LXC, Cgroups and Advanced Linux Container Technology Lecture

SUS15

Novell Training Services

www.novell.com

ATT LIVE 2012 LAS VEGAS

Novell®

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Contents

Introduction to Cgroups

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 1

ATT Live 2012
Session SUS15

Objectives

- Kernel Control Groups
- Using Cgroups in Linux
- Building and Using Linux Containers with LXC
- Create Cgroups when the System Starts

5

Kernel Control Groups

What Are Control Groups?

Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behavior.

- cgroup is another name for Control Groups
- Partition tasks (processes) into a one or many groups of tree hierarchies
- Associate a set of tasks in a group to a set subsystem parameters
- Subsystems provide the parameters that can be assigned
- Tasks are affected by the assigning parameters

7

Definitions (from the Kernel-Docu)

- cgroup
- Subsystem (also called Resource Controller or simply Controller)
- Hierarchy

8

A *cgroup* associates a set of tasks with a set of parameters for one or more subsystems.
•A *subsystem* is a module that makes use of the task grouping facilities provided by cgroups to treat groups of tasks in particular ways. A subsystem is typically a "resource controller" that schedules a resource or applies per-cgroup limits, but it may be anything that wants to act on a group of processes, e.g. a virtualization subsystem.
•A *hierarchy* is a set of cgroups arranged in a tree, such that every task in the system is in exactly one of the cgroups in the hierarchy, and a set of subsystems; each subsystem has system-specific state attached to each cgroup in the hierarchy. Each hierarchy has an instance of the cgroup virtual filesystem associated with it.:

Example of the Capabilities of a cgroup

Consider a large university server with various users - students, professors, system tasks etc. The resource planning for this server could be along the following lines:

CPUs

```
      Top cpuset (20%)
          /     \
CPUSet1        CPUSet2
   |              |
 (Profs)      (Students)
  60%            20%
```

Memory

Professors = 50%

Students = 30%

System = 20%

Disk I/O

Professors = 50%

Students = 30%

System = 20%

Network I/O

```
WWW browsing = 20%
        /     \
Prof (15%)   Students (5%)
```

Network File System (60%)

Others (20%)

Source: /usr/src/linux/Documentation/cgroups/cgroups.txt

9

Control Group Subsystems

Two types of subsystems

- Isolation and special controls
  - cpuset, namespace, freezer, device, checkpoint/restart
- Resource control
  - cpu(scheduler), memory, disk i/o, network

Source: http://jp.linuxfoundation.org/jp_uploads/seminar20081119/CgroupMemcgMaster.pdf

10

Using Cgroups in Linux

Initializing Cgroups

- Run `/etc/init.d/boot.cgroup start`
- To activate cgroups automatically at system start, enter `chkconfig boot.cgroup on`
- This will populate the `/sys/fs/cgroup` directory
- Enter `mount` to see how it's accomplished
- Each available subsystem is represented by a subdirectory in `/sys/fs/cgroup`, such as `blkio cpu cpuacct cpuset devices freezer memory net_cls perf_event`

12

Notes:

Administering Cgroups

- cgroups are administered by creating subdirectories in the subsystem directories (such as `/sys/fs/cgroup/cpuset`) and changing files in those subdirectories
- There are tools that facilitate this, but the “raw” way using `mkdir` and `echo` helps understand the underlying mechanics

13

Notes:

Cgroups Example 1

- You want processes to share the time of the cpu in the ratio 6:4
- To do that, you create directories in the cpu subsystem and change files in that subdirectory (the subdirectory is automatically populated with several files after you create it)
- The commands are contained in the notes section

14

Note: The use of taskset simplifies the example as it binds the xterm calls to the first core. You could use the cpuset subsystem to achieve the same effect, but that would require additional commands. If you don't bind them to the same core, you don't see the effect in top, as they will probably run on different cores, using all the available cpu time.

The following commands can be used:
Create two groups:
cd /sys/fs/cgroup/cpu
mkdir higherload lowerload

Set the values:
echo 6 > higherload/cpu.shares
echo 4 > lowerload/cpu.shares

Start processes and assign them to one of the groups
taskset -c 0 xterm -bg orange &
taskset -c 0 xterm -bg green &

In the orange xterm, enter
echo $$ > /sys/fs/cgoup/cpu/higherload/tasks
In the green xterm, enter
echo $$ > /sys/fs/cgoup/cpu/lowerload/tasks

In the the terminal window, enter top
In each of the xterms, enter
md5sum /dev/urandom

Cgroups Example 2

- You want processes to share the time of the cpu in the ratio 6:4
- There are various `cg*` commands that simplify your task:
  - `cgcreate`: Creates a new cgroup
  - `cgset`: Set parameters within a cgroup
  - `cgexec`: Executes a command and puts it in a cgroup
  - `cgclear`: Removes all cgroups
- See the notes section for commands

15

The following commands can be used, using cpuset as well:
Create two groups:
cgcreate -g cpu,cpuset:higherload
cgcreate -g cpu,cpuset:lowerload

Set the values:
cgset -r cpu.shares=6 -r cpuset.cpus=0 higherload
cgset -r cpu.shares=4 -r cpuset.cpus=0 lowerload
The following is required for cpusets.cpus to work properly (see man cpuset)
cgset -r cpuset.mems=0 higherload
cgset -r cpuset.mems=0 lowerload

Start processes and assign them to one of the groups
cgexec -g cpu,cpuset:higherload xterm -bg orange &
cgexec -g cpu,cpuset:lowerload xterm -bg green &

In the the terminal window, enter top
In each of the xterms, enter
md5sum /dev/urandom

LAB 1-1: Use Linux Control Groups

Summary: In this exercise, you install, enable, and use Linux control groups.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

16

Lab Notes:

Create cgroups when the System Starts

Administering cgroups

- Commandline with mkdir and echo
- Commandline with cg* commands
- Startscript in /etc/init.d/ and /etc/cgconfig.conf to make cgroups persist over reboots

18

Notes:

cgconfig.conf

- groups section defines a cgroup
  - Permissions can be set for the task file (who may add tasks) and the rest of the files
  - Values are set for the individual ressources within the subsystems (such as cpu, cpuset, blkio, etc.)
- mount sections define were the cgroup subsystems are mounted (in SLES11SP2 this is usally done by the /etc/init.d/boot.cgroup script)

19

```
group daemons/www {
      perm {
            task {
                  uid = root;
                  gid = webmaster;
            }
            admin {
                  uid = root;
                 gid = root;
            }
      }
      cpu {
            cpu.shares = 1000;
      }
}
```

```
group daemons/ftp {
	perm {
		task {
			uid = root;
			gid = ftpmaster;
		}
		admin {
			uid = root;
			gid = root;
		}
	}
	cpu {
		cpu.shares = 500;
	}
}

#mount {
#	cpu = /mnt/cgroups/cpu;
#	cpuacct = /mnt/cgroups/cpuacct;
#}
```

Adding Processes to the cgroups

- /etc/init.d/cgconfig start
- chkconfig cgconfig on
- Modify service start scripts to put daemons into cgroups, for instance by adding cgexec as part of the start up of the daemon

21

Notes:

LAB 1-2: Configure cgconfig.conf

Summary: In this exercise, you configure cgroups so they get created when the system boots.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

22

Lab Notes:

Introduction to LXC

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 2

ATT Live 2012
Session SUS15

Objectives

- Introduction to Linux Containers and LXC
- Building and Using Linux Containers with LXC

Introduction to Linux Containers and LXC

What are Linux Containers

- Instances of Kernel-based isolation that go beyond simple chroot jail
- Not virtual machines exactly
- Rather virtual ENVIRONMENTS similar to chroot
- Provides more isolation than chroot
- Leverages Linux Control groups for container isolation and resource and process limits

26

Notes:

What is LXC?

- Project to create and manage Linux Containers
- Templates for creating virtualized containers
- Provides operating System level virtualization
  - (Without a "hypervisor" virtualization layer)
- Allows for multiple isolated server installs on a single host
- Only one kernel running on the host, not a separate kernel in each virtual machine as with Xen/KVM/VMware virtualization

27

Notes:

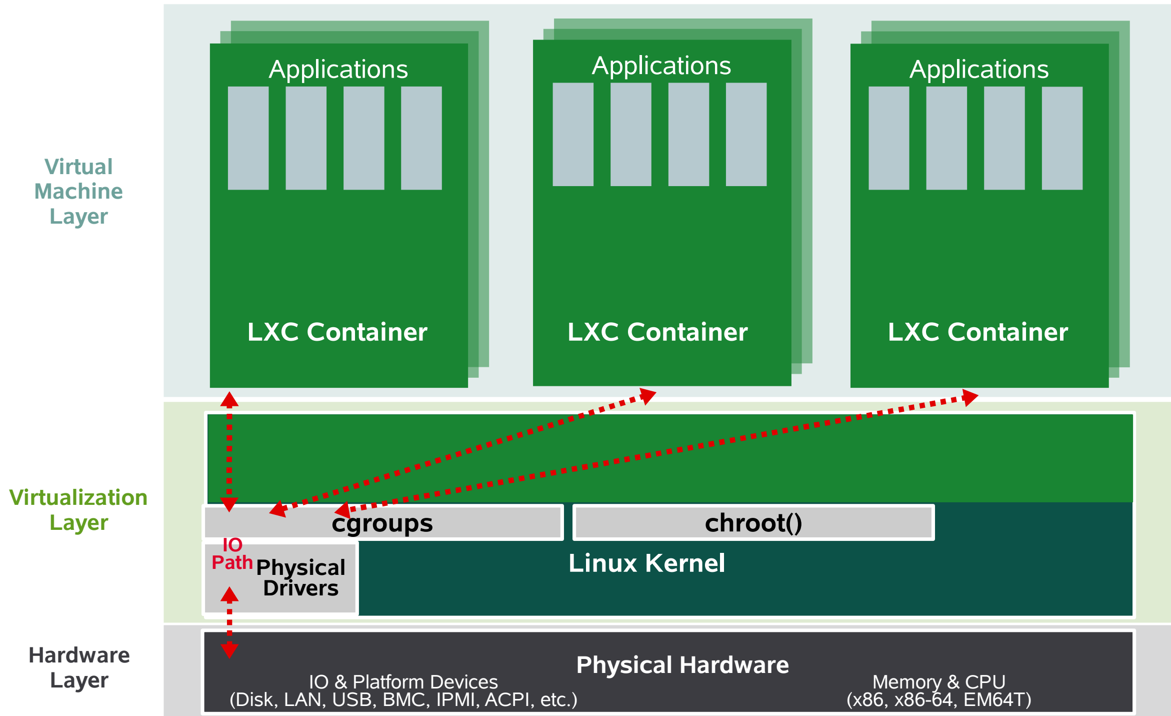
What are Linux Containers (LXC) (2)

- Does not provide virtual machines but rather virtual environments similar to chroot
- Provides more isolation than chroot
- Leverages Linux Control groups for container isolation and resource and process limits

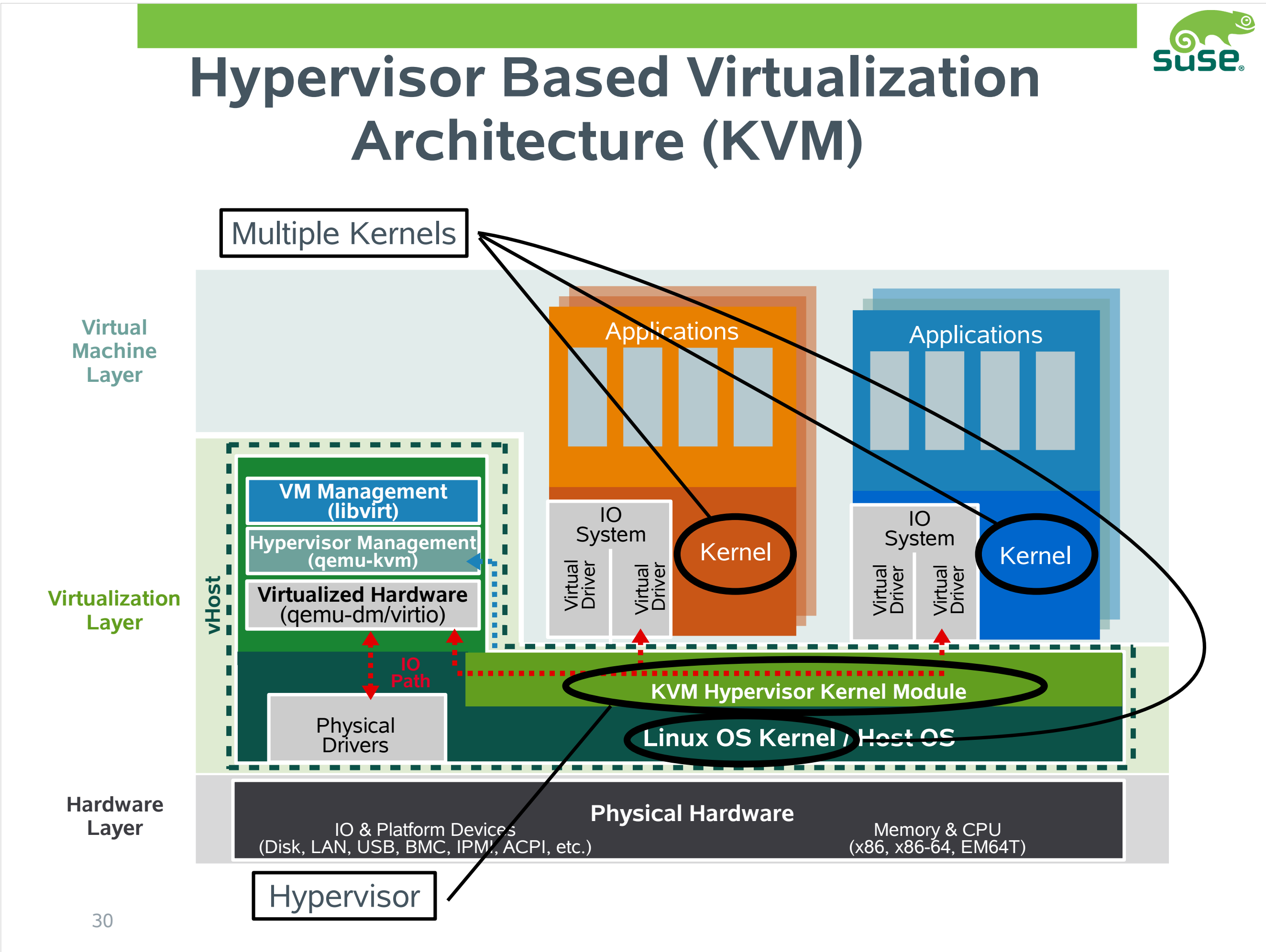
28

Notes:

Container Based Virtualization Architecture



29



With KVM (or the Kernel Virtual Machine), a kernel module is loaded into the Linux kernel that turns it into a hypervisor. KVM would essentially be a Type I Hypervisor because it is running directly on top of the hardware.

With KVM, the Linux kernel becomes a “fat” hypervisor because it not only mediates access to the underlying hardware but also loads physical drivers and shares access to the underlying hardware devises with virtual drivers.

Device emulation and VM management are handled by a modified version of QEmu running in user space.

Because KVM is a hypervisor based virtualization architecture multiple kernels can run on the hardware at the same time. These kernels can be virtualy anu operating system, not just linux.

Building and Using Linux Containers with LXC

Linux Containers – Virtualization

OS Level Virtualization – i.e. virtualization without a hypervisor (also known as “Lightweight virtualization”)

Similar technologies include: Solaris Zones, BSD Jails,Virtuozzo or OpenVZ

Advantages of OS Level Virtualization

- Minor I/O overhead
- Storage advantages
- Dynamic changes to parameters without reboot
- Combining virtualization technologies

Disadvantages

- Higher impact of a crash, especially in the kernel area
- Unable run another OS that cannot use the host's kernel

32

Linux Containers – Security

Missing user namespaces

- Allows evading from containers
- This is actively being worked on

Shared kernel with the host

- Syscall exploits can be exploited from within the container
- Solution proposed (seccomp2)

Secure containers with SELinux, AppArmor

- SELinux policy applies to complete container
- Support for SELinux with LXC on a case by case basis
- AppArmor support is work in progress

33

Linux Containers – Feature Overview

Supported in SLES 11 SP2:

- Support for system containers
  - A full SLES11 SP2 installation into a chroot directory structure
- Bridged networking required
- Only SLES11 SP2 supported in container

Why begin adopting now?

Planned for SLES 11 SP3 and future:

- Filesystem copy-on-write (btrfs integration)
- Application containers support
  - Just the application being started within the container
- Easy application containers creation and management
- Research support for AppArmor and LXC

34

Linux Containers – Use Cases (1)

- Hosting business
  - Give a user / developer (root) access without full (root) access to the “real” system.
- Datacenter use
  - Limit applications which have a tendency to grab all resources on a system:
    - Memory (databases)
    - CPU cycles / scheduling (compute intensive applications)
- Oursourcing business
  - Guarantee a specific amount of resources (SLAs!) to a set of applications for a specific customer without more heavy virtualization technologies

35

Linux Containers – Use Cases (2)

- Supporting applications
  - Duplication of issues without endangering a "real" system
- Researching cause-and-effect
  - Reduce unknown/undesirable changes to a host machines
- Performance Virtualization of SLES 11 SP2
- Application virtualization
  - Simultaneous application versions running
    - Run more than one version of software on the same machine
  - Migration of data across version
- Compartmentalization of services
  - Keep access to system resource confined to a container

36

Using LXC

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 3

ATT Live 2012
Session SUS15

Objectives

- LXC Files, Directories and Commands
- Creating LXC Containers
- Mirrored System Containers

38

LXC Files, Directories, and Commands

LXC Container Configuration Files

/etc/lxc/<container_name>/config

-configuration file defining the LXC container

/etc/lxc/<container_name>/fstab

-fstab for the container

-typically specified /proc and /sys mount points

40

Notes:

Container config File

```
lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up
lxc.utsname = basic-sles
```

Network configuration

```
lxc.tty = 4
lxc.pts = 1024
lxc.rootfs = /var/lib/lxc/basic-sles/rootfs
lxc.mount  = /var/lib/lxc/basic-sles/fstab
```

Root FS, TTY/PTY configuration

```
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
# /dev/{,u}random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
lxc.cgroup.devices.allow = c 136:* rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# rtc
lxc.cgroup.devices.allow = c 254:0 rwm
```

cgroup device access configuration

41

Notes:

LXC Container Files

/var/lib/lxc/<container_name>/rootfs/

-directory containing the root file system for the container

42

Notes:

LXC Administration Files

/etc/init.d/lxc (rclxc)

-script use to start/stop LXC containers at boot/shutdown time

/usr/lib64/lxc/templates/

-directory containing template scripts used to create containers

43

Notes:

Common LXC Commands

lxc-create	-creates a new container from a template script
lxc-ls	-lists containers existing on the system
lxc-start	-starts a container
lxc-stop	-stops a container
lxc-info	-displays information about a container
lxc-console	-launches a console for the specified container

44

Notes:

LXC Container Files and Directories

/var/lib/lxc/<container_name>/config

-configuration file defining the LXC container

/var/lib/lxc/<container_name>/fstab

-fstab for the container

-typically specified /proc and /sys mount points

/var/lib/lxc/<container_name>/rootfs/

-directory containing the root file system for the container

45

Notes:

Creating LXC Containers

Create a Linux Container From Scratch

- Prepare the network configuration on the host
- Create the Linux container
- Start, access and stop your container

47

Notes:

Prepare the Network Configuration

- Install the following packages:
  - lxc
  - bridge-utils
- Using YaS, deconfigure the existing physical NIC
- Create a bridge and add the physical network interface to it
- If SuSEFirewall is active, assign the br0 interface to the proper zone, such as external

48

Notes:

Set Up the Linux Container (1)

- Create basic configuration file, such as /tmp/my_container.conf
  - contains at least the basic networking config:

```
lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up
lxc.network.hwaddr = 00:30:6E:01:23:45
lxc.network.ipv4  = 192.168.1.10
lxc.network.name = eth0
# The container name
lxc.utsname = my_container
```

49

Notes:

Set Up the Linux Container (2)

- Decide on the OS template (such as sles or opensuse
  - /usr/lib64/lxc/templates/lxc-**<template>**
- Create container

```
lxc-create -n <container name> -f <config file> -t
<template>
```

- The container will be created in

  `/var/lib/lxc/`*`container`*`/`

  (Filling the container with the necessary files will take a bit of time.)

50

Notes:

Set Up the Linux Container (3)

- Change the root password by doing the following
  - `chroot /var/lib/lxc/container/rootfs`
  - `passwd root`
- Create a non-privileged user
  - `useradd -m geeko`
  - `passwd geeko`
- Leave the chroot environment by entering `exit`

51

Notes:

Start, Access, and Stop Your Container

- Start the container:
  - `lxc-start -n container_name`
- Connect to the container:
  - `lxc-console -n container_name`
  - Disconnect from the console by pressing Ctrl+a q
- Stop the container:
  - `lxc-stop -n container_name`
  - You can also shut down the container from inside the container with the `shutdown` or `init 0` command

52

Notes:

LAB 2-1: Create a Linux Container

Summary: In this exercise, you create a simple container based on a minimal installation of SLES11-SP2.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

53

Lab Notes:

Mirrored System Containers

2 Type of Containers

- Newly installed systems
  - Typical method for LXC
  - Requires installation media
  - Acts as a new "virtualized" instance of the installed OS
- System Mirrors
  - Replicas of the existing system
  - Do not REQUIRE installation media
    - Although it can use it
  - Act as virtualized "FORKS" of the existing system

55

Notes:

Mirroring SLE into LXC: lxc-jailbird

- Script provided by SUSE ATT
  - Designed to create system mirrors in LXC
  - Follows many of the concepts of the jailbird PWS function
  - Makes mirroring the system very easy
- Co-Written by ATT Engineers
  - Brandon Heaton
  - Björn Lotz
  - ATT Live 2012 Debut
    - Let's try it out!

56

Notes:

LAB 2-2: Mirror a System in LXC

Summary: In this exercise, you create a container and mirror your existing system into the container using LXC, Cgroups and the lxc-jailbird.sh script provided by SUSE ATT.

Special Instructions You must have the lxc-jailbird.sh script to perform this exercise.

Duration: 30 min.

57

Lab Notes:

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Contents

Introduction to Cgroups

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 1

ATT Live 2012
Session SUS15

Objectives

- Kernel Control Groups
- Using Cgroups in Linux
- Building and Using Linux Containers with LXC
- Create Cgroups when the System Starts

5

Kernel Control Groups

What Are Control Groups?

Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behavior.

- cgroup is another name for Control Groups
- Partition tasks (processes) into a one or many groups of tree hierarchies
- Associate a set of tasks in a group to a set subsystem parameters
- Subsystems provide the parameters that can be assigned
- Tasks are affected by the assigning parameters

7

Definitions (from the Kernel-Docu)

- cgroup
- Subsystem (also called Resource Controller or simply Controller)
- Hierarchy

8

A *cgroup* associates a set of tasks with a set of parameters for one or more subsystems.
•A *subsystem* is a module that makes use of the task grouping facilities provided by cgroups to treat groups of tasks in particular ways. A subsystem is typically a "resource controller" that schedules a resource or applies per-cgroup limits, but it may be anything that wants to act on a group of processes, e.g. a virtualization subsystem.
•A *hierarchy* is a set of cgroups arranged in a tree, such that every task in the system is in exactly one of the cgroups in the hierarchy, and a set of subsystems; each subsystem has system-specific state attached to each cgroup in the hierarchy. Each hierarchy has an instance of the cgroup virtual filesystem associated with it.:

Example of the Capabilities of a cgroup

Consider a large university server with various users - students, professors, system tasks etc. The resource planning for this server could be along the following lines:

CPUs

Top cpuset (20%)

/ \

CPUSet1 CPUSet2

| |

(Profs) (Students)

60% 20%

Memory

Professors = 50%

Students = 30%

System = 20%

Disk I/O

Professors = 50%

Students = 30%

System = 20%

Network I/O

WWW browsing = 20%

/ \

Prof (15%) Students (5%)

Network File System (60%)

Others (20%)

Source: /usr/src/linux/Documentation/cgroups/cgroups.txt

9

Control Group Subsystems

Two types of subsystems

- Isolation and special controls
  - cpuset, namespace, freezer, device, checkpoint/restart
- Resource control
  - cpu(scheduler), memory, disk i/o, network

Source: http://jp.linuxfoundation.org/jp_uploads/seminar20081119/CgroupMemcgMaster.pdf

10

Using Cgroups in Linux

Initializing Cgroups

- Run `/etc/init.d/boot.cgroup start`
- To activate cgroups automatically at system start, enter `chkconfig boot.cgroup on`
- This will populate the `/sys/fs/cgroup` directory
- Enter `mount` to see how it's accomplished
- Each available subsystem is represented by a subdirectory in `/sys/fs/cgroup`, such as `blkio cpu cpuacct cpuset devices freezer memory net_cls perf_event`

12

Notes:

Administering Cgroups

- cgroups are administered by creating subdirectories in the subsystem directories (such as `/sys/fs/cgroup/cpuset`) and changing files in those subdirectories
- There are tools that facilitate this, but the “raw” way using `mkdir` and `echo` helps understand the underlying mechanics

13

Notes:

Cgroups Example 1

- You want processes to share the time of the cpu in the ratio 6:4
- To do that, you create directories in the cpu subsystem and change files in that subdirectory (the subdirectory is automatically populated with several files after you create it)
- The commands are contained in the notes section

Note: The use of taskset simplifies the example as it binds the xterm calls to the first core. You could use the cpuset subsystem to achieve the same effect, but that would require additional commands. If you don't bind them to the same core, you don't see the effect in top, as they will probably run on different cores, using all the available cpu time.

The following commands can be used:
Create two groups:
cd /sys/fs/cgroup/cpu
mkdir higherload lowerload

Set the values:
echo 6 > higherload/cpu.shares
echo 4 > lowerload/cpu.shares

Start processes and assign them to one of the groups
taskset -c 0 xterm -bg orange &
taskset -c 0 xterm -bg green &

In the orange xterm, enter
echo $$ > /sys/fs/cgoup/cpu/higherload/tasks
In the green xterm, enter
echo $$ > /sys/fs/cgoup/cpu/lowerload/tasks

In the the terminal window, enter top
In each of the xterms, enter
md5sum /dev/urandom

Cgroups Example 2

- You want processes to share the time of the cpu in the ratio 6:4
- There are various `cg*` commands that simplify your task:
  - `cgcreate`: Creates a new cgroup
  - `cgset`: Set parameters within a cgroup
  - `cgexec`: Executes a command and puts it in a cgroup
  - `cgclear`: Removes all cgroups
- See the notes section for commands

15

The following commands can be used, using cpuset as well:
Create two groups:
cgcreate -g cpu,cpuset:higherload
cgcreate -g cpu,cpuset:lowerload

Set the values:
cgset -r cpu.shares=6 -r cpuset.cpus=0 higherload
cgset -r cpu.shares=4 -r cpuset.cpus=0 lowerload
The following is required for cpusets.cpus to work properly (see man cpuset)
cgset -r cpuset.mems=0 higherload
cgset -r cpuset.mems=0 lowerload

Start processes and assign them to one of the groups
cgexec -g cpu,cpuset:higherload xterm -bg orange &
cgexec -g cpu,cpuset:lowerload xterm -bg green &

In the the terminal window, enter top
In each of the xterms, enter
md5sum /dev/urandom

LAB 1-1: Use Linux Control Groups

Summary: In this exercise, you install, enable, and use Linux control groups.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

16

Lab Notes:

Create cgroups when the System Starts

Administering cgroups

- Commandline with mkdir and echo
- Commandline with cg* commands
- Startscript in /etc/init.d/ and /etc/cgconfig.conf to make cgroups persist over reboots

18

Notes:

cgconfig.conf

- groups section defines a cgroup
  - Permissions can be set for the task file (who may add tasks) and the rest of the files
  - Values are set for the individual ressources within the subsystems (such as cpu, cpuset, blkio, etc.)
- mount sections define were the cgroup subsystems are mounted (in SLES11SP2 this is usally done by the /etc/init.d/boot.cgroup script)

19

```
group daemons/www {
      perm {
            task {
                  uid = root;
                  gid = webmaster;
            }
            admin {
                  uid = root;
                 gid = root;
            }
      }
      cpu {
            cpu.shares = 1000;
      }
}
```

```
group daemons/ftp {
	perm {
		task {
			uid = root;
			gid = ftpmaster;
		}
		admin {
			uid = root;
			gid = root;
		}
	}
	cpu {
		cpu.shares = 500;
	}
}

#mount {
#	cpu = /mnt/cgroups/cpu;
#	cpuacct = /mnt/cgroups/cpuacct;
#}
```

Adding Processes to the cgroups

- /etc/init.d/cgconfig start
- chkconfig cgconfig on
- Modify service start scripts to put daemons into cgroups, for instance by adding cgexec as part of the start up of the daemon

21

Notes:

LAB 1-2: Configure cgconfig.conf

Summary: In this exercise, you configure cgroups so they get created when the system boots.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

22

Lab Notes:

Introduction to LXC

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 2

ATT Live 2012
Session SUS15

Objectives

- Introduction to Linux Containers and LXC
- Building and Using Linux Containers with LXC

24

Introduction to Linux Containers and LXC

What are Linux Containers

- Instances of Kernel-based isolation that go beyond simple chroot jail
- Not virtual machines exactly
- Rather virtual ENVIRONMENTS similar to chroot
- Provides more isolation than chroot
- Leverages Linux Control groups for container isolation and resource and process limits

26

Notes:

What is LXC?

- Project to create and manage Linux Containers
- Templates for creating virtualized containers
- Provides operating System level virtualization
  - (Without a "hypervisor" virtualization layer)
- Allows for multiple isolated server installs on a single host
- Only one kernel running on the host, not a separate kernel in each virtual machine as with Xen/KVM/VMware virtualization

27

Notes:

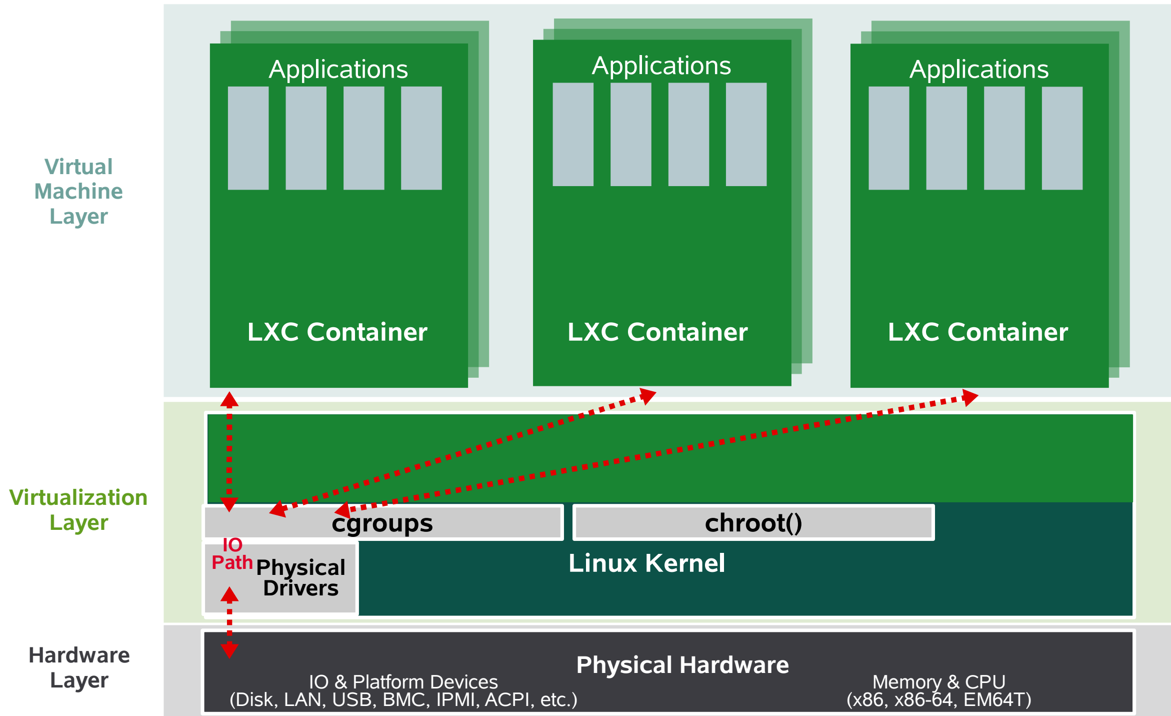
What are Linux Containers (LXC) (2)

- Does not provide virtual machines but rather virtual environments similar to chroot
- Provides more isolation than chroot
- Leverages Linux Control groups for container isolation and resource and process limits

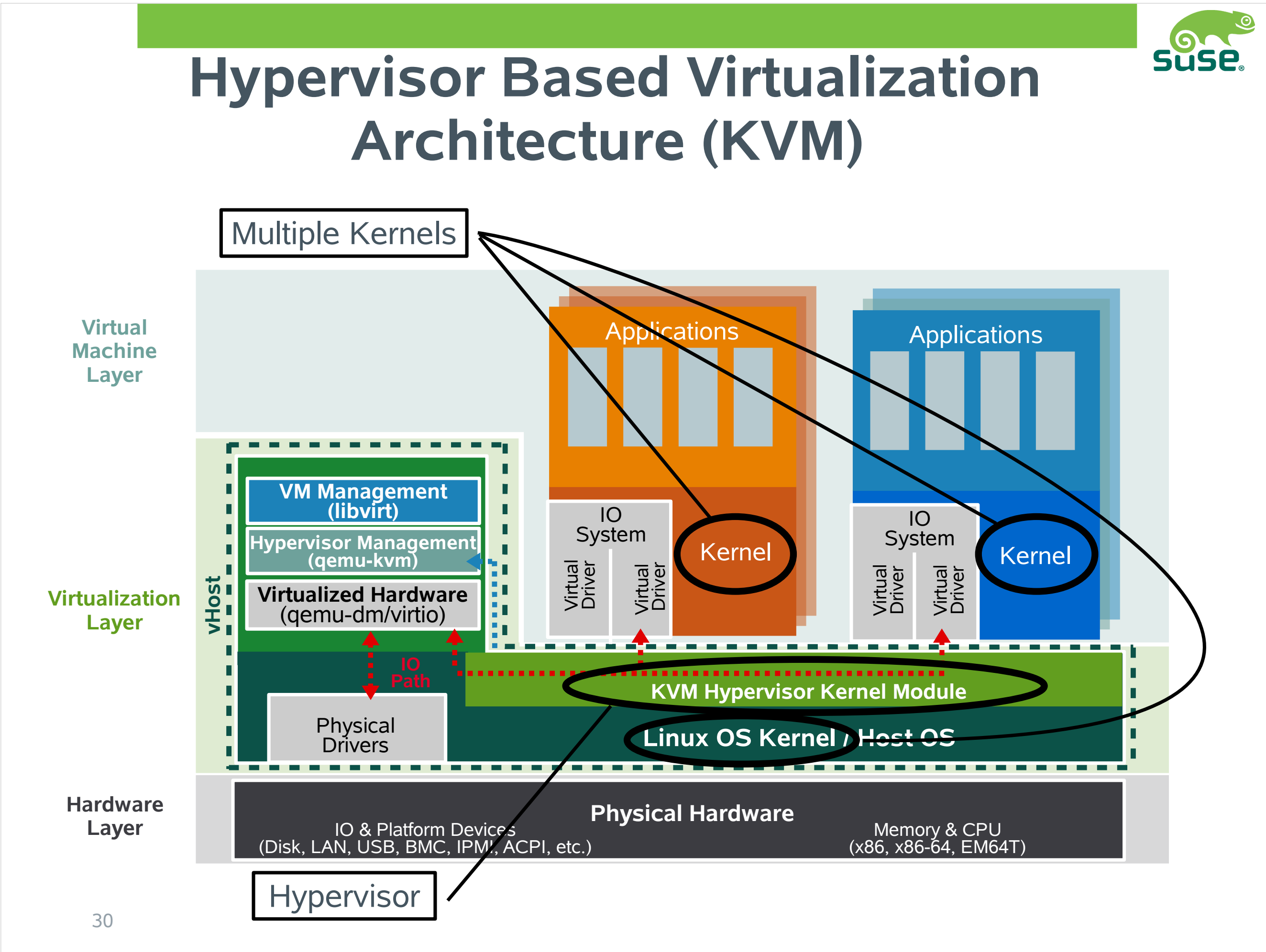
28

Notes:

Container Based Virtualization Architecture



29



With KVM (or the Kernel Virtual Machine), a kernel module is loaded into the Linux kernel that turns it into a hypervisor. KVM would essentially be a Type I Hypervisor because it is running directly on top of the hardware.
With KVM, the Linux kernel becomes a “fat” hypervisor because it not only mediates access to the underlying hardware but also loads physical drivers and shares access to the underlying hardware devises with virtual drivers.
Device emulation and VM management are handled by a modified version of QEmu running in user space.
Because KVM is a hypervisor based virtualization architecture multiple kernels can run on the hardware at the same time. These kernels can be virtualy anu operating system, not just linux.

Building and Using Linux Containers with LXC

Linux Containers – Virtualization

- OS Level Virtualization – i.e. virtualization without a hypervisor (also known as “Lightweight virtualization”)
- Similar technologies include: Solaris Zones, BSD Jails,Virtuozzo or OpenVZ
- Advantages of OS Level Virtualization
  - Minor I/O overhead
  - Storage advantages
  - Dynamic changes to parameters without reboot
  - Combining virtualization technologies
- Disadvantages
  - Higher impact of a crash, especially in the kernel area
  - Unable run another OS that cannot use the host's kernel

32

Linux Containers – Security

Missing user namespaces

- Allows evading from containers
- This is actively being worked on

Shared kernel with the host

- Syscall exploits can be exploited from within the container
- Solution proposed (seccomp2)

Secure containers with SELinux, AppArmor

- SELinux policy applies to complete container
- Support for SELinux with LXC on a case by case basis
- AppArmor support is work in progress

33

Linux Containers – Feature Overview

Supported in SLES 11 SP2:

- Support for system containers
  - A full SLES11 SP2 installation into a chroot directory structure
- Bridged networking required
- Only SLES11 SP2 supported in container

Why begin adopting now?

Planned for SLES 11 SP3 and future:

- Filesystem copy-on-write (btrfs integration)
- Application containers support
  - Just the application being started within the container
- Easy application containers creation and management
- Research support for AppArmor and LXC

34

Linux Containers – Use Cases (1)

- Hosting business
  - Give a user / developer (root) access without full (root) access to the “real” system.
- Datacenter use
  - Limit applications which have a tendency to grab all resources on a system:
    - Memory (databases)
    - CPU cycles / scheduling (compute intensive applications)
- Oursourcing business
  - Guarantee a specific amount of resources (SLAs!) to a set of applications for a specific customer without more heavy virtualization technologies

35

Linux Containers – Use Cases (2)

- Supporting applications
  - Duplication of issues without endangering a "real" system
- Researching cause-and-effect
  - Reduce unknown/undesirable changes to a host machines
- Performance Virtualization of SLES 11 SP2
- Application virtualization
  - Simultaneous application versions running
    - Run more than one version of software on the same machine
  - Migration of data across version
- Compartmentalization of services
  - Keep access to system resource confined to a container

36

Using LXC

SUSE Linux Enteprise 11 SP2
Virtualization with LXC
Section 3

ATT Live 2012
Session SUS15

Objectives

- LXC Files, Directories and Commands
- Creating LXC Containers
- Mirrored System Containers

38

LXC Files, Directories, and Commands

LXC Container Configuration Files

/etc/lxc/<container_name>/config

-configuration file defining the LXC container

/etc/lxc/<container_name>/fstab

-fstab for the container

-typically specified /proc and /sys mount points

40

Notes:

Container config File

```
lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up
lxc.utsname = basic-sles

lxc.tty = 4
lxc.pts = 1024
lxc.rootfs = /var/lib/lxc/basic-sles/rootfs
lxc.mount  = /var/lib/lxc/basic-sles/fstab

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
# /dev/{,u}random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
lxc.cgroup.devices.allow = c 136:* rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# rtc
lxc.cgroup.devices.allow = c 254:0 rwm
```

Network configuration

Root FS, TTY/PTY configuration

cgroup device access configuration

41

Notes:

LXC Container Files

/var/lib/lxc/<container_name>/rootfs/

-directory containing the root file system for the container

42

Notes:

LXC Administration Files

/etc/init.d/lxc (rclxc)

-script use to start/stop LXC containers at boot/shutdown time

/usr/lib64/lxc/templates/

-directory containing template scripts used to create containers

43

Notes:

Common LXC Commands

lxc-create	-creates a new container from a template script
lxc-ls	-lists containers existing on the system
lxc-start	-starts a container
lxc-stop	-stops a container
lxc-info	-displays information about a container
lxc-console	-launches a console for the specified container

44

Notes:

LXC Container Files and Directories

/var/lib/lxc/<container_name>/config

-configuration file defining the LXC container

/var/lib/lxc/<container_name>/fstab

-fstab for the container

-typically specified /proc and /sys mount points

/var/lib/lxc/<container_name>/rootfs/

-directory containing the root file system for the container

45

Notes:

Creating LXC Containers

Create a Linux Container From Scratch

- Prepare the network configuration on the host
- Create the Linux container
- Start, access and stop your container

47

Notes:

Prepare the Network Configuration

- Install the following packages:
  - lxc
  - bridge-utils
- Using YaS, deconfigure the existing physical NIC
- Create a bridge and add the physical network interface to it
- If SuSEFirewall is active, assign the br0 interface to the proper zone, such as external

48

Notes:

Set Up the Linux Container (1)

- Create basic configuration file, such as /tmp/my_container.conf
  - contains at least the basic networking config:

```
lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up
lxc.network.hwaddr = 00:30:6E:01:23:45
lxc.network.ipv4  = 192.168.1.10
lxc.network.name = eth0
# The container name
lxc.utsname = my_container
```

49

Notes:

Set Up the Linux Container (2)

- Decide on the OS template (such as sles or opensuse
  - /usr/lib64/lxc/templates/lxc-**<template>**
- Create container

```
lxc-create -n <container name> -f <config file> -t
<template>
```

- The container will be created in

  `/var/lib/lxc/`*`container/`*

  (Filling the container with the necessary files will take a bit of time.)

50

Notes:

Set Up the Linux Container (3)

- Change the root password by doing the following
  - `chroot /var/lib/lxc/container/rootfs`
  - `passwd root`
- Create a non-privileged user
  - `useradd -m geeko`
  - `passwd geeko`
- Leave the chroot environment by entering `exit`

51

Notes:

Start, Access, and Stop Your Container

- Start the container:
  - `lxc-start -n container_name`
- Connect to the container:
  - `lxc-console -n container_name`
  - Disconnect from the console by pressing Ctrl+a q
- Stop the container:
  - `lxc-stop -n container_name`
  - You can also shut down the container from inside the container with the `shutdown` or `init 0` command

52

Notes:

LAB 2-1: Create a Linux Container

Summary: In this exercise, you create a simple container based on a minimal installation of SLES11-SP2.

Special Instructions

Use the following values in the exercise:

(none)

Duration: 20 min.

53

Lab Notes:

Mirrored System Containers

2 Type of Containers

- Newly installed systems
  - Typical method for LXC
  - Requires installation media
  - Acts as a new "virtualized" instance of the installed OS
- System Mirrors
  - Replicas of the existing system
  - Do not REQUIRE installation media
    - Although it can use it
  - Act as virtualized "FORKS" of the existing system

55

Notes:

Mirroring SLE into LXC: lxc-jailbird

- Script provided by SUSE ATT
  - Designed to create system mirrors in LXC
  - Follows many of the concepts of the jailbird PWS function
  - Makes mirroring the system very easy
- Co-Written by ATT Engineers
  - Brandon Heaton
  - Björn Lotz
  - ATT Live 2012 Debut
    - Let's try it out!

56

Notes:

LAB 2-2: Mirror a System in LXC

Summary: In this exercise, you create a container and mirror your existing system into the container using LXC, Cgroups and the lxc-jailbird.sh script provided by SUSE ATT.

Special Instructions You must have the lxc-jailbird.sh script to perform this exercise.

Duration: 30 min.

57

Lab Notes: