



LXCBENCH

**Understanding the capabilities of
Linux Containers in IVI
applications through
benchmarking**

Gianpaolo Macario

Mentor Embedded – Linux Services

GENIVI – EG-SI Architect

Automotive Linux Summit Fall
Edinburgh, UK – October 2013

Android is a trademark of Google Inc. Use of this trademark is subject to Google Permissions.
Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Agenda

- **The problem statement**
- **LXC: The technology**
- **How LXC may address some IVI use cases**
- **The GENIVI LXCBENCH Incubation Project**
- **Summary**

Who am I?



Gianpaolo Macario

Architect, Mentor Graphics – ESD – Linux Services

System Architect, GENIVI Alliance

Past Experiences: System Architect, Magneti Marelli – Infotainment & Telematics
IT Manager, COMAU – Robotics Business Unit
Software Consultant, GlobalValue (a Fiat/IBM Italy JV)
...

Linux and Open Source user and developer since 1993 (linux-0.99pl13)

Very proud of my first contribution to Linux Kernel early in 1995 (linux-1.3.13)

My social life: <http://it.linkedin.com/in/gmacario/>

<https://twitter.com/gpmacario>

Embedded Industry Trends

Hardware Consolidation

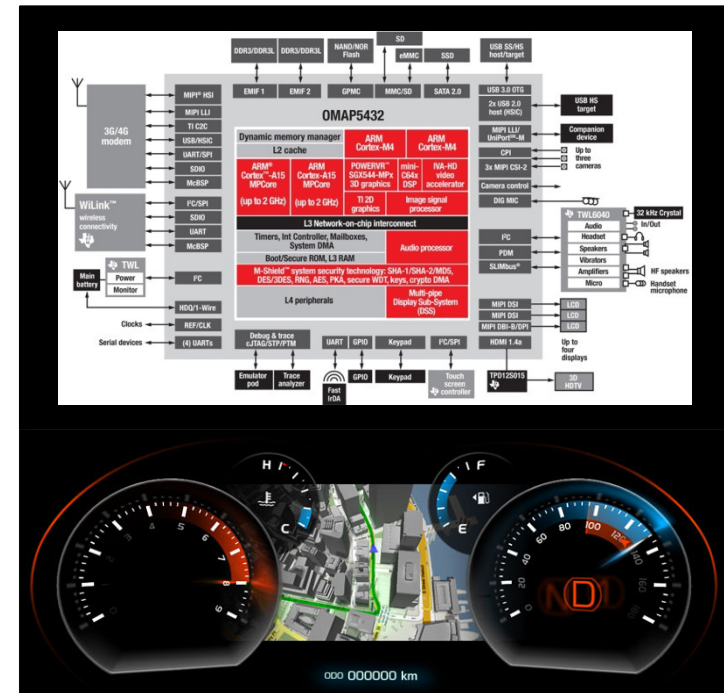
- SoC
- Subsystems
- Systems

Benefits

- Reducing BOM and power consumption
- Increasing performance and capacity of the system

Concerns

- Complexity of the system design, development, debugging



Example 1: Consolidation GENIVI/AUTOSAR

- **Most IVI systems have traditionally split the system architecture on separate CPUs/MCUs:**
 - **Infotainment domain: GENIVI Linux software stack, graphical resources, etc**
 - **Automotive domain: RTOS and AUTOSAR stack for access to vehicle network, ECU lifecycle, etc**
- **New SoC are coming to market which integrate CPU cores (i.e. Cortex A9/A15, M3/M4) and on-board peripherals optimized for the two domains**
- **How can the software architecture be deployed on such complex SoCs?**

Example 2: Linux Main Head Unit + Android RSE

- **In this scenario the functions of two units are being consolidated:**
 - Main Head Unit (navigation, radio broadcast, etc.) controlled by the driver and running → GENIVI Linux
 - Rear-Seat Entertainment (Internet browsing, downloadable apps, games) for rear passengers → Android
- **To realize such configuration the resources provided of one complex SoC (i.e. CPU cores, video ports, USB ports, etc) have to be allocated to the two domains to make sure the RSE functions are isolated and will not impact the performances of the main HU**

Approach to IVI domains consolidation

- **Establish a reduced set of interfaces between the domains**
- **Provide control over resource utilization**
- **Ease deployment and integration of function domains on the same ECU**

What are Linux Containers?

Linux Containers represent one of several techniques to realize system consolidation by providing a lightweight virtual system mechanism for Linux able to implement

- Resource management
- Process management
- File System Isolation

Each container provides a reduced view of the same kernel that has created the container itself. As a result, multiple containers may run on the same hardware and can share resources through the single Linux kernel that created them.

Linux Containers rely on a few features available in the Linux kernel since 2.6.x:

- cgroups
- namespaces

Additionally, the LXC user-space tools provide an abstraction to allow programs to: create containers, start/stop containers, etc.

What are NOT Linux Containers?

Linux Containers follow an opposite approach if compared to many other virtualization tools (most notably, hypervisors). In fact, rather than starting from an emulated hardware (completely isolated) and then trying to reduce overhead and improve performances, LXC use already efficient mechanisms and build up isolation.

Think about LXC as “chroot on steroids”

Containers can be used as an alternative to OS-level virtualization to run multiple isolated systems on a single host, and can provide different degrees of isolation.

For what security is concerned, Linux Containers leverage existing security mechanisms available in Linux:

- LXC rely on Discretionary Access Control (provided by the Linux kernel)
- Can leverage Mandatory Access Control (such as SE Linux and smack) if available

Hypervisors and Linux Containers: comparing features

| Feature | Type-1 Hypervisor | Linux Containers | Notes |
|---|---|--|---|
| Guest VM | Each guest runs inside a dedicated (virtual) hardware – and therefore is not limited to a Linux-based system | All guests share the same Linux kernel of the host – so only Linux-based OS are supported | Consolidation of mixed OS (i.e. AUTOSAR+Linux) cannot be realized with LXC |
| Guest Kernel | Each guest kernel is loaded into its own memory region | Only one Linux kernel image is loaded into physical memory | |
| Adaptations needed to guest OS kernel | Guest OS kernel needs to be made HV-aware | No – The Linux kernel device drivers validated on the Bare Metal Hardware are enough | Effort to make each guest OS HV-aware depends on how actual devices are allocated to guest OS |
| SoC dependencies | To minimize performance overhead over Bare Metal some Hypervisors take advantage of specific HV-oriented features available in most recent SoCs | No – kernel support for LXC is independent and neutral to hardware architecture/SoC | |
| Communication between guests | Through virtual (i.e. Ethernet or Serial Adapters) hardware devices realized by the Hypervisor | All standard Linux IPC mechanisms (sockets, pipes, signals message queues, etc) may be used | Communication between guests is achieved through fewer SW layers on LXC |
| Sharing libraries/filesystems between guests, or between guest and host | Not possible | Through LXC config options guests may transparently mount subdirectories of the host | This extra flexibility of LXC allows optimization of system storage/RAM usage but adds constraints when updating guests |
| Security mechanisms | Built into each HV, details vary between implementations | Although LXC adds no security to Linux, other technologies such as Mandatory Access Control may be leveraged | A thorough system security analysis should drive design choices here |
| Software License | Depends on HV vendor | Kernel Features: GPLv2 Userspace: LGPL-2.1 | |

Embedded Domain Separation Options

Safety Systems

Airbags, ABS, stability, etc

In-Vehicle Infotainment

Navigation, Multimedia

Powertrain

ECU, HEV/EV, Air-fuel analyzers

Telematics

Connected car, Web Services

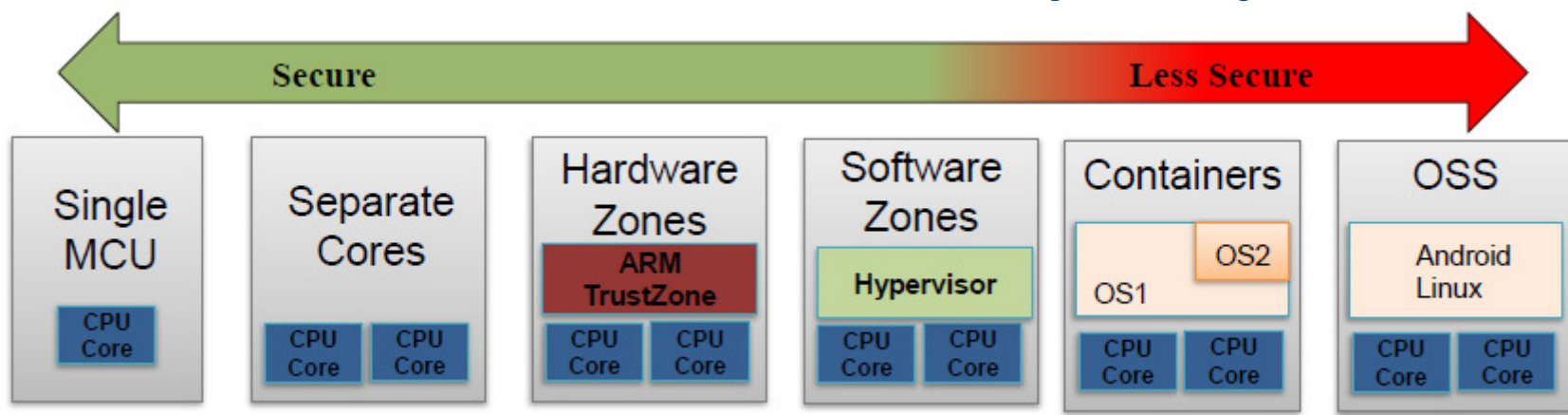
Body Electronics

Keyless, seat memory, etc.

Instrument Cluster

ADAS

Parking / Reversing



Sample LXC Configuration

Each container is customized by means of a LXC configuration file which specifies the resources that will be made available to the guest OS running inside the container. Example:

```
# Set cgroups CPU affinity to allow exclusive usage of one core by guest OS
```

```
lxc.cgroup.cpuset.cpus = 0,1
```

```
cpuset.cpu_exclusive = 1
```

```
# Configure the machine name as seen by guest OS
```

```
lxc.utsname = mycontainer1
```

```
# Define path of Root Filesystem and mount points for the guest OS
```

```
lxc.rootfs = /full/path/to/rootfs.mycontainer1
```

```
lxc.mount = /full/path/to/fstab.mycontainer1
```

```
# Deny all access to devices from the guest OS, except /dev/null and /dev/zero
```

```
lxc.cgroup.devices.deny = a
```

```
lxc.cgroup.devices.allow = c 1:3 rw
```

```
lxc.cgroup.devices.allow = c 1:5 rw
```

NOTE: many more options are available to control guest OS behaviour (see `man lxc.conf`)

LXCBENCH Project Goals



Among the different virtualization technologies, Linux Containers (LXC) are particularly appealing for implementing multi-hosting embedded systems by providing a userspace container object that supports full resource isolation and resource control for an application or a system.

The purpose of the LXCBENCH incubation project is to gain a better understanding of the capabilities of Linux Containers through benchmarking.

[Read more...](#)

LXCBENCH was the first project proposed by a non-GENIVI Member Organization (Politecnico di Torino) which has been accepted and launched as an Open Source Project by the GENIVI Alliance:

<http://projects.genivi.org/lxcbench/>

LXCBENCH Benchmarking Approach

The benchmarking activity aims to measure the performance of an LXC-equipped system versus a reference system not equipped with virtualization.

For this purpose we selected the Phoronix Test Suite (<http://www.phoronix-test-suite.com>), an Open Source automated testing framework.

The following measurement to be executed on a given hardware platform:

Out-of-the-box Linux system (e.g., Ubuntu or similar).

Run PTS to collect a set of measurements to be used as the baseline.

Out-of-the-box Linux system enriched with 1 container, running Linux (e.g., Ubuntu itself, or Android).

Run PTS in the baseline system only, run PTS in the container, run PTS in both concurrently.

After collecting the performance measurements in the above scenario, overheads are evaluated by comparing the performance score of the baseline system with those of the container-enriched system.

The LXC BENCH project is not...

The goal of the LXC BENCH project is NOT to measure raw performance of a given hardware platform, rather to understand the overhead coming from the adoption of virtualization on different hardware configurations.

For this purpose, we intend to repeat the same measurements on different hardware platforms, equipped with both single-core and multi-core (e.g., iMX5, iMX6), and to quantify the overheads before and after virtualization.

LXCBENCH Project Timeline

Completed Milestones:

- 2012-11-12: LXCBENCH Project proposal submitted to GENIVI
- 2013-02-13: LXCBENCH launched as GENIVI Incubation Project
- 2013-02-25: First version of benchmarking report
(from OSES 2013 students) released
- 2013-03-11: Yocto layer meta-lxcbench created
- 2013-05-17: Phoronix Test Suite recipes integrated into meta-lxcbench
- 2013-06-30: Added support for more embedded reference boards
- 2013-09-30: LXCBENCH featured in “Industry and Research Perspectives on Embedded System Design”
- 2013-10-25: LXCBENCH presented at ALS Fall 2013

LXCBENCH Preliminary Results (1/3)

Test Bench

Motherboard: OMAP4 Pandaboard

Processor: ARMv7 rev 2 @ 1.01GHz (2 Cores)

Memory: 1024MB

Disk: 8GB SU08G

OS: Debian 6.0.7

Kernel: 3.8.0 (armv7l)

Compiler: GCC 4.4.5

Root Filesystem: ext4

Screen Resolution: 1280x720

LXCBENCH Preliminary Results (2/3)

| Performance | PTS Test | Host (while Guest is idle) | Guest (while Host is idle) | Host concurrently with Guest | Guest concurrently with Host |
|-------------------|---------------------------|----------------------------|----------------------------|------------------------------|------------------------------|
| Stream [MB/s] | Copy | 1.00 | 1.00 | 0.99 | 0.95 |
| | Scale | 1.00 | 1.00 | 0.99 | 0.84 |
| | Triad | 1.00 | 1.00 | 0.99 | 0.77 |
| | Add | 1.00 | 1.00 | 0.99 | 0.83 |
| Cachebench [MB/s] | Read | 1.00 | 1.00 | 0.99 | 0.99 |
| | Write | 1.00 | 1.00 | 1.00 | 1.00 |
| | Read/ Write/ Modify | 1.00 | 1.00 | 1.00 | 1.00 |
| LAME MP3 [s] | Encoding | 1.00 | 1.00 | 1.00 | 1.00 |

Note: relative performance numbers normalized to column “Host (while Guest is idle)”

LXCBENCH Preliminary Results (3/3)

Containers are very effective in showing low overhead (host and guest are indeed running on real hardware).

When the same benchmark is running on both the host and the guest concurrently, a negligible performance hit is observed (it is expected, as both are looking for the same memory resource).

Some strange figures ask for further investigations (e.g., the 30% performance hit on Stream/Triad running on the guest).

Basic considerations

Benchmarking is a complex subject:

- Requires an accurate description of the intended use case
- Requires dependable configuration of the test harness

Since there are so many factors which will affect the overall performance, we want to instrument and benchmark the basic mechanisms first.

Rather than the focusing on the actual benchmark results – which will depend on the chosen configuration, we chose LXC BENCH to deliver tools and mechanisms that may be reused in different specific product configurations.

LXCBENCH Reusable artifacts

In order to:

- 1) Realize a dependable software configuration
- 2) Allow configuration according to different use cases
- 3) Support complete build from sources
- 4) Reduce complexity and interference from other factors
(not related to the actual PTS benchmark being run)

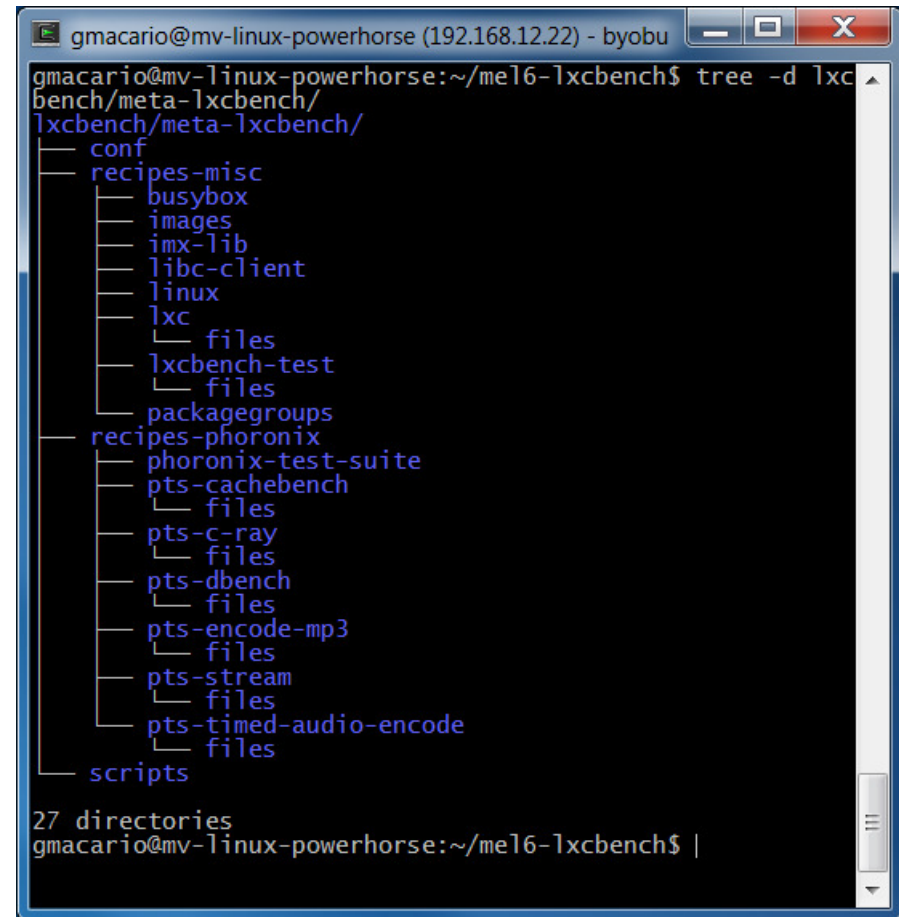
As part of the LXCBENCH project we have been developing a simplified, stripped down embedded Linux distribution

- based on the Yocto™ project
- supporting a few embedded HW targets that have been made available to Politecnico di Torino.

The meta-lxcbench Yocto Layer

The **meta-lxcbench** subdirectory of the LXC BENCH git repository contains a Yocto 1.3 layer with the recipes and changes needed to build a reproducible embedded Linux distribution able to run the LXC BENCH tests on several target reference boards.

The meta-lxcbench layer is developed and tested against Mentor Embedded Linux (<http://www.mentor.com/embedded-software/linux/>) however it should also work with other Yocto 1.3 based distributions. Contributions are welcome!



```
gmacario@mv-linux-powerhorse (192.168.12.22) - byobu
gmacario@mv-linux-powerhorse:~/me16-lxcbench$ tree -d lxc
bench/meta-lxcbench/
lxcbench/meta-lxcbench/
├── conf
├── recipes-misc
│   ├── busybox
│   ├── images
│   ├── imx-lib
│   ├── libc-client
│   ├── linux
│   ├── lxc
│   │   ├── files
│   ├── lxcbench-test
│   │   ├── files
│   ├── packagegroups
│   ├── recipes-phoronix
│   ├── phoronix-test-suite
│   ├── pts-cachebench
│   │   ├── files
│   ├── pts-c-ray
│   │   ├── files
│   ├── pts-dbench
│   │   ├── files
│   ├── pts-encode-mp3
│   │   ├── files
│   ├── pts-stream
│   │   ├── files
│   ├── pts-timed-audio-encode
│   │   ├── files
├── scripts
└── 27 directories
gmacario@mv-linux-powerhorse:~/me16-lxcbench$ |
```

meta-lxcbench: HW Targets

Currently supported (MACHINE in local.conf):

| | |
|-------------------|------------------------------------|
| imx53qsb | Freescale i.MX53 Quick Start Board |
| pandaboard | TI OMAP (PandaBoard) |
| qemuarm | QEMU for ARM (Virtual Hardware) |
| qemux86 | QEMU for x86 (Virtual Hardware) |

Support for other physical HW targets is under internal development but not yet ready for public release:

- Freescale i.MX6 Sabre Auto
- Renesas R-Car H1 (Marzen)
- Wandboard

LXCBENCH: Build distro from sources

```
# Prerequisite: Sourcery CodeBench and Mentor Embedded Linux 6.0 installed
# (other Yocto-1.3 compatible environments may work as well)

cd ~/mel6-lxcbench

# Check out the meta-lxcbench layer
git clone git://git.projects.genivi.org/lxcbench.git

# Choose the target hardware (example: Freescale i.MX53 QSB)
source meta-mentor/setup-environment -bbuild-imx53qsb imx53qsb

# Customize conf/bblayers.conf
cat <<END >>conf/bblayers.conf
BBLAYERS += "/scratch/gmacario/mel6-lxcbench/lxcbench/meta-lxcbench"
END

# Customize conf/local.conf
cat <<END >>conf/local.conf
EXTERNAL_TOOLCHAIN ?= "<Sourcery CodeBench ARM GNU/Linux installation directory>"
END

# Build it!
bitbake core-image-lxcbench
```


LXCBENCH distro: Build Results

```
gmacario@mv-linux-powerhorse:/scratch/gmacario/mel6-lxcbench/build-imx53qsb$ ls -la tmp/deploy/images/
total 91872
drwxrwxr-x 2 gmacario gmacario      4096 Mar 10 15:24 .
drwxrwxr-x 6 gmacario gmacario      4096 Mar 10 15:07 ..
-rw-r--r-- 1 gmacario gmacario      3620 Mar 10 15:09 core-image-lxcbench-imx53qsb-20130310135537.license_manifest
-rw-r--r-- 1 gmacario gmacario      1590 Mar 10 15:09 core-image-lxcbench-imx53qsb-20130310135537.license_manifest.csv
-rw-r--r-- 1 gmacario gmacario    61735936 Mar 10 15:09 core-image-lxcbench-imx53qsb-20130310135537.rootfs.ext3
-rw-r--r-- 1 gmacario gmacario 3400000000 Mar 10 15:09 core-image-lxcbench-imx53qsb-20130310135537.rootfs.sdcard
-rw-r--r-- 1 gmacario gmacario     4921147 Mar 10 15:09 core-image-lxcbench-imx53qsb-20130310135537.rootfs.tar.bz2
lrwxrwxrwx 1 gmacario gmacario        55 Mar 10 15:09 core-image-lxcbench-imx53qsb.ext3 -> core-image-lxcbench-imx53qsb-20130310135537.rootfs.ext3
lrwxrwxrwx 1 gmacario gmacario        60 Mar 10 15:09 core-image-lxcbench-imx53qsb.license_manifest -> core-image-lxcbench-imx53qsb-20130310135537.license_manifest
lrwxrwxrwx 1 gmacario gmacario        64 Mar 10 15:09 core-image-lxcbench-imx53qsb.license_manifest.csv -> core-image-lxcbench-imx53qsb-20130310135537.license_manifest.csv
lrwxrwxrwx 1 gmacario gmacario        57 Mar 10 15:09 core-image-lxcbench-imx53qsb.sdcard -> core-image-lxcbench-imx53qsb-20130310135537.rootfs.sdcard
lrwxrwxrwx 1 gmacario gmacario        58 Mar 10 15:09 core-image-lxcbench-imx53qsb.tar.bz2 -> core-image-lxcbench-imx53qsb-20130310135537.rootfs.tar.bz2
-rw-rw-r-- 1 gmacario gmacario    357723 Mar  8 23:33 modules-2.6.35.3-11.09.01+yocto+g012a4b8-r33.16-imx53qsb.tgz
-rw-rw-r-- 1 gmacario gmacario       294 Mar 10 15:08 README_-_DO_NOT_DELETE_FILES_IN_THIS_DIRECTORY.txt
lrwxrwxrwx 1 gmacario gmacario        31 Mar  8 23:24 u-boot.imx -> u-boot-imx53qsb-v2012.07-r1.imx
lrwxrwxrwx 1 gmacario gmacario        31 Mar  8 23:24 u-boot-imx53qsb.imx -> u-boot-imx53qsb-v2012.07-r1.imx
-rwxr-xr-x 1 gmacario gmacario    285788 Mar  8 23:24 u-boot-imx53qsb-v2012.07-r1.imx
lrwxrwxrwx 1 gmacario gmacario        50 Mar  8 23:33 uImage -> uImage-2.6.35.3-r33.16-imx53qsb-20130308222009.bin
-rw-r--r-- 1 gmacario gmacario    3295976 Mar  8 23:33 uImage-2.6.35.3-r33.16-imx53qsb-20130308222009.bin
lrwxrwxrwx 1 gmacario gmacario        50 Mar  8 23:33 uImage-imx53qsb.bin -> uImage-2.6.35.3-r33.16-imx53qsb-20130308222009.bin
gmacario@mv-linux-powerhorse:/scratch/gmacario/mel6-lxcbench/build-imx53qsb$
```

LXCBENCH distro: Deploy on Target HW

Example (for MACHINE="imx53qsb")

Plug a uSDHC into the host, type "dmesg" to identify the block device

that has been created (in our example: /dev/sdX)

dmesg

SDCARD=/dev/sdX

sudo umount \${SDCARD}?

cat tmp/deploy/images/core-image-lxcbench-imx53qsb.sdcard | sudo dd of=\${SDCARD}

That's it! Then boot the target, login as root and type on the serial console

lxcbench-test01.sh

This script will configure and run the PTS suite.

After a few hours results will be available under ~/.phoronix-test-suite/test-results

LXCBENCH Project Roadmap: short term

To improve the accuracy of the measurements we collect during benchmarking, using specific tools

- e.g.: Mentor Embedded System Analyzer along with Phoronix Test Suite benchmarks

To enrich the set of benchmarks we are using

- e.g.: we are looking at specific tests for measuring GPU performance, and test specifically devised for Android

To extend the analysis by measuring performance overhead on different hardware architectures:

- i.MX53 done, i.MX6 in progress, Renesas R-Car in progress
- Zynq (Xilinx Dual-Core A9) planned hw already available at PoliTO
- Intel Atom (desired but no hw available so far)

and different OS combinations:

- e.g.: Linux+Linux, Linux+Android, commercial Linux+Linux, commercial Linux+Android

and different kernel configuration

- e.g., using different kernel scheduling policies

By comparing performance overhead vs. Bare Metal with those of other technologies (mainly hypervisors)

LXCBENCH Project Roadmap: medium

To measure robustness (i.e., security) of containers

- we are considering using known root kits and other attacks to "jailbreak" from one container to another

To compare container robustness with other technologies
(again here we are looking at hypervisors)

Do you have further suggestions, ideas or want to contribute?

Let's get in touch!



mentor
embedded

Mentor
Graphics

Questions?

Android is a trademark of Google Inc. Use of this trademark is subject to Google Permissions.
Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

For further information

About the GENIVI LXC BENCH Project

LXC BENCH Project Homepage: <http://projects.genivi.org/lxcbench>

LXC BENCH Git Repository: <http://git.projects.genivi.org/?p=lxcbench.git>

Subscribe to Mailing List: genivi-lxcbench@lists.genivi.org

GENIVI project stats at Ohloh: <https://www.ohloh.net/orgs/genivi>

Related Mentor Embedded Webinars

Implementing Android Based Automotive Infotainment systems:

<http://go.mentor.com/2x922>

Using Linux in Automotive Systems with Linux Containers:

<http://www.mentor.com/embedded-software/events/using-android-in-automotive-systems-with-linux-containers>



mentor
embedded

Mentor
Graphics

End

Android is a trademark of Google Inc. Use of this trademark is subject to Google Permissions.
Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.