shivalikisam, FallingGravity, Ruinjames, Ramanrawal, Acaliguirán, Vanamonde93, S.sameeraman, JaredRClemence, I am One of Many, John-readyspace, Whitecanvas, Jlamus, Manishrai tester, FunkyMonk101, Carrot Lord, Jp4gs, Melonkelon, Mangai Vellingiri, Lena322, Craig developer, DanielJohnc, Alf32, Solomon35, Rkocher, 5c0tt-noe, Thinkcd, Lsteinb, Tentinator, Marinae93, Aaronito, Evan1van, Kapils1255, Olynickjeff, Marcio10Luiz, Cookingwithrye, Jopgro, Fsandlinux, Backendgaming, Sanderw1, Nextlevelwb, Maura Driscoll, Flat Out, Halkemp, Couth, Dreamfigure, Murus, Saqibazmat, Jugaste, Babitaarora, Bloonstdfan360, Comp.arch, Kewi69, Metadox, Jbrucb, Podger.the, Henhuawang, Katepressed, Tbilisi2013, Asarada, Rzicari, AcidBlob, Rodie151, Fatdan786, Max1685, Mariatim, Ginsuloft, ArmitageAmy, Didi.hristova, IMMS, Insomniac14, Corey Rose, Acalcine, Jackmcbarn, Dudewhereismybike, PracticalScrum, TDBA, Bob Staggert, Rkanojia, Pcpded, Jora8488, Deepak1300, Harshac89, Goofette, WikiJuggernaut, Gracecheung08, Cloud guru28, Lucy1982, CloudBuster, PierreCoyne, Sweetadamali, LookToLuke, Justuj, Mareep, Wlwell67833, JaconaFrere, Theworm4321, JenniferAndy, Skr15081997, Rax sa, Ssgmu55, Nclenem2, 7Sidz, Sofia Lucifairy, Abhinavgupta007, Sfroberts, Musabhai2, Ajabak, Edwardsmith285, Nyashinski, MtthwAndrn, Shanhuayang, Nassaraf, Amenychtas, Uk1211, Deepika Sreerama, Monkbot, Cjbinwin, Allanamiller, Sylvesta101, Lucylloo10, Kalpesh radadiya, Dansullivanpdx, Bingoarunprasath, BethNaught, Security.successfactors, Jblews, Mboxell, Mannanse, MorePix, ThatWriterBloke, Ipsrsolutions, JoelAaronSeely, Biblioworm, Science.Warrior, Daniel.moldovan, Schwarrtz, Gk9999, Northbridge Secure, Twoosh, Cloudwizard, Thandi moyo, Wrightandru, Ferozahmed0382, Mkchendil, Tony-smith2014, Manjaribalu, Ramvalleru, Daniela C DeMaria, Brandon Connor, Andy7809, Chicodoodoo, MONISHA GEORGE, MVMeena, Garywfcchan, Sahit1109, Tweeks-va, Ss.jarvisboyle, Cubexsweatherly, Stephenzhang.cs, Srinivas.dgm, Dtechinspiration, Amagi82, Lalith269, Headinthecloud, Shantan 1995, Edavinmccoy, Zellabox, Abcdudtc, Enkakad, Nei Wg Khang, Selvarajrajakanna, Naveenwilder, Alex-smith22 and Anonymous: 2456

- Elasticity (cloud computing)** *Source:* [http://en.wikipedia.org/wiki/Elasticity%20\(cloud%20computing\)?oldid=640777329](http://en.wikipedia.org/wiki/Elasticity%20(cloud%20computing)?oldid=640777329) *Contributors:* Herostratus, SchreiberBike, Cristiklein, Yobot, MenoBot II, John of Reading, ChrisGualtieri, Kmzayeeem, Amenychtas, Monkbot, Daniel.moldovan and Anonymous: 3
- Platform as a service** *Source:* <http://en.wikipedia.org/wiki/Platform%20as%20a%20service?oldid=644433086> *Contributors:* Michael Hardy, Kku, Ihcoyc, Reddi, Joy, Nurg, Wjhnson, Discospinster, Scullder, Shuveb, Rd232, Dennis Bratland, Woohookitty, RHaworth, Dm, Chobot, RadioFan, SteveLoughran, SamJohnston, NawlinWiki, Kawika, Zzuuzz, DoriSmith, SmackBot, Dlohcierekim's sock, Frap, Jbbdude, Cybercobra, Marc-André Aßbrock, Jeremyb, Avibrazil, Camilo Sanchez, TastyPoutine, Gpierre, Dlohcierekim, FatalError, Bill.albing, Senseijack, LeoHeska, Dancter, Alaibot, Srmoon, Sodabottle, Mkdw, Sureshsambandam, J.delanoy, Antonioavmelo, Jdechambeau, Crunky, Vincent Lextrait, Shashibg, RonaldDuncan, LittleBenW, Scotty Wong, MJaggard, Toddst1, Jojalozzo, User5910, Ckeene, Fadesga, Czarkoff, Shkalantar, Jasapir, JP.Martin-Flatin, Alexbot, Dwkitchen, Msorsaleslie, Muro Bot, Djszot, Johnniq, Apparition11, DumZiBoT, XLinkBot, SilvononBot, Addbot, Ramu50, MrOllie, LaaknorBot, Favonian, Sharmagaurav03, Friarminor, Nicoosuna, Contrem, Editor0815, Middayexpress, Drnic, Yobot, Evagarfer, Ptbodygourou, Alagwiki, Javaeu, AnomieBOT, Jim1138, AdityaTandon, SunKing2, MaterialsScientist, JamesLWilliams2010, YoungManBlues, Metrisoft, Drilnoth, Scottlyons65, Dbake, Schultzter, Samantha datasyt, Achrris, Rkennedy593, Mlfung, Roberts jacobs, Jakeburns99, JBurns2009, DrilBot, Ivanlinhares, MastiBot, Adammmwanda, Juliashapiro, Inlandmamba, Fcaesar, Franciscosouza, Tbhotch, Balvord, Judglz, Lopifalko, EmausBot, Tuankiet65, Gmm24, Jeffdlane, ZéroBot, 9Engine, Makecat, ClueBot NG, BenAriAtMicrosoft, Vswapnel, Andypiperuk, Bernie44, Dimos2k, Rezabot, Kipsteele, Fearlessafraid, Mr.TAMER.Slash, Hallows AG, Sanju270, Colonel a4, BattyBot, D rousslan, ChrisGualtieri, Nixfu, Lemeb, Piyushdan, Rjimenezperis, Bastichelaar, Uri1803, SJames1, Usicecool, Piarshed, Os connect, Tentinator, Arun.Mariappan, Jakec, HCK12345, Clemensberndt, Eeepex, Timhanley, Lcarvalho1, RogueSly, Grankvist, RiskNerd, Amnorge, PatDougRo, Gracecheung08, Amenychtas, Ztpztp, Kawaja, Johnmathon, Pasindur, DGMASST, Ad120687, Fmlemos, Jbeck695, Cpaumelle and Anonymous: 202
- Software as a service** *Source:* <http://en.wikipedia.org/wiki/Software%20as%20a%20service?oldid=643608692> *Contributors:* Damian Yerrick, Deb, Ant, Dzof, Edward, Michael Hardy, Kku, Sheldon Rampton, Skysmith, Rw2, Ronz, Julesd, Chuunen Baka, Paul W, Robot, Altenmann, Nurg, Rossgk, Jasenlee, Everyking, Zinnmann, SarekOfVulcan, ShakataGaNai, OwenBlacker, Bryanlharris, Ukexpat, Freakofnurture, Discospinster, Rich Farmbrough, ESkog, S.K., Barcelova, AJP, Cretog8, John Vandenberg, Sanjiv swarup, I9Q79oL78KiL0QTFHgc, Espoo, Gary, Ixnay, Jezmck, Ashley Pomeroy, Gurulegend, Velella, Stephan Leeds, RainbowOfLight, Kenyob, Martiniano, Woohookitty, RHaworth, Andrewspencer, Grika, Mandarax, Trogl, Reisio, Rjwilmsi, Pjetter, Davidp, Lordsatri, Tawker, Bodhran, Durin, Sleepy-head81, Fred Bradstadt, Crazycomputers, Riki, Weaselville, Pkg2, Gurubrahma, Chobot, DVdm, Pinecar, Wavelength, Seahcj, WAVEgetarian, Lar, RadioFan, SteveLoughran, SamJohnston, Bachrach44, Dipskinny, Nick, Davemck, Grafikm fr, Sleepnomore, Alpha 4615, Zzuuzz, Maxis4132, Closedmouth, AnimeJanai, ViperSnake151, Samuel Blanning, SmackBot, Royalguard11, Gigs, Prenagha, Kyliptix, Arunpc, Gilliam, Ohnoitsjamie, Carl.bunderson, Delinck, Neo-Jay, Behaafarid, Amy Crescenzo, Xkra, ESteege, Ctbold, Renewolf, Frap, Pkchan, JonHarder, AlejandroCh, Benjamin Mako Hill, Rgill, Nonforma, Emetamktg, KnowBuddy, Scjnsn, Pgillman, Swatjester, Smithh, JzG, Kuru, Forestfighting, Axelschultze, Bilby, JHunterJ, Beetstra, Sireland, TastyPoutine, Shinie21, Hu12, Pjrm, Stephen B Streater, Es peer, Clarityfiend, Eggen, IvanLanin, Mudgen, UncleDouggy, Disaas, Eilon.reshef, Bwv582, FleetCommand, VoxLuna, Ahy1, Wafulz, Stansult, Donaldpugh, Richard Botley, Pmerson, Scribblerman, Alpha0, Michael J. Mullany, Jumpfightgo, Cydebot, Lechevre, Jmleicht, Jasonbradfield, Adamg81, Ern queensu, Pascal.Tesson, Tawkerbot4, DickeySingh, DumbBOT, MSoares, Optimist on the run, Omnesoft, Ranaweeram, Epbr123, Kubanczyk, Insano70, Douglas Michael Massing, Fred.luddy, Hagins, JustAGal, Ramckay, RichardVeryard, Adobe unix, AntiVandalBot, RobotG, Gioto, QuiteUnusual, Matt Kaar, Franktsh, Djaipi, Fayenatic london, Krindels, Manjob, Darklilac, Hillermike, Gay Cdn, Nimtronican, Pipeline nick, Anne.davis, SiobhanHansa, Ntura, Bouktin, Decster, Aclfact, Jerowtf, Jontreese, Nyq, Kafeel, Ishi Gustaedr, Leighton1, David Terrar, Sodabottle, Jvhertum, Texxs, KConWiki, Steevm, Olivercorlett, Erpbox, Tmcnamee, Peters72, User A1, CSWL, Heereb, Gzoo, TAG2010, DerHexer, Kgfleischmann, Khobi, Chseto, Acroll, DRogers,

Downingtown, Khigs, Jim.henderson, Mharrist, Alro, Jackbower, Nima1981, Ddalton221, Tallik, Tseabourn, J.delanoy, Davidmcarson, Trusilver, Eurekasoft, Flandis, GarryLowther, Kshaikh1977, Nehurd, Herbprooy, Hightowe, Aerobe, Rblankens, Stephanwehner, Jpiekarz, 4johnny, Alexei potialgov, McDScott, Proactivity, LordAnubisBOT, Ryan Postlethwaite, Ericbussy, Bs4173, Detruedell, TerenceChia, Ermannonifazi, Peterjcooper, HoffmanPR, Rumpelstiltskin223, Jtinnerello, SaraPaceVincent, Mlegs, Bonadea, NRN4, Bolz46, Tzetsy, Deltaonline, Idioma-bot, Imtiyazali4all, Tazz2, Mooncaine, VolkovBot, Lrbianco, Olofmalmof, Ditmars, Saasglobal, Jsharp111, Taptamus, A.Ward, Esotericengineer, Rei-bot, Nduplan, Olafeldkamp, Figureskatingfan, BotKung, Jclamageran, BruceJudson, JasonLeeHansen, Johnpltsui, Rambo iyer, Garima.rai30, TronnaRob, Ebarbero, Johnnylm, LittleBenW, Daveh4h, Biscuitin, SieBot, Sushantmadhab, Chelseadude, YonaBot, BotMultichill, VVVBot, Zachpresnall, Caltas, AIMSzpc, Derekaiton, Flyer22, Jojalozzo, Oxy-moron83, Patoday, ERSACS, Aboluay, Rosiestep, ClueBot, Ron8hu, Stephen.andrew.lynch, Leongkahmun, Queenbeecooper, LawrenceCohen, RobBertholf, NovaDog, PRLady, Applicationit, Dhulme, Rockfang, Incapearl, Pointillist, RicharHMorgan, Somno, Aua, Copy-editor42, Alexbot, Socrates2008, Friendlydata, PixelBot, Steveg-in-boston, Acas7i11o, Razorflame, 1ForTheMoney, Victor McGuire, Mauten, Johnniq, CorpITGuy, Apparition11, DumZiBoT, Xsalia, Springcm, XLinkBot, Fastily, DGRO, Spector9, Richard.McGuire88, Kevbo11, Shakermover, Deineka, Addbot, Shayer, Pigr8, Spfanstiel, Lanesh, Kaitlin510, Earthboundtiger, Heanderson, Hwright001, GregKirch1, Amorrise, MrOllie, Favonian, SpBot, Williamglasby, Abytek, Terrilja, Texasanm, Evildeathmath, Tide rolls, Niccoosuna, Jarble, Matthieu.toulotte, Josephalter, Jmm78, Aiden McShane, Ben Ben, Fuadahasan, Luckas-bot, Yobot, Evagarfer, Bunnyhop11, TaBOT-zerem, Bodiam, Alusayman, Nallimbot, Jerebin, Fruehwirth, Jkgibbs, Look Sharp!, Jafreund, Spin IT, AnomieBOT, Jmeiers, Piano non troppo, Kingpin13, TParis, Jagiaz, AdityaTandon, Toko50, Ericcub7, Markcowan, JamesLWilliams2010, GB fan, ArthurBot, Quebec99, YoungManBlues, Metrisoft, PavelSolin, Xqbot, Capricorn42, Inteqcorp, Jcboss0440, Ruthstark, Aplixus, Jmundo, Tyrol5, Omnipaedista, Codex24, Sophus Bie, Mart becs, Figaronline, Jnlinn, Benjaminirvine, Shilpakaluti, Dougofborg, Zagyg, FrescoBot, Regional Marketing, Guerchoig, Techauthor, Mark Renier, Cgrunkemeyer, Plukyanchuk, Awhitney404, Haeinous, SaaSpioneer, Aminmalik, Vilva.moorthy, Aspardeshi, Dlee8888, Victortc, Jonesey95, Benkepess, JLRedperson, RedBot, MastiBot, Amazingtrails, Jandalhandler, Juliashapiro, Vinaysingla, Trappist the monk, Mns556, Newt Winkler, Mys 721tx, Jonkerz, Lotje, Fox Wilson, Vrenator, Jsanalyst, Mdmatey, Roman Doroshenko, Aoidh, David Hedlund, Crysb, Yeng-Wang-Yeh, Hobbes Goodyear, RjwilmsiBot, Ripchip Bot, Beleg Täl, Learn2009, Beerparrot, Aservices, Greenopedia, EmausBot, Emmess2005, WikitanvirBot, Higorrg, Super48paul, Gmm24, Upgrades software, Nicole.Hayward, Kkm010, Compcrawler, Jahub, Lutehawks, Rallu pm, Cobaltcigs, Bamyers99, H3llBot, Jnaranjo86, SporkBot, NickGagalis, Frankvanzanten, Ivanarj, Palosirkka, Wtferguson, Andreas111, Murgatroyd49, Bomazi, Casey Armstrong, Rkunaraju86, Lightspeedx, Sepersann, Pandoraholic, Ddulniak, Juanparve, ClueBot NG, Verbamundi, Dvivilkinsdvvilkins, Sebastian Scheuer, Admads, Satellizer, Happyinmaine, Merely curious, JoseLucassantos, Ffree, SilverSoul91911, Fedeman1977, BlackTigress, Helpful Pixie Bot, Lowercase sigmabot, Megan novalys, BG19bot, QAD MFuller, Vagobot, Northamerica1000, MusikAnimal, Falkirks, Leofoward, Abhishekjain.ims, Herwin.a, Zackmann08, Colonel a4, BattyBot, ChannelFan, Ajaijsps, Evermuse, Prelude after noon, Cloudymorning, Nmarek8, Zendooodles, Codename Lisa, Razulfellawen, Chauguleojas, Neeraj.viman, Ashy0107, Qbtarzan, Epicgenius, Smu edu, Jlew2871, Whitecanvas, Coquee, LibraryRattle, Aadrucumbrod, Dartoo, Heath1124, Delarrendale, Tbilisi2013, Kózka, Ldms020255, Samboo, Christian Lizardo Aligo, TylerGreenberg, Redrussell1980, Mattster42, JaconaFrere, Luckeck, Joshgarofalo, Editor0071, MithwAndrn, Ztpztp, 4rajmisra, Jasham, Andy Bahaquote, Lucasdealmeidasm, Nima1353, Skemptastic, Daptiv101, Prisoner62113 and Anonymous: 659

- **Cloud computing issues** *Source:* <http://en.wikipedia.org/wiki/Cloud%20computing%20issues?oldid=641633743> *Contributors:* SamJohnston, McGeddon, Cnwilliams, BG19bot, Amenychtas, Abcdudtc and Anonymous: 2
- **Cloud computing comparison** *Source:* <http://en.wikipedia.org/wiki/Cloud%20computing%20comparison?oldid=644509181> *Contributors:* Danhash, Tim@, Racklever, Kvng, Cydebot, ThatPeskyCommoner, Andyjsmith, Lathama, Magioladitis, Shar1R, Kozka, WKCole, Pointillist, SchreiberBike, MuZemike, Ganesh.rao, Amhytti, PabloCastellano, Alapolloni, Alvin Seville, FrescoBot, Txt.file, Davemccrory, Donner60, ClueBot NG, A520, Helpful Pixie Bot, Northamerica1000, Jourdrn, Carl.schutte, Piergiorgio.venuti, Pmgenglund, Turknix, Mikekx, Lindsaywest, Number0five, ArmitageAmy, Monkbot and Anonymous: 35

23.8.2 Images

- **File:Ambox_important.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:Ambox_question.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1b/Ambox_question.svg *License:* Public domain *Contributors:* Based on Image:Ambox important.svg *Original artist:* Mysid, Dsmurat, penubag
- **File:Ambox_scales.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/5c/Ambox_scales.svg *License:* Public domain *Contributors:* self-made using inkscape and based off of Image:Emblem-scales.svg *Original artist:* penubag and Tkgd2007 (scales image)
- **File:AppVirtual.svg** *Source:* <http://upload.wikimedia.org/wikipedia/commons/5/57/AppVirtual.svg> *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Egmason
- **File:Basilisk_snapshot9.png** *Source:* http://upload.wikimedia.org/wikipedia/en/b/bb/Basilisk_snapshot9.png *License:* ? *Contributors:* Originally uploaded by commons:User:Pabloignacio 02 *Original artist:* ?
- **File:Bus_icon.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/ca/Bus_icon.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:CloudComputingSampleArchitecture.svg** *Source:* <http://upload.wikimedia.org/wikipedia/commons/7/79/CloudComputingSampleArchitecture.svg> *License:* GFDL *Contributors:* Scalable Vector Graphic created by Sam Johnston using OminGroup's OmniGraffle *Original artist:* Sam Johnston, Australian Online Solutions Pty Ltd
- **File:Cloud_Computing.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/2/23/Cloud_Computing.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Davide Lamanna
- **File:Cloud_computing.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/b/b5/Cloud_computing.svg *License:* CC BY-SA 3.0 *Contributors:* Created by Sam Johnston using OmniGroup's OmniGraffle and Inkscape (includes Computer.svg by Sasa Stefanovic) *Original artist:* Sam Johnston
- **File:Cloud_computing_icon.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/12/Cloud_computing_icon.svg *License:* CC BY-SA 3.0 *Contributors:* Own work. Cloud icon is from <img alt="Weather-overcast.svg" src="//upload.wikimedia.org/wikipedia/commons/thumb/f/f6/Weather-overcast.svg/48px-Weather-overcast.

- svg.png' width='48' height='48' srcset='//upload.wikimedia.org/wikipedia/commons/thumb/f/f6/Weather-overcast.svg/72px-Weather-overcast.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/f/f6/Weather-overcast.svg/96px-Weather-overcast.svg.png 2x' data-file-width='48' data-file-height='48' />(Public domain), computer icon is from (GPL) *Original artist:* 百樂兔
- **File:Cloud_computing_layers.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/3c/Cloud_computing_layers.png *License:* Public domain *Contributors:* ? *Original artist:* ?
 - **File:Cloud_computing_types.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/8/87/Cloud_computing_types.svg *License:* CC BY-SA 3.0 *Contributors:* wikipedia *Original artist:* Sam Joton
 - **File:Commons-logo.svg** *Source:* <http://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg> *License:* ? *Contributors:* ? *Original artist:* ?
 - **File:Crystal_Clear_app_kedit.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/e8/Crystal_Clear_app_kedit.svg *License:* LGPL *Contributors:* Sabine MINICONI *Original artist:* Sabine MINICONI
 - **File:DOSBox_screenshot.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/e/e2/DOSBox_screenshot.png *License:* GPL *Contributors:* Transferred from en.wikipedia; transferred to Commons by User:IngerAlHaosului using CommonsHelper. *Original artist:* Original up loader was SF007 at en.wikipedia. Later version(s) were up loaded by Tornado500, McLoaf at en.wikipedia.
 - **File:Edit-clear.svg** *Source:* <http://upload.wikimedia.org/wikipedia/en/f/f2/Edit-clear.svg> *License:* Public domain *Contributors:* The Tango! Desktop Project. *Original artist:* The people from the Tango! project. And according to the meta-data in the file, specifically: “Andreas Nilsson, and Jakub Steiner (although minimally).”
 - **File:Emu_running_emu.jpg** *Source:* http://upload.wikimedia.org/wikipedia/en/8/80/Emu_running_emu.jpg *License:* ? *Contributors:* Shot by Chaheel Riens (talk · contribs) *Original artist:* ?
 - **File:Folder_Hexagonal_Icon.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/4/48/Folder_Hexagonal_Icon.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?
 - **File:Free_Software_Portal_Logo.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/31/Free_and_open-source_software_logo_%282009%29.svg *License:* Public domain *Contributors:* FOSS Logo.svg *Original artist:* Free Software Portal Logo.svg (FOSS Logo.svg); ViperSnake151
 - **File:Hardware_Virtualization_(copy).svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/0/08/Hardware_Virtualization_%28copy%29.svg *License:* Public domain *Contributors:* Own work *Original artist:* John Aplesed
 - **File:Hyperviseur.png** *Source:* <http://upload.wikimedia.org/wikipedia/commons/e/e1/Hyperviseur.png> *License:* CC0 *Contributors:* Own work *Original artist:* Scsami
 - **File:Internet_map_1024.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/d/d2/Internet_map_1024.jpg *License:* CC BY 2.5 *Contributors:* Originally from the English Wikipedia; description page is/was here. *Original artist:* The Opte Project
 - **File:Memory_virtualization_addressable_level.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/5/56/Memory_virtualization_addressable_level.png *License:* Public domain *Contributors:* Own work *Original artist:* Patrick Sullivan
 - **File:Memory_virtualization_application_level.png** *Source:* http://upload.wikimedia.org/wikipedia/commons/c/cb/Memory_virtualization_application_level.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Expatrick
 - **File:Mergefrom.svg** *Source:* <http://upload.wikimedia.org/wikipedia/commons/0/0f/Mergefrom.svg> *License:* Public domain *Contributors:* ? *Original artist:* ?
 - **File:Question_book-new.svg** *Source:* http://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:* Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:* Tkgd2007
 - **File:SDN-architecture-overview-transparent.png** *Source:* <http://upload.wikimedia.org/wikipedia/commons/e/e6/SDN-architecture-overview-transparent.png> *License:* CC BY-SA 3.0 *Contributors:* SDN Architecture Overview (PDF), Version 1.0, December 12, 2013. *Original artist:* Open Networking Foundation (ONF)
 - **File:Text_document_with_red_question_mark.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)
 - **File:USB_flash_drive.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/6/67/USB_flash_drive.jpg *License:* Public domain *Contributors:* Photograph taken by Dori *Original artist:* Photograph taken by Dori
 - **File:VirtualBox2.png** *Source:* <http://upload.wikimedia.org/wikipedia/commons/a/af/VirtualBox2.png> *License:* GFDL *Contributors:* Hidro (talk) *Original artist:* VirtualBox: Sun Microsystems, Inc., innotek GmbH; other works: various
 - **File:Wiki_letter_w_cropped.svg** *Source:* http://upload.wikimedia.org/wikipedia/commons/1/1c/Wiki_letter_w_cropped.svg *License:* CC-BY-SA-3.0 *Contributors:*
 - Wiki_letter_w.svg *Original artist:* Wiki_letter_w.svg: Jarkko Piironen
 - **File:Wiktionary-logo-en.svg** *Source:* <http://upload.wikimedia.org/wikipedia/commons/f/f8/Wiktionary-logo-en.svg> *License:* Public domain *Contributors:* Vector version of Image:Wiktionary-logo-en.png. *Original artist:* Vectorized by Fvasconcellos (talk · contribs), based on original logo tossed together by Brion Vibber
 - **File:Wzonka-Lad_on_E-UAE_on_Linux.png** *Source:* http://upload.wikimedia.org/wikipedia/en/4/45/Wzonka-Lad_on_E-UAE_on_Linux.png *License:* Fair use *Contributors:* my own desktop *Original artist:* ?

23.8.3 Content license

- Creative Commons Attribution-Share Alike 3.0