Slides

with subtitle

Bogdan Purcareata

April 25, 2013

Why?

What?
- Virtualization
- LinuX Containers

How?
- SO Concepts
- Control Groups
- Namespaces

Usage
- Schematics
- Demo
- Benchmarks

IT trends

- more resources
  - high performance technology available at lower costs
  - software optimizations for resource usage
- more users
  - devices are everywhere
  - shared access
- data consolidation
  - rise of data centers
  - cloud computing
  - data belonging to different accounts in the same place
- increased flexibility
  - easier to access
  - easier to configure
  - easier to use

Embedded World

- Networking
  - traffic belonging to different departments on the same device
  - QoS policy for each container
- Smartphones
  - separate RTOS (Real Time OS) from HLOS (High Level OS)
  - run legacy applications
  - separate account privileges

Network

- device delegation - move one interface to another namespace
- create virtual device inside that namespace
  - veth - virtual ethernet tunnel
  - vlan
    - 802.1Q - dedicated VLAD ID in packet header
    - MAC VLAN - uses VLAN device MAC as ID

References

- why network namespace sucks and how to make it suck faster
- Performance Evaluation of Container-based Virtualization for High Performance Computing Environments

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)
```

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

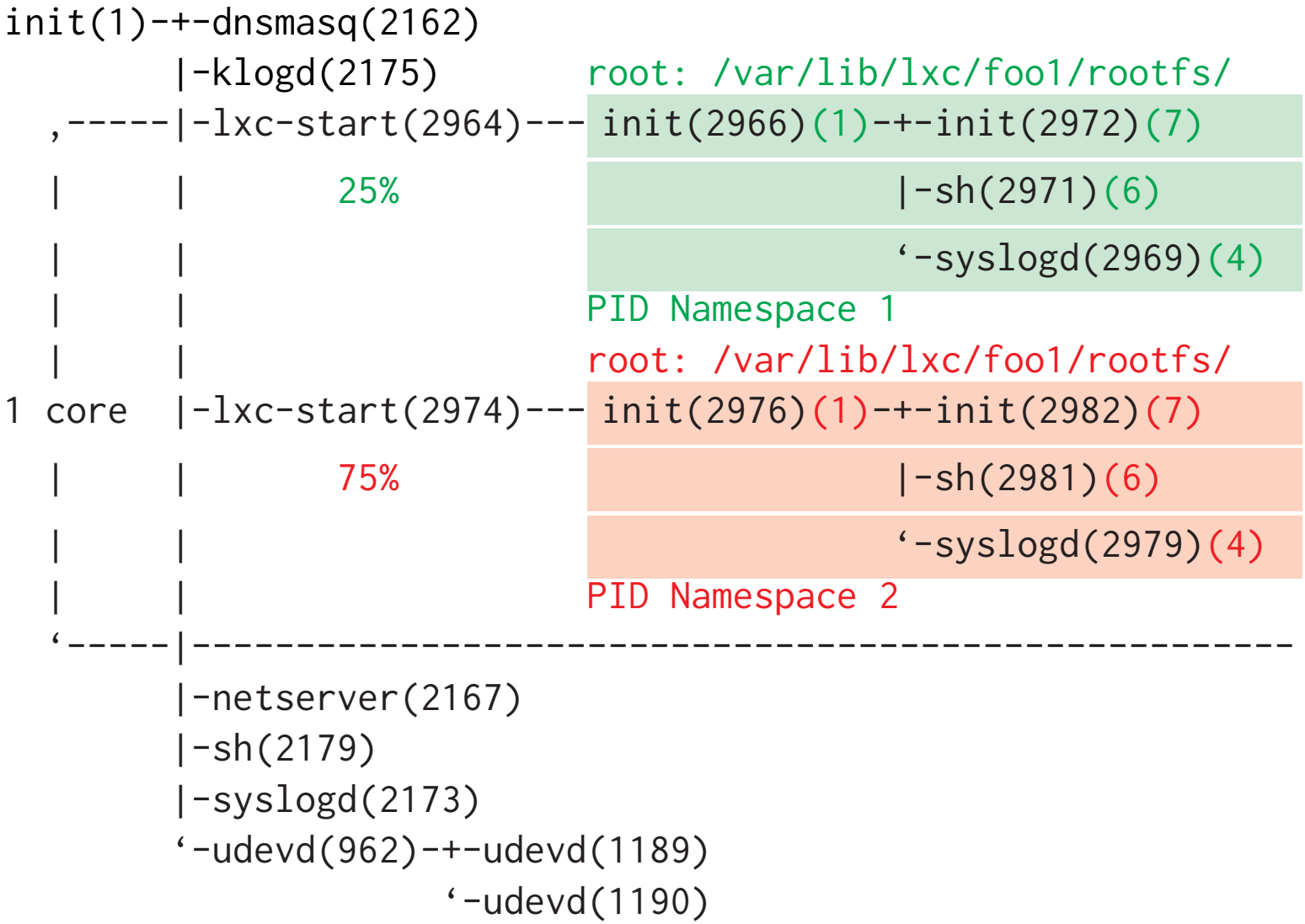
```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)--- init(2966)(1)-+-init(2972)(7)
        |                                  |-sh(2971)(6)
        |                                  `-syslogd(2969)(4)
        |                    PID Namespace 1
        |
        |-lxc-start(2974)--- 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)
```

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

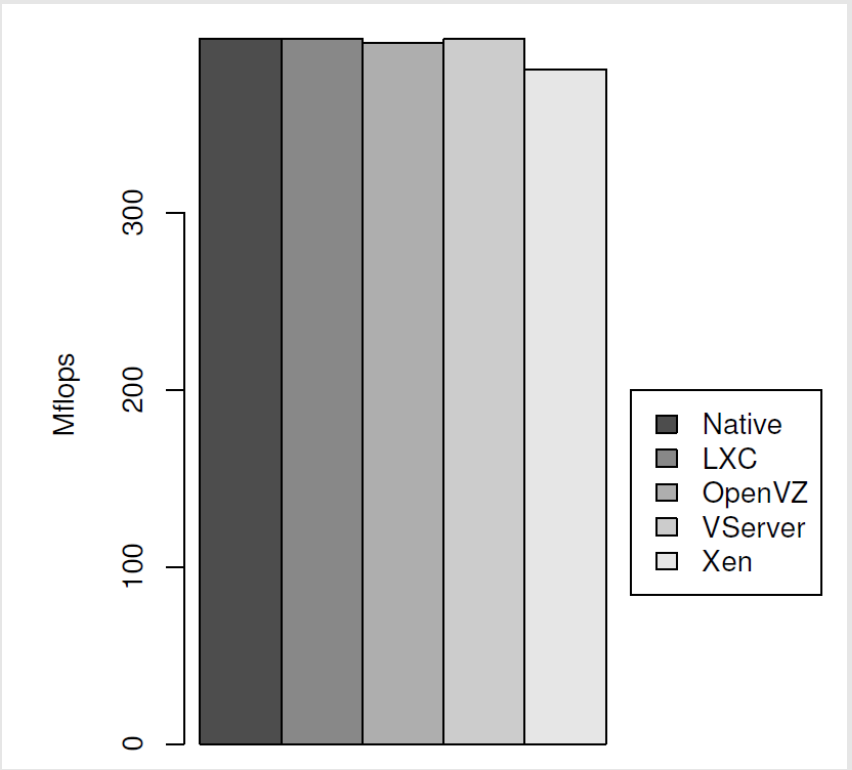


Demo

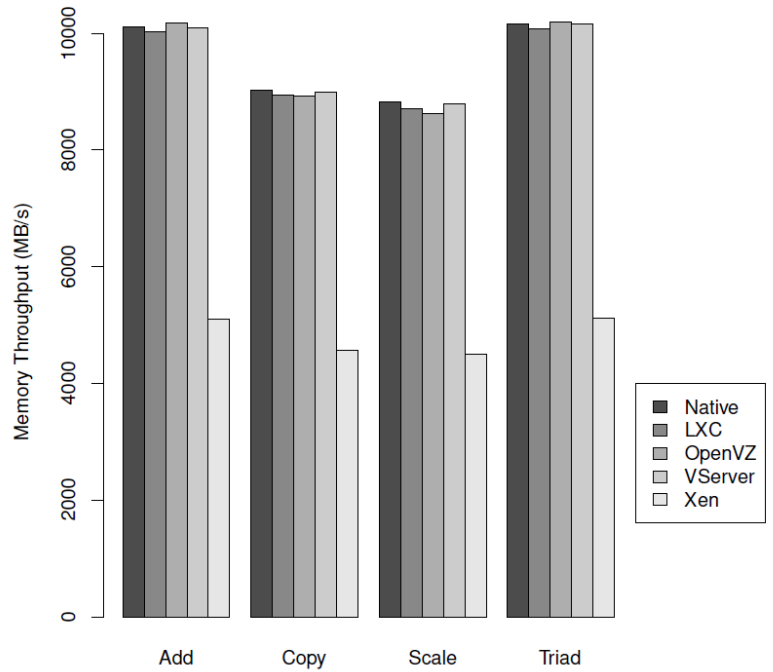
1. start 2 containers
2. check PIDs
3. assign them a single core on the host
4. balance CPU usage 25% - 75%

System Performance

CPU performance
Linpack

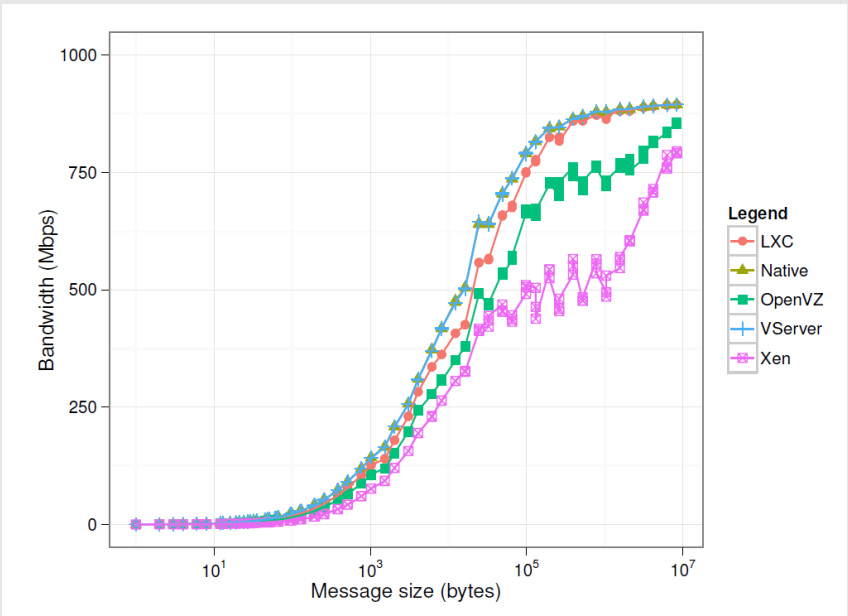


Mem throughput
Stream

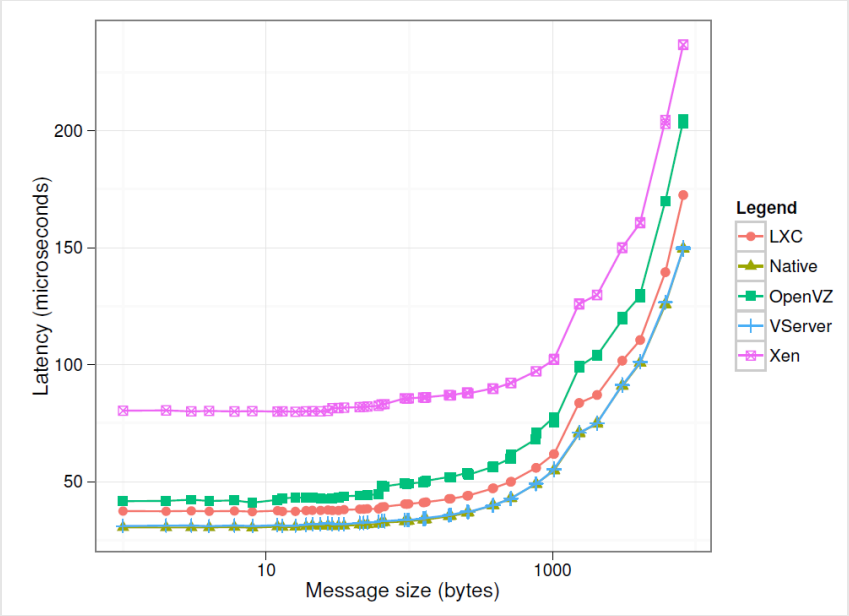


Networking Performance

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%