LinuX Containers

Bogdan Purcăreaţă

May 8, 2013

- Concepts
  - Operating Systems
  - Virtualization
- LinuX Containers
- Usage
  - Schematics
  - Demo
  - Benchmarks
- Relevance
  - Spread
  - Use Cases
- QA

freescale™

IT trends

- More resources
  - Better hardware at lower costs
  - Higher standards for software quality
- More users
  - Contact with technology at an earlier age
  - Shared access to the same device
- Data consolidation
  - Data warehousing
  - Service unification
  - Differentiated access
- Increased flexibility
  - Versatile configuration
  - Focus on usability

OS Recap

- Resources
  - CPU
  - Memory
  - Peripherals
- Structures
  - The scheduler
  - The pager
  - Filesystems
- The kernel
  - Handles hardware
  - Exposes capabilities
  - Manages resources

Figure: The Memory Pager

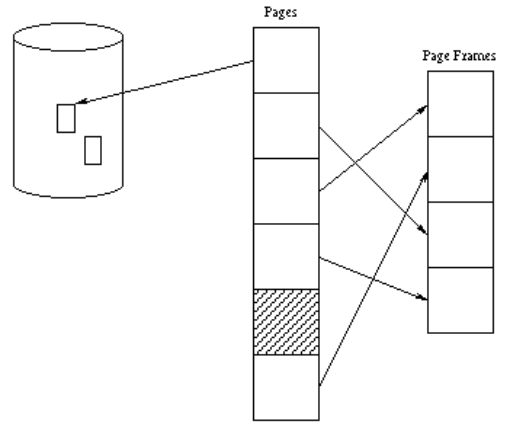
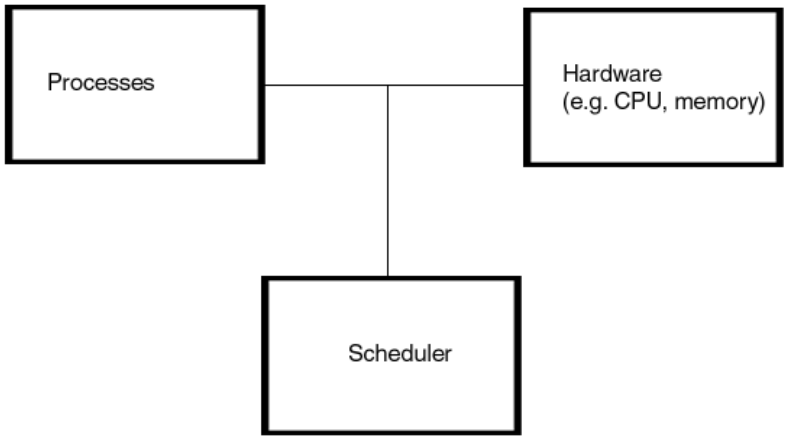


Figure: The Scheduler



Virtualization

- Key aspects:
  - Simulation (of HW / SW)
  - Virtual machines
  - Autonomous computing
  - Utility computing
- Advantages:
  - Better resource usage
  - Lower running costs
  - Improved security
- 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: **shared kernel**

freescale™

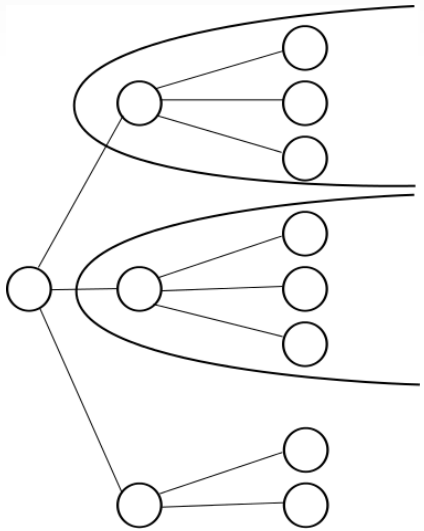
LinuX Containers

- a.k.a. LXC:
  - Mature technology implementation
  - Mainline kernel support
  - Application vs. System
  - Active development
- Components:
  - *Kernel features*
  - *Userspace tools*
  - *Configuration files*
  - *Template files*

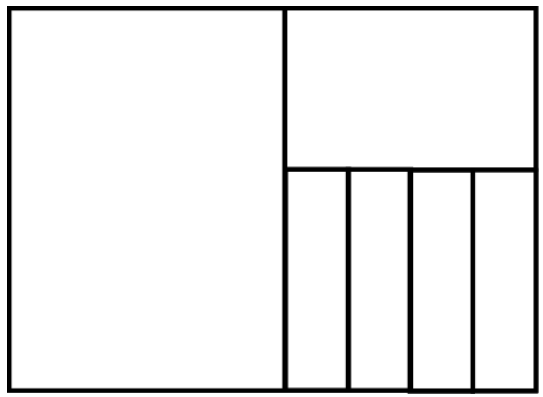
freescale™

Kernel Support

- Namespaces:
  - Abstract resources
  - Processes see the resource as their own
  - Isolation between namespaces



- Control Groups
  - Resource management among processes
  - Hierarchical support
  - Interaction with resource responsible structures:
    - Scheduler
    - 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)
```

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/foo2/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/foo2/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)
```

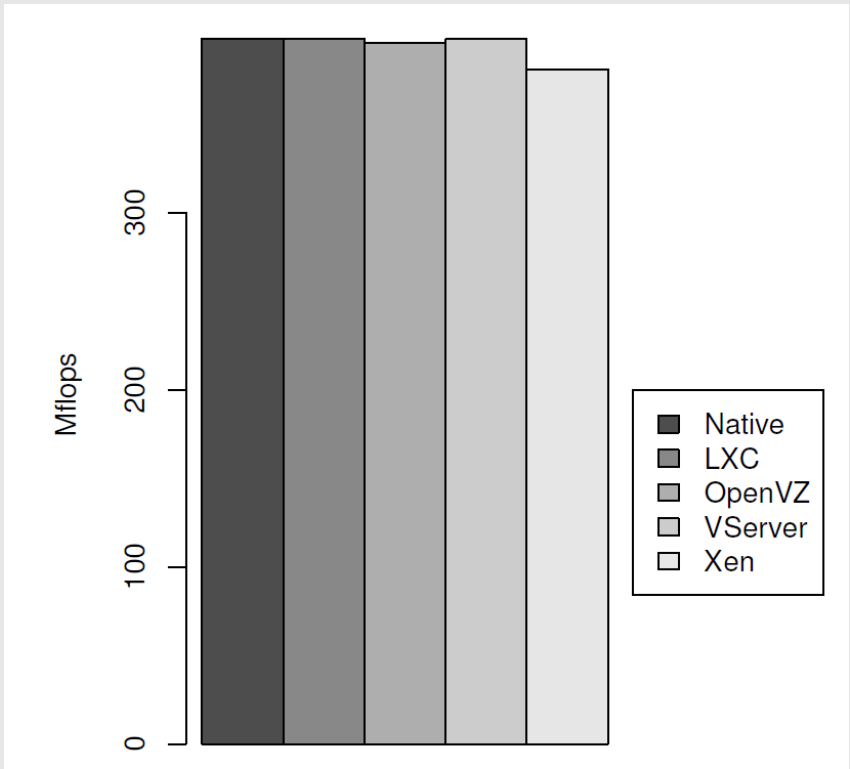
Demo

1. Start 2 containers
2. Check PIDs
3. Assign them a single core on the host
4. Balance CPU usage 25% - 75%

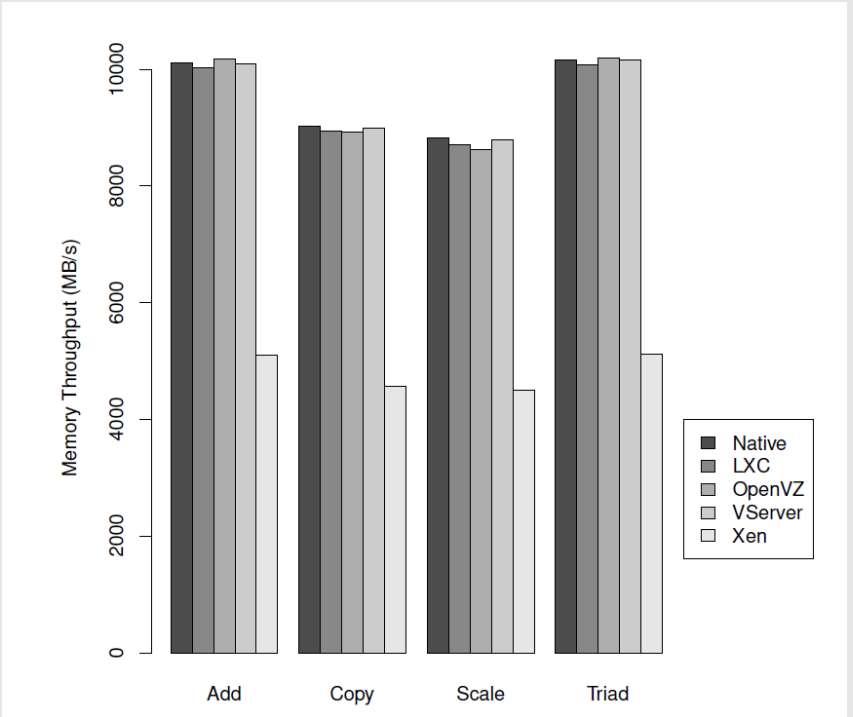
freescale™

System Performance

CPU performance
Linpack



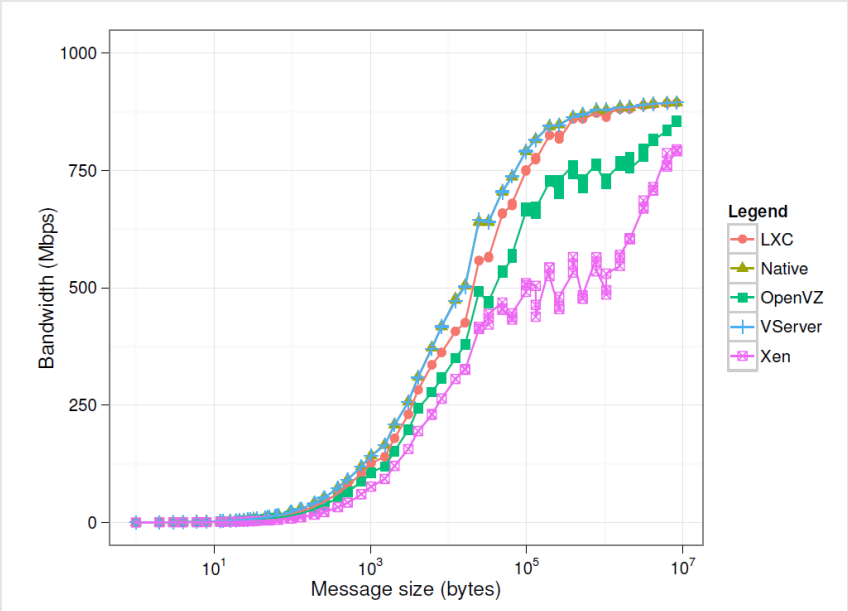
Mem throughput
Stream



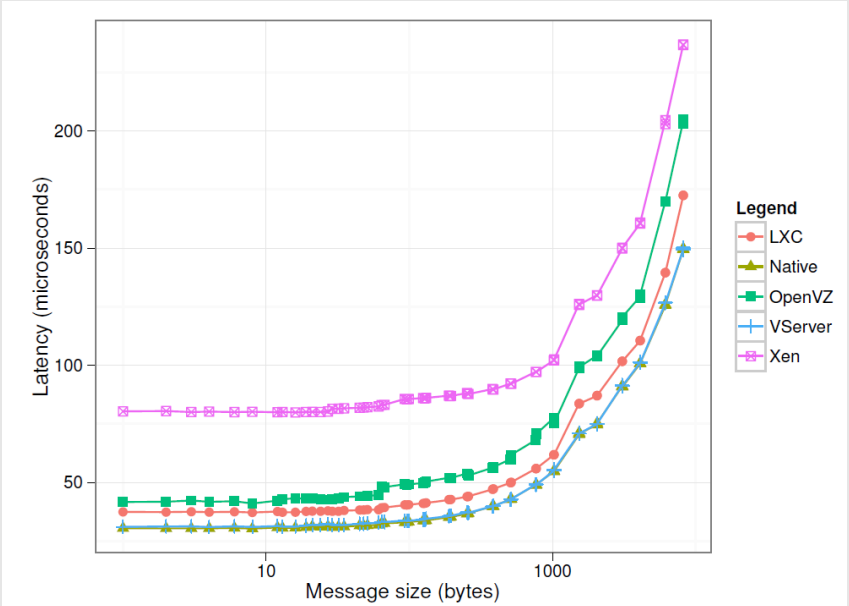
freescale™

Networking Performance

Bandwidth
NetPIPE



Latency
NetPIPE



freescale™

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 with 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 USDPAA in Containers

- DPAA - DataPath Acceleration Architecture
  - HW architecture providing advanced networking capabilities
  - Present in dedicated networking equipment
  - Traffic shaping, package accelerators, cryptography engine
- USDPAA - User Space DPAA
  - Userspace drivers based on the kernel UIO framework
  - Increased flexibility in application development
  - Reduced risk of bugging the kernel
  - Better error handling and system protection
  - Performance overhead
- Multiple USDPAA instances in containers
  - Improved isolation
  - Additional protection layer
  - Finer resource tuning

References

- `lxc.sourceforge.net`
- Yang Yu: OS-level Virtualization and Its Applications
- Miguel G. Xavier: Performance Evaluation of Container-based Virtualization for High Performance Computing Environments

freescale™

Thank you!

Questions?

freescale™