LinuX-Container

Erkan Yanar

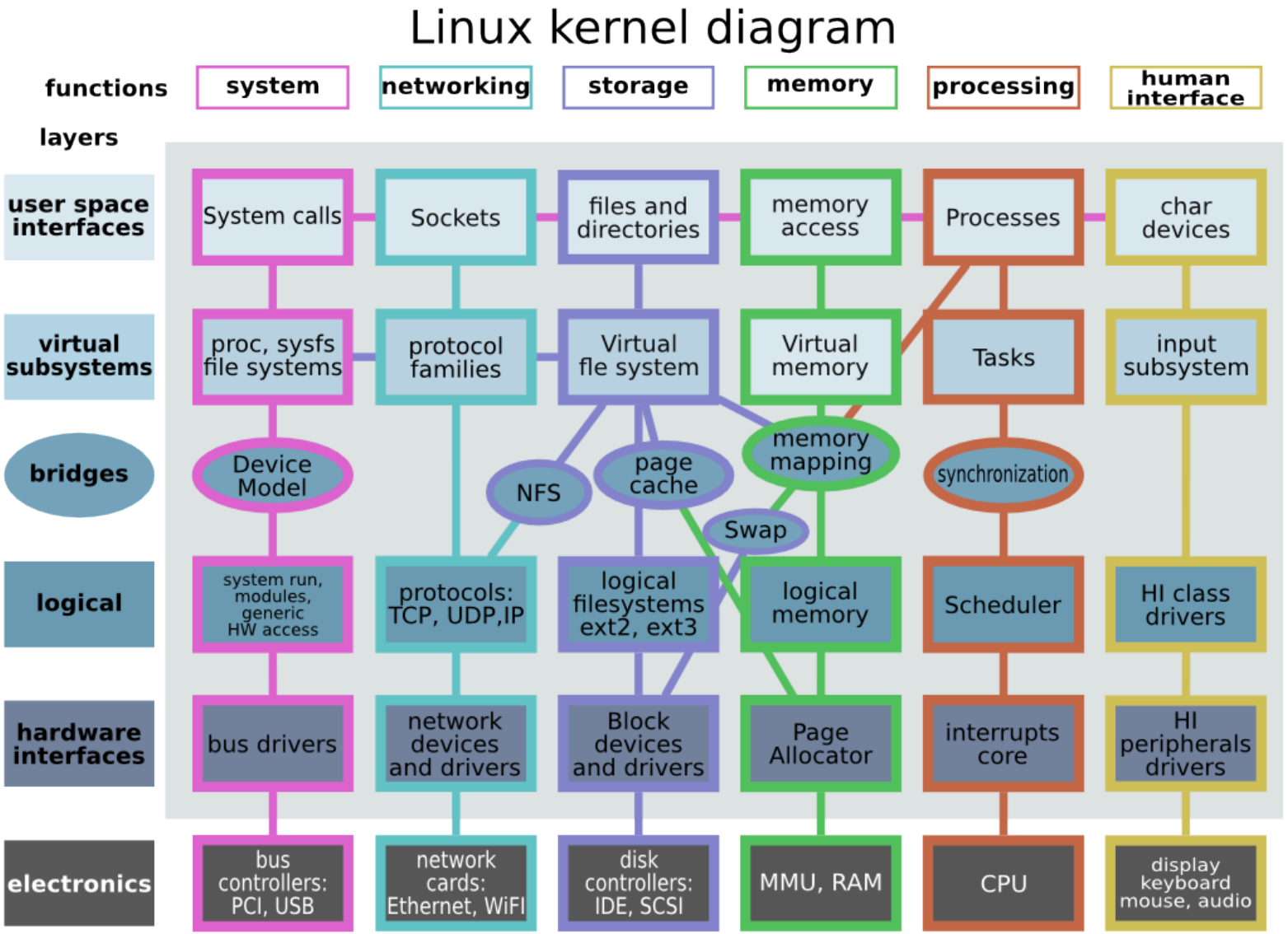
1. März 2012

Container aka OS-Virtualisierung und die Anderen unter Linux

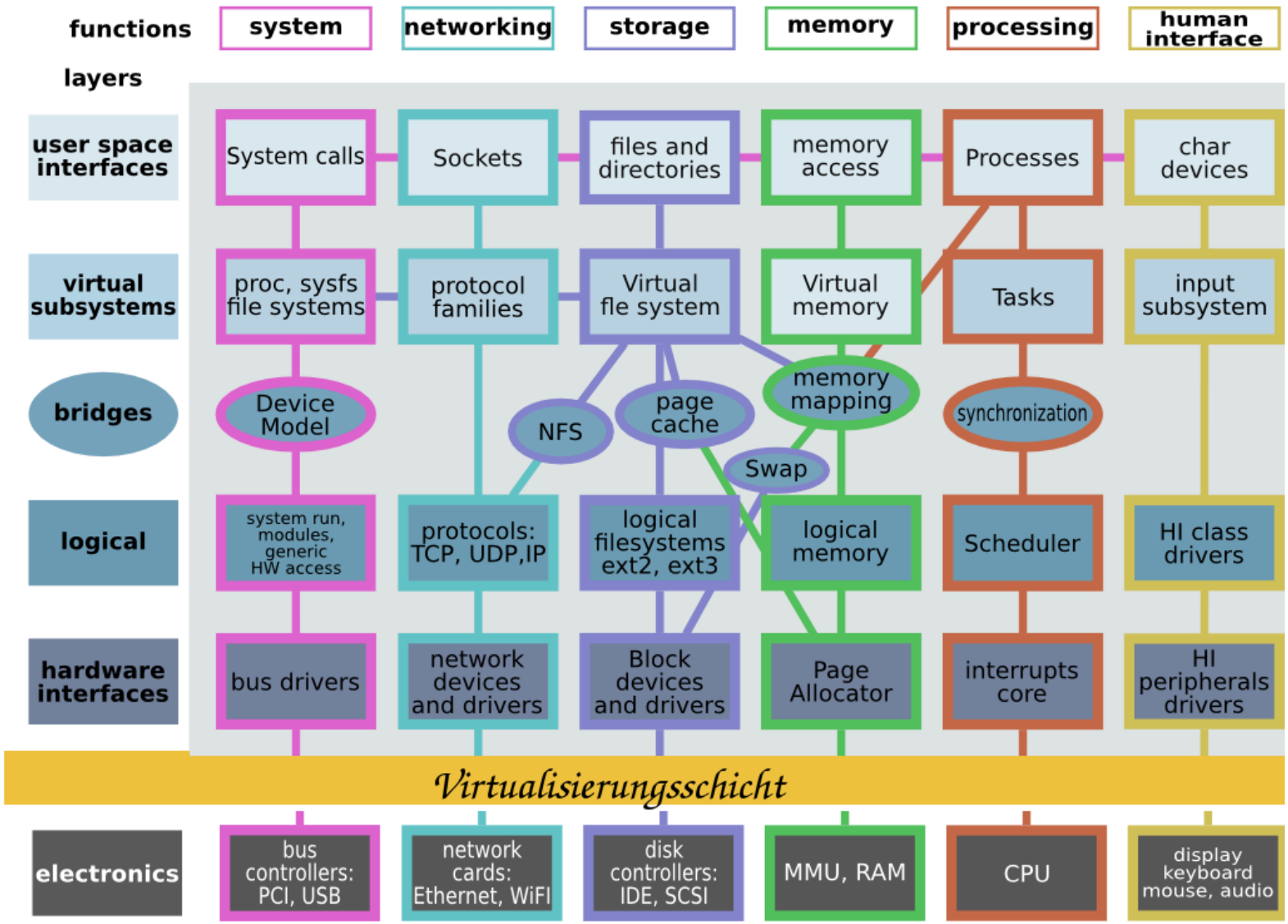
Möglichkeiten:

LXC

Linux

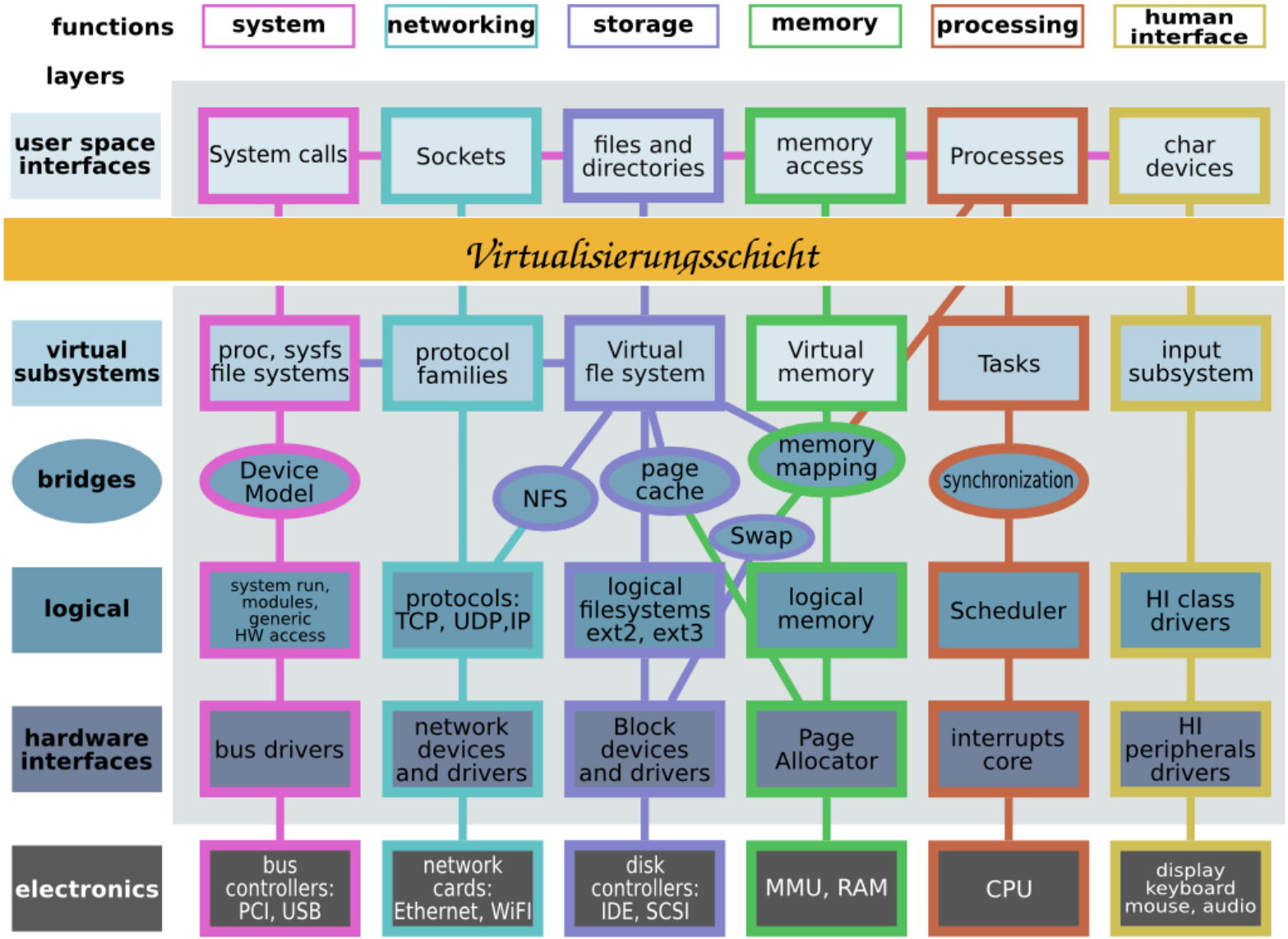

Linux kernel diagram
functions
system
networking
storage
memory
processing
human interface
layers
user space interfaces
virtual subsystems
bridges
logical
hardware interfaces
electronics
System calls
Sockets
files and directories
memory access
Processes
char devices
proc, sysfs file systems
protocol families
Virtual fle system
Virtual memory
Tasks
input subsystem
Device Model
NFS
page cache
memory mapping
synchronization
Swap
system run, modules, generic HW access
protocols: TCP, UDP,IP
logical filesystems ext2, ext3
logical memory
Scheduler
HI class drivers
bus drivers
network devices and drivers
Block devices and drivers
Page Allocator
interrupts core
HI peripherals drivers
bus controllers: PCI, USB
network cards: Ethernet, WiFI
disk controllers: IDE, SCSI
MMU, RAM
CPU
display keyboard mouse, audio
© 2007-2009 Constantine Shulyupin http://www.MakeLinux.net/kernel/diagram

KVM etc.



© 2007-2009 Constantine Shulyupin http://www.MakeLinux.net/kernel/diagram

Container



© 2007-2009 Constantine Shulyupin http://www.MakeLinux.net/kernel/diagram

Two Worlds

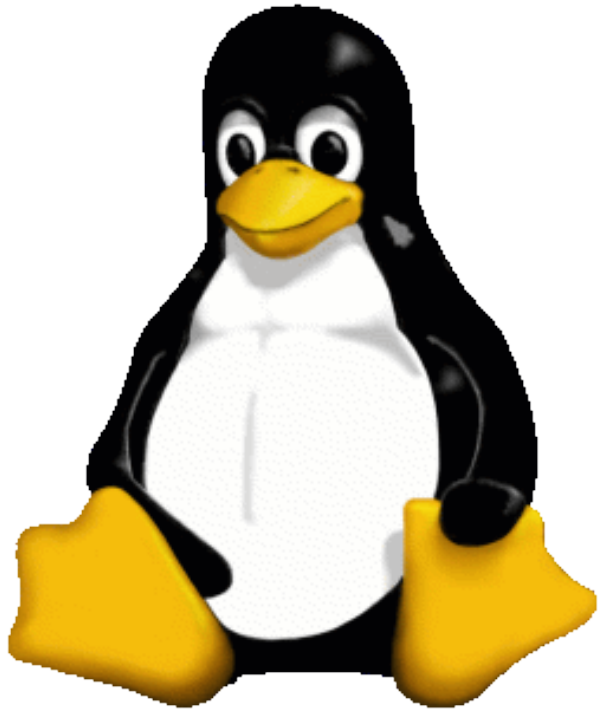
Hardware Virtualisierung kann mehr!

	Hardware Virt.	Betriebssystem Virt.
	KVM, Xen	LXC, OpenVZ
Separates OS	X	-
Seperater Kernel	X	-
Andere Hardwarearchitekturen	qemu	-
Geringerer Overhead	-	X

Unterschied von Para- und Full-Virtualisierung wird vernachlässigt

Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

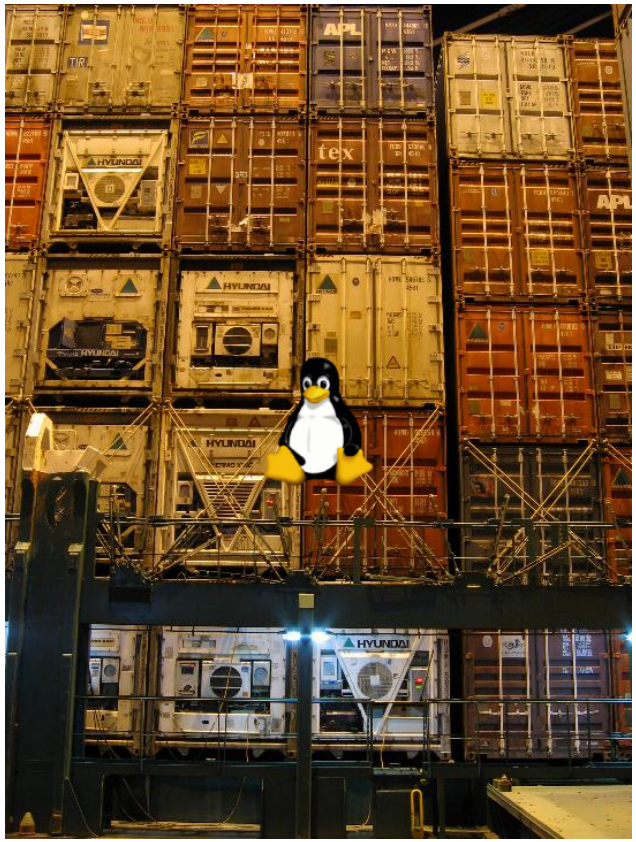


Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

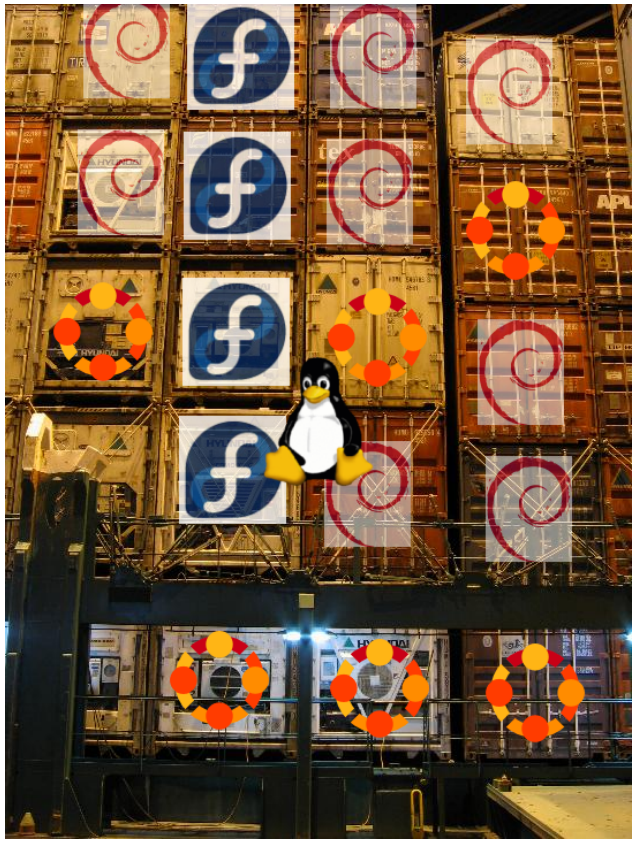
Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler



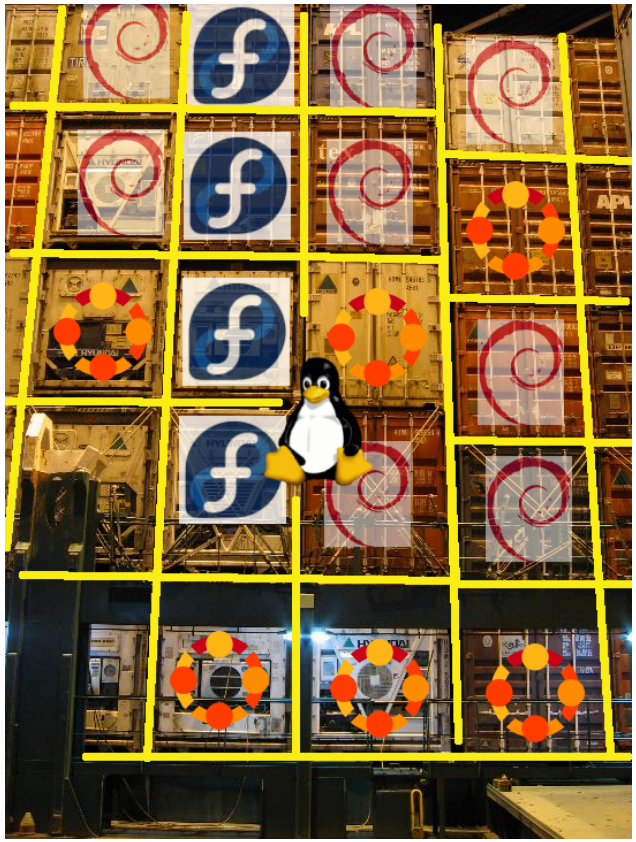
Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler



Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler



Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

Container sind:

- Virtualisierung im OS
- Container / Verzeichnisse
- Hostkernel übernimmt die Verwaltung
- → Nur ein OS-Typ
- Dünne Virtualisierungsschicht
- Prozessvirtualisierung
- Dynamische Zuweisung von Ressourcen
- CPU und I/O Scheduler

Überblick

Look@Container ala LXC

Überblick

Look@Container ala LXC

1. cgroups: Ressourcenmanagement
2. LXC: (Applikations)Container on top
3. OpenVZ vs. LXC kurzer Überblick

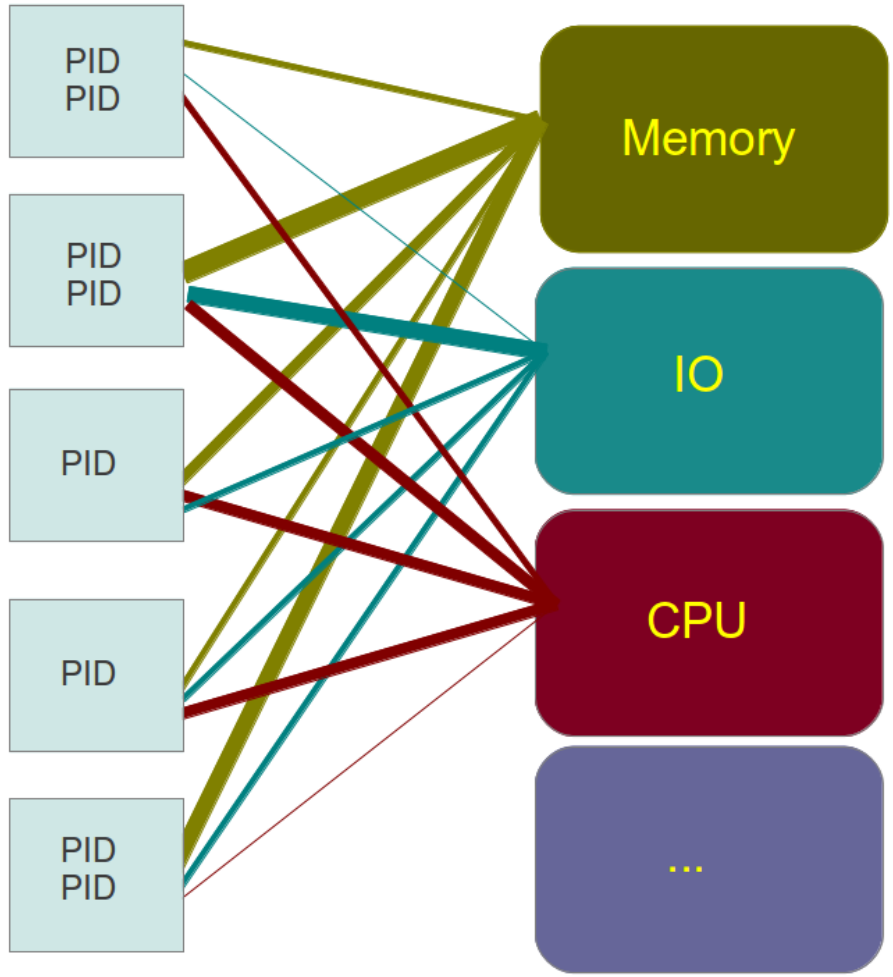
Ressourcenmanagement mit cgroups

Control Groups

- Gruppieren von Prozessen
- Gemeinsame Ressourcen
- Childs bleiben in der Gruppe

Control Groups

- VFS
- $\geq$ Kernel 2.6.24
- unabhängig von LXC
- mount:
  `cgroup /cgroups cgroup defaults 0 0`



Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1                                                    CPU
cpuacct 3 4 1
memory 4 4 1
devices 5 4 1
freezer 6 4 1
net_cls 7 1 1
blkio 8 4 1
```

Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1
cpuacct 3 4 1
memory 4 4 1                                    Speicher
devices 5 4 1
freezer 6 4 1
net_cls 7 1 1
blkio 8 4 1
```

Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1
cpuacct 3 4 1
memory 4 4 1
devices 5 4 1                                    mknod
freezer 6 4 1
net_cls 7 1 1
blkio 8 4 1
```

Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1
cpuacct 3 4 1
memory 4 4 1
devices 5 4 1
freezer 6 4 1                                   FROZEN/THAWED
net_cls 7 1 1
blkio 8 4 1
```

Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1
cpuacct 3 4 1
memory 4 4 1
devices 5 4 1
freezer 6 4 1
net_cls 7 1 1                                    Markieren
blkio 8 4 1
```

Subsysteme/Controlgroups

```
cat /proc/cgroups
 #subsys_name hierarchy num_cgroups enabled
cpuset 1 4 1
cpu 2 4 1
cpuacct 3 4 1
memory 4 4 1
devices 5 4 1
freezer 6 4 1
net_cls 7 1 1
blkio 8 4 1
```

CFQ

```
# ls /cgroups #2.6.38 (Auszug) --- To be deleted --
blkio.throttle.read_bps_device  cpuset.memory_pressure
blkio.throttle.read_iops_device cpuset.memory_pressure_enabled
blkio.throttle.write_bps_device cpuset.mems
blkio.throttle.write_iops_device cpuset.sched_load_balance
blkio.weight                    cpuset.sched_relax_domain_level
cgroup.clone_children           cpu.shares
cgroup.procs                    devices.allow
cpuacct.stat                    memory.limit_in_bytes
cpuacct.usage                   memory.memsw.limit_in_bytes
cpuacct.usage_percpu            memory.oom_control
cpuset.cpu_exclusive            memory.stat
cpuset.cpus                     memory.swappiness
cpuset.mem_exclusive            memory.usage_in_bytes
cpuset.mem_hardwall             net_cls.classid
cpuset.memory_migrate           tasks
```

LXC

LXC
LinuXContainer

LXC

LinuXContainer

Ein chroot macht auf virtuell

LXC
LinuXContainer

Grundprinzipien und Stolpersteine

LinuXContainer

LXC: better cgroups?

- Spätestens seit 2.6.26 im Kernel (Network-Namespace)
- Erzeugt mit Hilfe von Namespaces Container.
- cgroups dienen zur Resourcenverwaltung (auch bei KVM).
- LXC übernimmt die Verwaltung der Prozessgruppen
- Modulares Design!

Namespaces Die Seele der Virtualisierung

utsname	hostname	[Modular]
Pid	private PIDs	[Automatisch]
User	private UIDs	[Automatisch]
Network	privates Interface	[Modular]
Ipc	privates IPC	[Automatisch]

LXC virtualisiert chroot() Umgebungen

Konfiguration

/var/lib/lxc/$CONTAINER Konfigurationsverzeichnis des Containers

/var/lib/lxc/$CONTAINER/config Konfigurationsdatei des Containers

Wo ist das chroot Verzeichnis?

(lxc.)rootfs Filesystem des Containers

LXC startet dieses „System“

LXC-Tools

Auszug

lxc-create Erstellt einen Container

lxc-destroy Löscht rootfs und das Configverzeichnis

LXC-Tools

Auszug

lxc-create Erstellt einen Container

lxc-destroy Löscht rootfs und das Configverzeichnis

Unnötiges Commando?

lxc-create -n name [-f config_file] [-t template]

- Schreibe mit config_file nach /var/lib/lxc/$name/config
- Nutze Template (Skript) zum Erstellen eines Containers
- Mehr zu Templates? Probleme?

Container Filesystem

Erstelle einen Container:

- debootstrap, febootstrap ..
- udevd entfernen
  mknod ...
- hwclock entfernen ...

lxc-debian

```
/usr/sbin/update-rc.d -f checkroot.sh remove
/usr/sbin/update-rc.d -f umountfs remove
/usr/sbin/update-rc.d -f hwclock.sh remove
/usr/sbin/update-rc.d -f hwclockfirst.sh remove
/usr/sbin/update-rc.d -f module-init-tools remove
```

`/usr/lib/lxc/templates`

Container Filesystem

Erstelle einen Container:

- debootstrap, febootstrap ..
- udevd entfernen
  mknod ...
- hwclock entfernen ...

lxc-debian

```
/usr/sbin/update-rc.d -f checkroot.sh remove
/usr/sbin/update-rc.d -f umountfs remove
/usr/sbin/update-rc.d -f hwclock.sh remove
/usr/sbin/update-rc.d -f hwclockfirst.sh remove
/usr/sbin/update-rc.d -f module-init-tools remove
```

`/usr/lib/lxc/templates`

Container zeigen

Man erstelle eine Konfigurationsdatei

Konfiguration

lxc.rootfs	chroot
lxc.mount.entry	Ein Mountpunkt im fstab-Format
lxc.mount	Pfad zu einem File mit Mountp. im fstab Format
lxc.tty	Virtuelle Consolen: lxc-console
lxc.pts	Pseudo ttys
lxc.cap.drop	man capabilities

```
lxc.tty = 4
lxc.rootfs = /lxc/debian/rootfs
lxc.mount = /lxc/debian/fstab
```

Network

lxc.network.type

Kein Eintrag	Interfaceeinstellungen des Hosts
empty	loopback
veth	Virtual Ethernet (bridge)
macvlan	MAC-Address based Vlan
phys	physisches Interface

```
lxc.network.type = veth
lxc.network.flags= up
lxc.network.link = br0
lxc.network.ipv4 =
192.168.1.69/24
lxc.network.name = eth0
lxc.network.veth.pair =
this-veth
```

/var/lib/lxc/$CONTAINER/config

```
lxc.utsname = zeig
lxc.tty = 4
lxc.pts = 1024
lxc.mount = /lxc/debian/fstab
lxc.rootfs = /lxc/debian/rootfs
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.hwaddr = 08:00:12:34:56:78
lxc.network.ipv4 = 192.168.1.69/24
lxc.network.name = eth0
lxc.cgroup.devices.deny = a
# /dev/null and zero
lxc.cgroup.devices.allow = c 1:3 rwm
lxc.cgroup.devices.allow = c 1:5 rwm
# consoles
lxc.cgroup.devices.allow = c 5:1 rwm
lxc.cgroup.devices.allow = c 5:0 rwm
lxc.cgroup.devices.allow = c 4:0 rwm
lxc.cgroup.devices.allow = c 4:1 rwm
```

LXC-*Tools*

Auszug:

lxc-ls	Zeigt alle konfigurierten und laufenden Container
lxc-start/stop	Starten/Stoppen eines Containers
lxc-ps	Wrapper um ps mit Containername
lxc-console	Konsolenverbindung zum Container
lxc-execute	Startet einen Prozess im ContainerEnvironment

```
# lxc-ls
busy01  debian  first  hiho  lucid
busy01
#
```

LXC-*Tools*

Auszug:

lxc-ls	Zeigt alle konfigurierten und laufenden Container
lxc-start/stop	Starten/Stoppen eines Containers
lxc-ps	Wrapper um ps mit Containername
lxc-console	Konsolenverbindung zum Container
lxc-execute	Startet einen Prozess im ContainerEnvironment

```
# lxc-start -n busy01
init started: BusyBox v1.13.3 (Ubuntu 1:1.13.3-1ubuntu11)

Please press Enter to activate this console.
```

LXC-*Tools*

Auszug:

lxc-ls	Zeigt alle konfigurierten und laufenden Container
lxc-start/stop	Starten/Stoppen eines Containers
lxc-ps	Wrapper um ps mit Containername
lxc-console	Konsolenverbindung zum Container
lxc-execute	Startet einen Prozess im ContainerEnvironment

```
# lxc-ps --lxc
CONTAINER    PID TTY          TIME CMD
busy01      3971 ?        00:00:00 init
busy01      3975 ?        00:00:00 busybox
busy01      3977 pts/4    00:00:00 getty
busy01      3978 ?        00:00:00 sh
```

LXC-*Tools*

Auszug:

lxc-ls	Zeigt alle konfigurierten und laufenden Container
lxc-start/stop	Starten/Stoppen eines Containers
lxc-ps	Wrapper um ps mit Containername
lxc-console	Konsolenverbindung zum Container
lxc-execute	Startet einen Prozess im ContainerEnvironment

```
# lxc-console -n busy01

Type <Ctrl+a q> to exit the console

busy01 login:
```

LXC-*Tools*

Auszug:

lxc-ls	Zeigt alle konfigurierten und laufenden Container
lxc-start/stop	Starten/Stoppen eines Containers
lxc-ps	Wrapper um ps mit Containername
lxc-console	Konsolenverbindung zum Container
lxc-execute	Startet einen Prozess im ContainerEnvironment Applikationscontainer

```
# lxc-execute -n shell /bin/bash
root@shell:/#
```

Show me that stuff!

- Container start
- Host Zugriff
- Applikationskontainer mit Ressourcemanagement

Security

Capabilities

remind the fstab

lxc.cap.drop

root im Container zu mächtig

- module sys_module
- mount sys_admin

echo b > /proc/sysrq-trigger

- SELinux
- Smack

lxc.mount.entry=proc $lxc.rootfs/proc proc nodev,noexec,nosuid,ro 0 0

Applikationscontainer

lxc-execute

- Schlüssel zum Applikationscontainer
- braucht *kein* lxc.rootfs
- Config kann mehrmals verwendet werden ($--$ *name* , -f)
- Modularität Ausnutzen
- libcgroup-Ersatz

OpenVZ vs. LXC

Topic	LXC	OpenVZ
Kernelintegration	X	-
Livemigration	-	X
Host Konfigtools (vzctl)	-	X
Sicheres Netz (venet)	-	X
Sichere Container	-	X
Applikationscontainer	X	-
diskspace	-	X
Cgroups	X	-
Quota	-	X
Distro-Support	X	-
Produktionsreif	-	X
libvirt-Integration	X	(X)
Modular	X	-

Ende Gelände

erkan yanar
erkan.yanar@linsenraum.de
linsenraum.de/erkules
www.xing.com/profile/Erkan_Yanar