Linux® VServer™

An overview of Virtualization and Isolation

Herbert Pötzl

1 Introduction

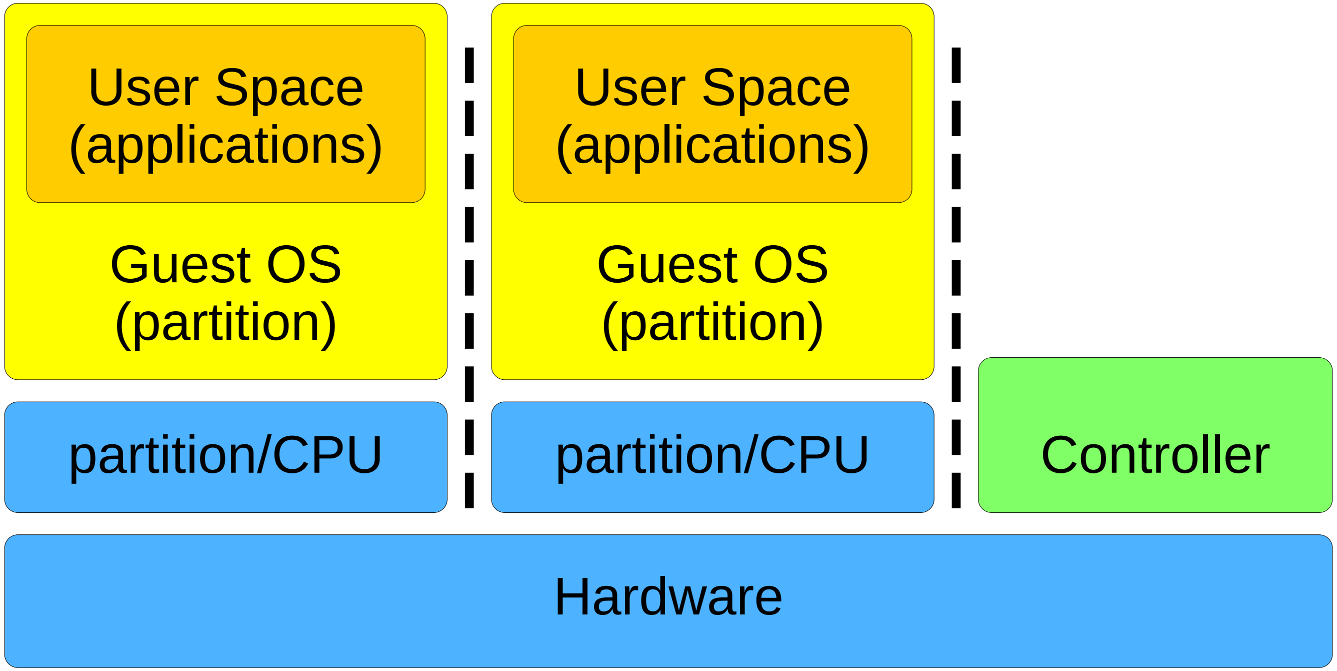
Computers have become sufficiently powerful to use „virtualization“ to create the illusion of many smaller virtual machines, each running a separate operating system instance.

- Partitioning
- Virtual Machines
- System Emulators

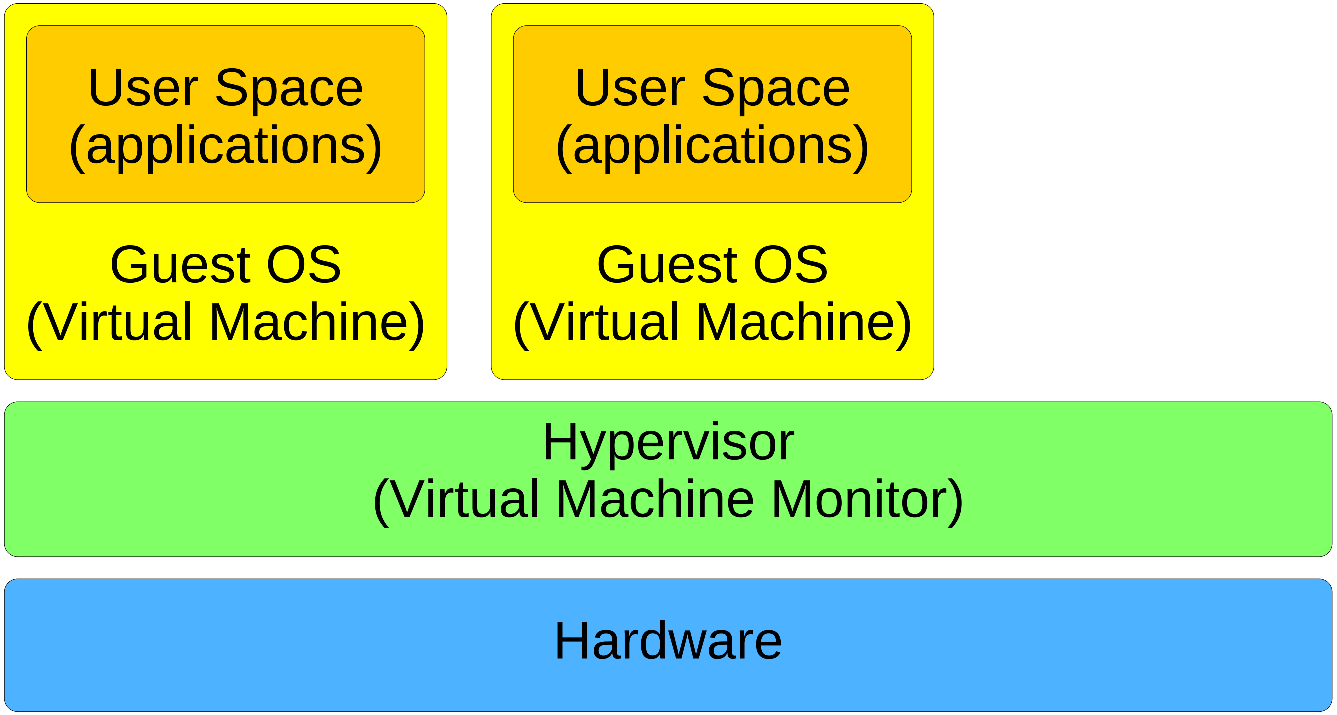
But not always a separate operating system instance is required or useful at all. Isolation better known as operating system-level virtualization and implemented as Security Contexts, Jails, or Zones can provide improved performance, increased flexibility and reduced resource usage.

- ➠ Security Contexts
- ➠ BSD Jails
- ➠ Solaris Zones

2 Partitioning (HW Approach)



3 Virtualization (VM Approach)



User Space
(applications)

Guest OS
(Virtual Machine)

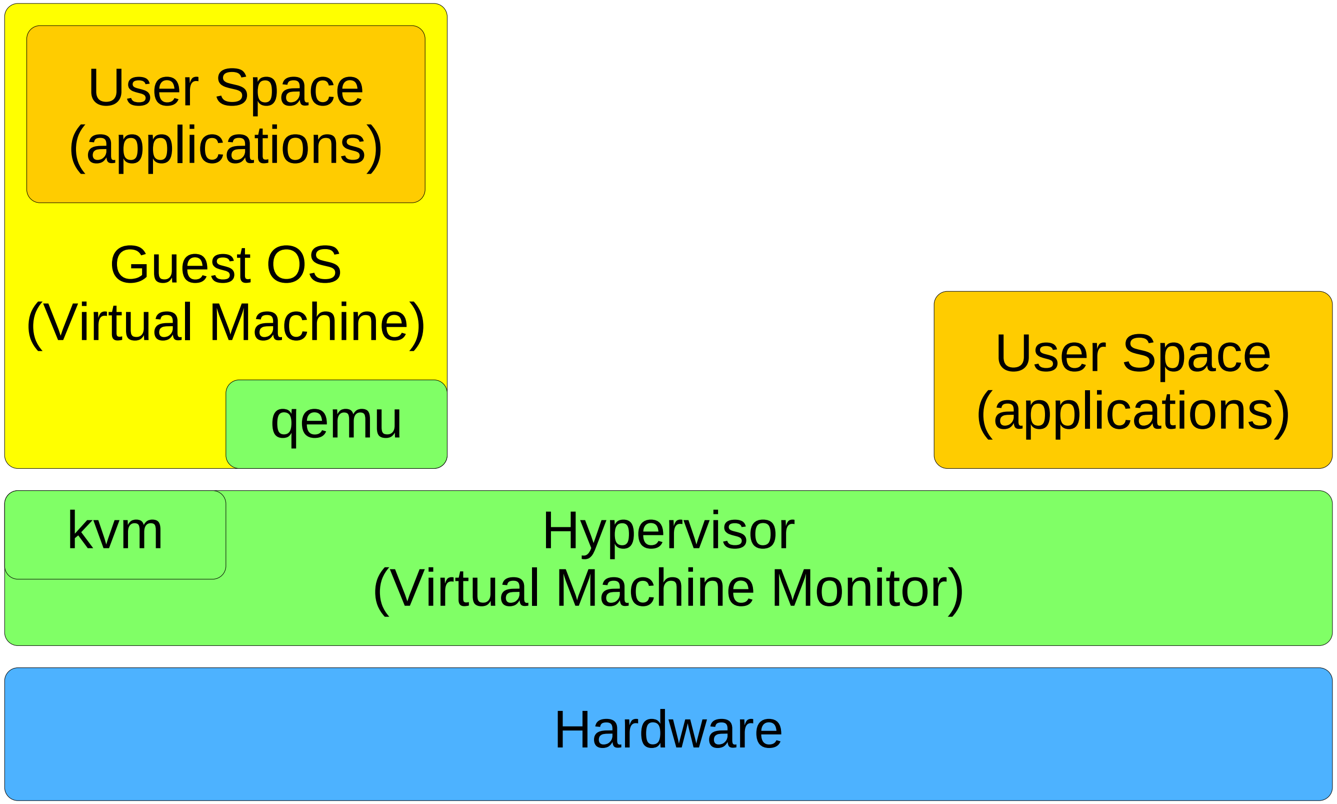
User Space
(applications)

Guest OS
(Virtual Machine)

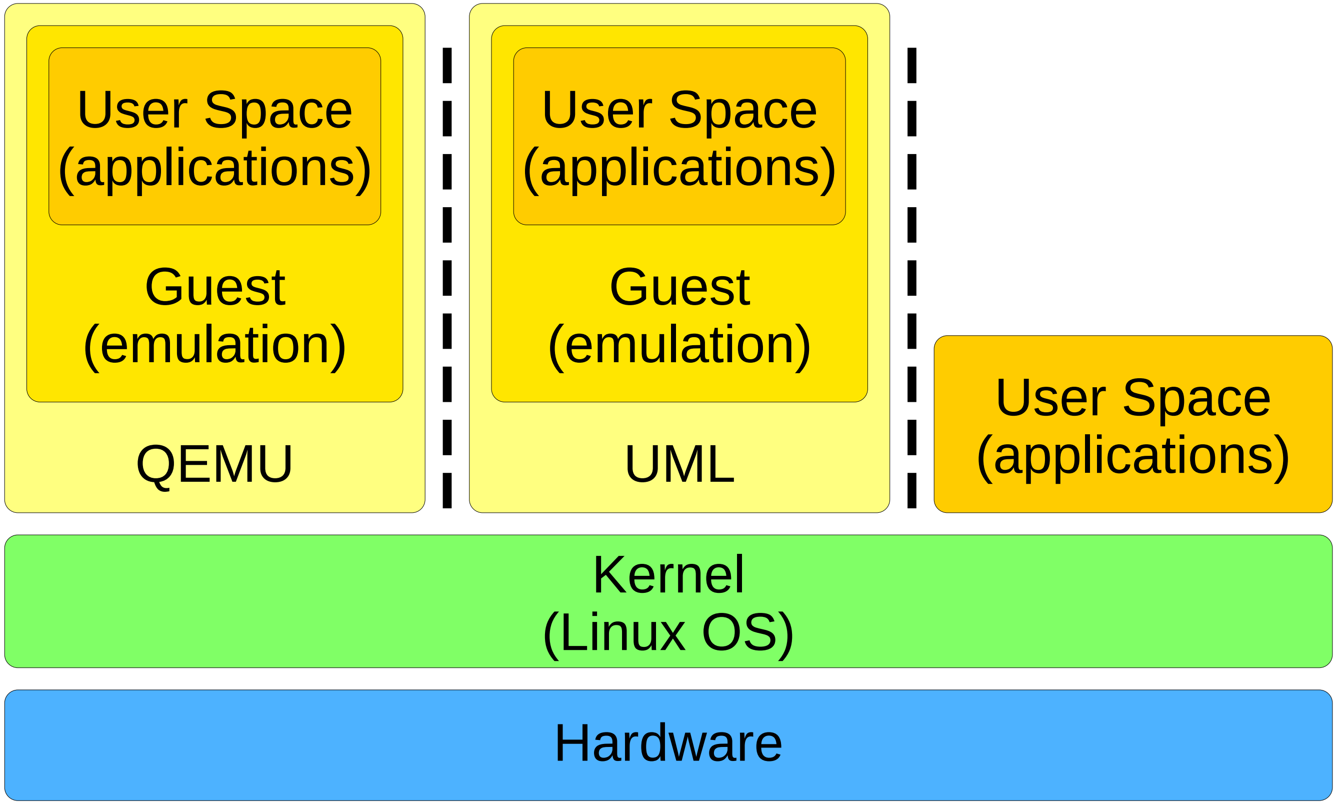
Hypervisor
(Virtual Machine Monitor)

Hardware

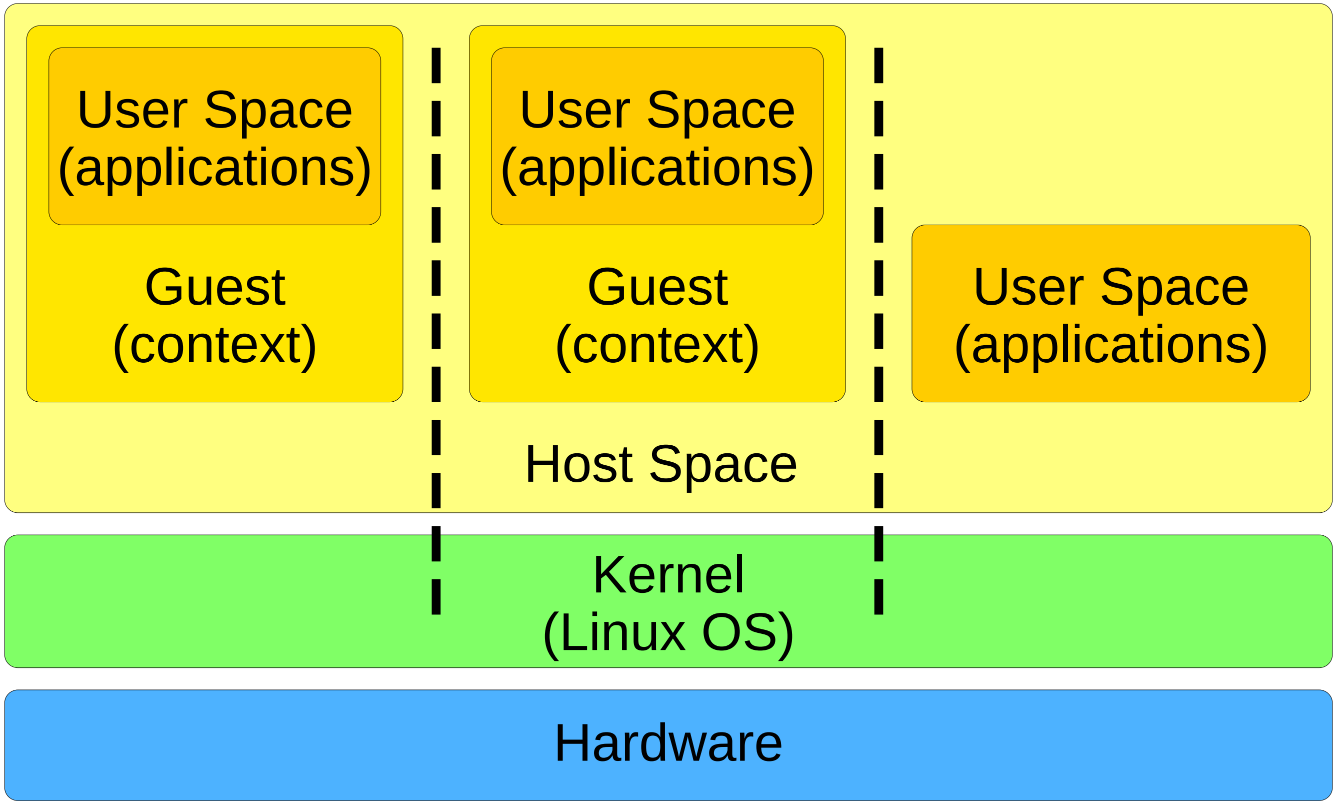
4 Virtualization (KVM Approach)



5 Emulation (QEMU/UML Approach)



6 Isolation (Jail Approach)



7 Virtualization Timeline

1941	Zuse Z3 (one of the first computers)
1959	IBM 7090 gets installed
1963	IBM 7040/7044 is available
1964	IBM M44/44X provides simulated 7044 machines
1965	IBM CP-40 goes full virtualization with VM/360
1966	IBM CP-67 and VM/370 have become standard
1975	MITS Altair 8800 is sold as mail-order kit (PC)
1998	VMware, Inc. is founded
2003	Xen^{TM} is first released
2004	QEMU is released into the public
2007	KVM is introduced in mainline Linux (2.6.20)

8 Isolation Timeline

2000	FreeBSD 4.x introduces Jails
2001	Linux-VServer (Security Contexts) released
2001	SWsoft (Now Parallels) releases Virtuozzo^{TM}
2005	Solaris Zones are released with Solaris 10
2006	SWsoft (Now Parallels) releases OpenVZ
2008	Linux (mainline) starts implementing isolation

Linux® VServer™

Light-Weight and Resource Efficient OS-Level Virtualization

Herbert Pötzl

9 The Concept

Virtual Servers do not necessarily require a separate operating system for each instance

Resources directly map to money – more servers require more CPU power, RAM, disk space, network bandwith and general I/O throughput.

Isolation allows to put several Servers on a Host, which will share the available resources efficiently.

9.1 Advantages

- ✘ Minimal Overhead
- ✘ Hardware Abstraction
- ✘ Shared Resources

9.2 Possible Drawbacks

- ✘ Kernel as Single Point of Failure?
- ✘ Kernel Security Issues?

10 Nomenclature

Host: the real or virtual machine running the Linux-VServer enabled Kernel.

Guest: the virtual private server (or short VPS) composed of a chrooted environment, isolated processes, and restricted IP ranges.

Context: the isolated and partially virtualized environment to which processes are *confined*.

11 Project History

Jul. 2001	first public release
Oct. 2001	Rik van Riel shows interest
Nov. 2001	new Immutable-Linkage-Invert flag
Jan. 2002	chroot exploit and barrier idea
Jul. 2002	Herbert Pötzl suggests context quota :)
Jul. 2003	Sam Vilain suggests 'going mainline'
Sep. 2003	Change of Project Maintainership
Mar. 2004	First Pre-Release for 2.6.x
May. 2004	First Devel Release (1.9.0) for 2.6
Aug. 2005	First Stable Release (2.0) for 2.6

12 Isolation vs. Virtualization

- ★ IP Layer **Network Isolation**

  . . . instead of **Virtual Network Stacks**

- ★ **Namespaces** and **Shared Partitions**

  . . . instead of **Virtual Filesystems**

- ★ **Accounting**, **Limits**, and **TB Scheduling**

  . . . instead of **vResources** and **vCPUs**

12.1 Lightweight Guests

Isolation allows to have very small Guests (down to a single process) without creating measurable overhead.

12.2 Shared Services

Isolation areas can overlap (to some extend) and services can be shared between Guests

12.3 Flexible Resources

Because there is a common pool of Resources, and no static allocation to the Guests, they can be easily . . .

- ✘ adjusted and shared
- ✘ monitored on the Host System
- ✘ limited or extended

13 Optional Virtualizations

- ✘ Init PID(1) *[pstree, init]*
- ✘ Network Interface Information
- ✘ Memory Information *[free, meminfo]*
- ✘ Available Disk Space *[df]*
- ✘ System Uptime *[guest start]*
- ✘ System Load *[guest processes]*
- ✘ System Time *[adjustable]*

14 Field of Application

- ✘ Virtual Server Hosting
- ✘ Administrative Separation
- ✘ Service Separation
- ✘ Enhancing Security
- ✘ Easy Maintenance
- ✘ Fail-over Scenarios
- ✘ Simplified Testing

15 Existing Infrastructure

- ✘ Linux Capability System
- ✘ Resource Limits (ulimit)
- ✘ File Attributes (xattr)
- ✘ The chroot(1) Command
- ✘ Private Namespaces

16 Required Modifications

- ✘ Context Separation
- ✘ Network Separation
- ✘ The Chroot Barrier
- ✘ Upper Bound for Caps
- ✘ Resource Isolation
- ✘ Filesystem XID Tagging

17 Additional Modifications

- ✘ Context Flags
- ✘ Context Capabilities
- ✘ Context Accounting
- ✘ Context Limits
- ✘ Virtualization
- ✘ Improved Security
- ✘ Kernel Helper

18 Features and Bonus Material

- ✘ Unification
- ✘ CoW Link Breaking
- ✘ The Linux-VServer Proc-FS
- ✘ TB Per CPU Scheduler
- ✘ Context Disk Limits
- ✘ Context Quota and VRoot Proxy
- ✘ Information Isolation

19 Intrusiveness

patch	lines	chars	hunks	new
vs1.00	2845	95567	178	997
vs1.20	4305	131922	216	1857
vs2.00	19673	557988	856	8987
vs2.01	20300	572752	898	9362
vs2.02	21330	602493	977	9464
vs2.1.0	25948	759709	1222	10394
vs2.2.0	27857	790256	1218	12989
openvz-2.6.22	122567	3384793	3654	73781
patch-2.6.23Δ	1072513	31824779	32650	359297

20 Non Intel x86 Hardware

- ✔ ia64, x86_64
- ✔ alpha, arm
- ✔ hppa, hppa64
- ✔ ppc, ppc64
- ✔ sparc, sparc64
- ✔ mips o/n32, mips64
- ✔ s390, s390x
- ✔ um, xen

Q & A

www: http://linux-vserver.org
irc: #vserver @ irc.oftc.net