LinuX Containers

Bogdan Purcăreaţă

April 26, 2013

Concepts
- Operating Systems
- Virtualization

LinuX Containers

Usage
- Schematics
- Demo
- Benchmarks

Relevance
- Spread
- Use Cases

QA

IT trends

- more resources
  - better hardware at lower costs
  - higher standards for software quality
- more users
  - better education available at an earlier age
  - shared access to the same device
- data consolidation
  - data warehousing
  - cloud computing
  - separate access policies
- increased flexibility
  - differentiated access
  - straighforward configuration
  - focus on usability

freescale™

OS Recap

- Resources
  - CPU
  - memory
  - peripherals
- Structures
  - the Scheduler
  - the Pager
  - Filesystems
- the Kernel
  - handles hardware
  - exposes capabilities
  - manages resources

Figure: The Memory Pager

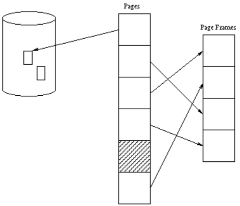
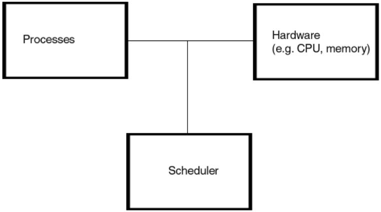


Figure: The Scheduler



Virtualization

- Key aspects:
  - simulation of software and/or hardware
  - virtual machines
  - autonomous computing
  - utility computing
- Advantages:
  - better resource usage
  - lower running costs
  - application sandboxing
- Concerns:
  - management
  - isolation
  - performance
  - applicability

OS-level Virtualization

- one host
- multiple running OS instances
- rootfs, system libs, binaries

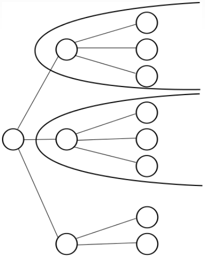
OS instance = a process hierarchy
OS level virtualization = **partitioning** the process tree
Advantage: **close to 0% performance overhead**
Flaw: **same kernel**

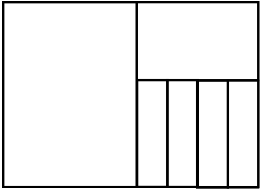
LinuX Containers Overview

- Features:
  - mainline kernel support
  - application vs. system containers
  - in active development
- Components:
  - *kernel features*
  - *userspace tools*
    - binaries, scripts
  - *configuration files*
    - container-host interface
  - *template files*
    - container applications

Kernel Support

- Namespaces:
  - wrap system resources in an abstraction
  - processes see the resource as their own
  - isolation between processss in different namespaces
- Control Groups
  - resource management among processes
  - hierarchical support
  - interaction with resource responsible structures:
    - the scheduler
    - the pager





Sample Process Hierarchy

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)----+-init(2972)
        |                                 |-sh(2971)
        |                                 `-syslogd(2969)
        |
        |
        |-lxc-start(2974)---init(2976)----+-init(2982)
        |                                 |-sh(2981)
        |                                 `-syslogd(2979)
        |
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

freescale™

Process IDs

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)(1)-+-init(2972)(7)
        |                                 |-sh(2971)(6)
        |                                 `-syslogd(2969)(4)
        |
        |
        |-lxc-start(2974)---init(2976)(1)-+-init(2982)(7)
        |                                 |-sh(2981)(6)
        |                                 `-syslogd(2979)(4)
        |
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

Namespace Segregation

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)--- init(2966)(1)-+-init(2972)(7)
        |                                  |-sh(2971)(6)
        |                                  `-syslogd(2969)(4)
        |                    PID Namespace 1
        |
        |-lxc-start(2974)--- init(2976)(1)-+-init(2982)(7)
        |                                  |-sh(2981)(6)
        |                                  `-syslogd(2979)(4)
        |                    PID Namespace 2
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

freescale™

Filesystem Segregation

"chroot on steroids"

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)          root: /var/lib/lxc/foo1/rootfs/
        |-lxc-start(2964)--- init(2966)(1)-+-init(2972)(7)
        |                                  |-sh(2971)(6)
        |                                  `-syslogd(2969)(4)
        |                    PID Namespace 1
        |                    root: /var/lib/lxc/foo1/rootfs/
        |-lxc-start(2974)--- init(2976)(1)-+-init(2982)(7)
        |                                  |-sh(2981)(6)
        |                                  `-syslogd(2979)(4)
        |                    PID Namespace 2
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

CPU Partitioning

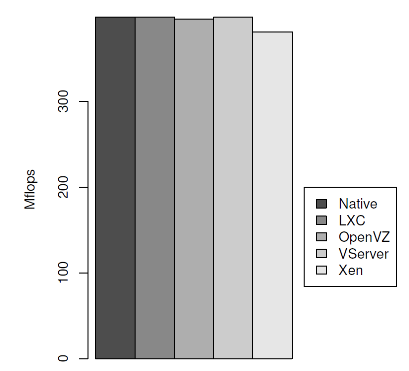
```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)        root: /var/lib/lxc/foo1/rootfs/
  ,-----|-lxc-start(2964)--- init(2966)(1)-+-init(2972)(7)
  |     |       25%                        |-sh(2971)(6)
  |     |                                  `-syslogd(2969)(4)
  |     |                    PID Namespace 1
  |     |                    root: /var/lib/lxc/foo1/rootfs/
1 core  |-lxc-start(2974)--- init(2976)(1)-+-init(2982)(7)
  |     |       75%                        |-sh(2981)(6)
  |     |                                  `-syslogd(2979)(4)
  |     |                    PID Namespace 2
  `-----|------------------------------------------------------
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

Demo

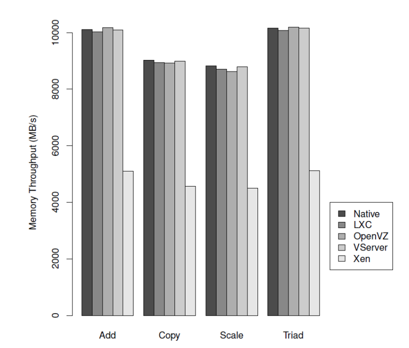
1. start 2 containers
2. check PIDs
3. assign them a single core on the host
4. balance CPU usage 25% - 75%

System Performance

CPU performance
Linpack



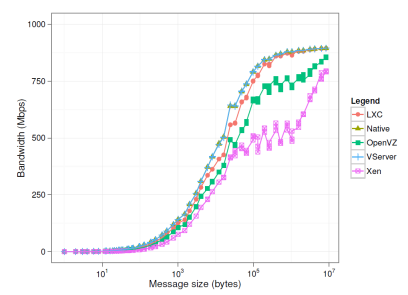
Mem throughput
Stream



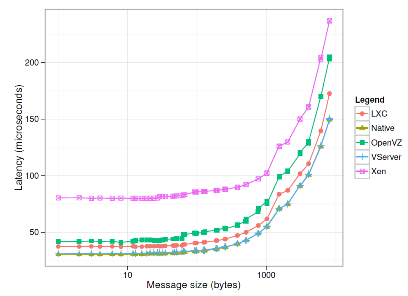
freescale™

Networking Performance

Bandwidth
NetPIPE



Latency
NetPIPE



Isolation

PERFORMANCE ISOLATION FOR LU APPLICATION. THE RESULTS REPRESENT HOW MUCH THE APPLICATION PERFORMANCE IS IMPACTED BY DIFFERENT STRESS TESTS IN ANOTHER VM/CONTAINER. DNR MEANS THAT APPLICATION WAS NOT ABLE TO RUN.

	LXC	OpenVZ	VServer	Xen
CPU Stress	0	0	0	0
Memory	88.2%	89.3%	20.6%	0.9%
Disk Stress	9%	39%	48.8%	0
Fork Bomb	DNR	0	0	0
Network Receiver	2.2%	4.5%	13.6%	0.9%
Network Sender	10.3%	35.4%	8.2%	0.3%

Popularity

- running on:
  - major distros: Fedora, Debian, Ubuntu, ...
  - Android
  - virtually any system running Linux >= 2.6.26
- integrated with high(er) level tools:
  - `docker.io` - The Linux Container Runtime
  - `libvirt.org` - The Virtualization API
  - `criu.org` - Checkpoint-Restart in Userspace
- maintained by both kernel and userspace developers

Use Cases

- general:
  - server replication
  - application sandboxing
  - legacy software support
  - live migration
  - GPL insulation
- embedded (networking, smartphones):
  - separate traffic from different departments
  - separate QoS policies
  - run RTOS and HLOS at the same time

freescale™

Freescale USDPAA in Containers

- DPAA - DataPath Acceleration Architecture
  - used in dedicated networking equipment
  - HW architecture providing high networking capabilities
  - traffic shaping, package accelerators, cryptography engine
- USDPAA - User Space DPAA
  - userspace drivers based on the uio framework
  - increased flexibility in application development
  - reduced risk in bugging in kernel
  - better error handling and system protection
  - performance overhead
- multiple USDPAA instances in containers
  - improved isolation
  - additional protection layer
  - finer resource tuning

freescale™

References

- OpenVZ's: Why does the Network Namespace Suck and How to Make It Suck Faster?
- IEEE: Performance Evaluation of Container-based Virtualization for High Performance Computing Environments
- http://www.pcgameshardware.com/screenshots/250x375/2009/05/Virtual_Windows_XP_Logo.png
- http://2.bp.blogspot.com/-47sakFH6uSw/UXgrhNqYF8I/AAAAAAAAHzQ/0W8zFVgR--w/s1600/lxc.png

Thank you!

Questions?