Using Linux Containers as a Virtualization Option

Michal Svec
Product Manager
msvec@suse.com

Mike Friesenegger
Sales Engineer
mfriesenegger@suse.com
Containers
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

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Minor I/O overhead</td>
<td>- Higher impact of a crash, especially in the kernel area</td>
</tr>
<tr>
<td>- Storage advantages</td>
<td>- Unable to run another OS that cannot use the host's kernel</td>
</tr>
<tr>
<td>- Dynamic changes to parameters</td>
<td></td>
</tr>
<tr>
<td>without reboot</td>
<td></td>
</tr>
<tr>
<td>- Combining virtualization</td>
<td></td>
</tr>
<tr>
<td>technologies</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

Control Groups

Introduction to Linux Containers (LXC)

Linux Containers Demo

Questions
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.

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

<table>
<thead>
<tr>
<th>CPUs</th>
<th>Memory</th>
<th>Network I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top cpuset (20%)</td>
<td>Professors = 50%</td>
<td>WWW browsing = 20%</td>
</tr>
<tr>
<td></td>
<td>Students = 30%</td>
<td>/</td>
</tr>
<tr>
<td>CPUSet1</td>
<td>System = 20%</td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>(Profs) = 60%</td>
<td>Disk I/O</td>
</tr>
<tr>
<td></td>
<td>(Students) = 20%</td>
<td>Professors = 50%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Students = 30%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System = 20%</td>
</tr>
</tbody>
</table>

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

Building and Using Linux Containers with LXC
Linux Containers – Architecture

- Physical Hardware:
  - Memory & CPU (x86, x86-64, EM64T)
  - IO & Platform Devices (Disk, LAN, USB, BMC, IPMI, ACPI, etc.)

- Physical Drivers

- Applications

- LXC Container

- Container Layer

- cgroups

- chroot() (Path)

- bridging

- Linux Kernel

- Hardware Layer
Linux Containers – Security

- User namespaces
  - Prevents evading from containers

- Shared kernel with the host
  - Syscall exploits can be exploited from within the container
  - Solution is in Linux kernel since 3.5 (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 ready upstream
Linux Containers – Feature Overview

- Only SUSE Linux Enterprise Server 11 SP3 supported in container
- Support for system containers
- A full SUSE Linux Enterprise Server installation into a chroot directory structure
- Resource control using cgroups
- Bridged networking required
Understanding Solaris Containers

• Also known as a Solaris Zone

• Two components of a Solaris Container
  - Solaris Zones software partitioning technology
  - Solaris Resource Management

• Solaris Zones
  - Virtual mapping from the application to the platform resources
  - Application components are isolated from one another

• Solaris Resource Management
  - Allocate, limit and deny available computing resources
  - Generate extended accounting information for analysis, billing, and capacity planning
## Comparing Linux Containers to Solaris Containers

<table>
<thead>
<tr>
<th>Feature</th>
<th>Solaris Containers</th>
<th>LXC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included with OS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zone Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Line Interface</td>
<td>Yes (zonecfg, zoneadm)</td>
<td>Yes (lxc-*)</td>
</tr>
<tr>
<td>GUI</td>
<td>Yes (added purchase)</td>
<td>Yes (YaST module)</td>
</tr>
<tr>
<td>Fault Isolation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Level</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Kernel Level</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Privacy/Security</td>
<td>Yes (inside container)</td>
<td>Yes (inside container)</td>
</tr>
<tr>
<td>Resource Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project/Task Identifiers</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Accounting</td>
<td>Yes</td>
<td>No (evaluating features)</td>
</tr>
<tr>
<td>CPU Control</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Memory Control</td>
<td>Yes</td>
<td>Yes (OOM)</td>
</tr>
<tr>
<td>Network Control</td>
<td>Yes</td>
<td>No (in progress)</td>
</tr>
</tbody>
</table>
Linux Containers – Use Cases

• Hosting Business
  – Give a user/developer (root) access without full (root) access to the “real” system.

• Enterprise Data Center
  – Limit applications which have a tendency to grab all resources on a system:
    – Memory (databases)
    – CPU cycles/scheduling (compute intensive applications)

• Outsourcing business
  – Guarantee a specific amount of resources (SLAs!) to a set of applications for a specific customer without more heavy virtualization technologies
Linux Containers – Outlook

• Cloud use case
  – Bare metal provisioning through the cloud interfaces

• Integration with libvirt: libvirt-lxc
  – Integration with virtualization management tools, incl. cloud

• Application containers
  – More effective storage use
  – Easy creation and management

• Research further container-based technologies
  – systemd, docker.io, …

• SELinux and AppArmor support for LXC

• File system copy-on-write (btrfs integration)
Linux Containers Demo
Unpublished Work of SUSE. All Rights Reserved.  
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer  
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.