CARE: the Comprehensive Archiver for Reproducible Execution

Yves Janin, Cédric Vincent, Rémi Duraffort

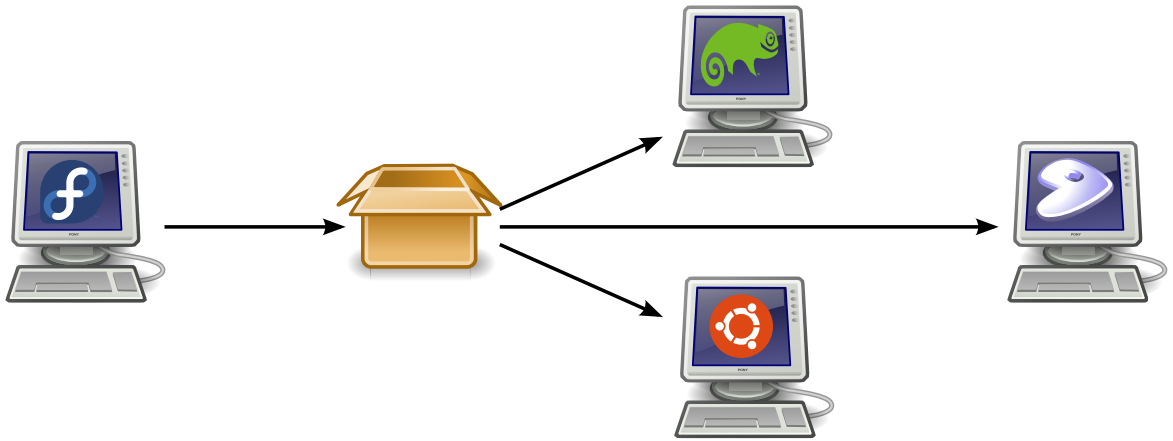
STMicroelectronics, Grenoble, France

TRUST'14
June 12th, 2014

Outline

CARE in short

- capturing files and environment for an experiment,
- building an archive,
- re-executing on another independent machine,
- no setup, no administrative privilege (*running in userland*),
- based on system call interposition.



A lightweight model for reproducibility

Main assumptions

- Linux kernels ($\geq$ 2.6.0), software stacks don't matter,
- backward compatible ISA, but emulation may help,
- targeting computational reproducibility.

A lightweight model for reproducibility

Main assumptions

- Linux kernels ($\geq$ 2.6.0), software stacks don't matter,
- backward compatible ISA, but emulation may help,
- targeting computational reproducibility.

CARE ambitions

- easy to use,
- suitable for scenarios with embedded Linux targets,
- flexible enough for various re-execution modes.

A simple use case - X86_64

```
## Archive creation on workstation 1
x86_64$ care -o foo.bin make

## Archive re-execution on workstation 2
x86_64$ ./foo.bin
x86_64$ ./foo/re-execute.sh
```

A simple use case - X86_64

```
## Archive creation on workstation 1
x86_64$ care -o foo.bin make

## Archive re-execution on workstation 2
x86_64$ ./foo.bin
x86_64$ ./foo/re-execute.sh
```

Behind the scenes

- syscall interposition for archive creation & re-execution,
- re-execution in a confined environment,
- archive content = partial **copy** of the original filesystem.

▸ Annex A

CARE architecture (1/2)

$$CARE = PRoot + Archiver$$

CARE architecture (1/2)

$CARE = PRoot + Archiver$

PRoot: used for archive creation and re-execution

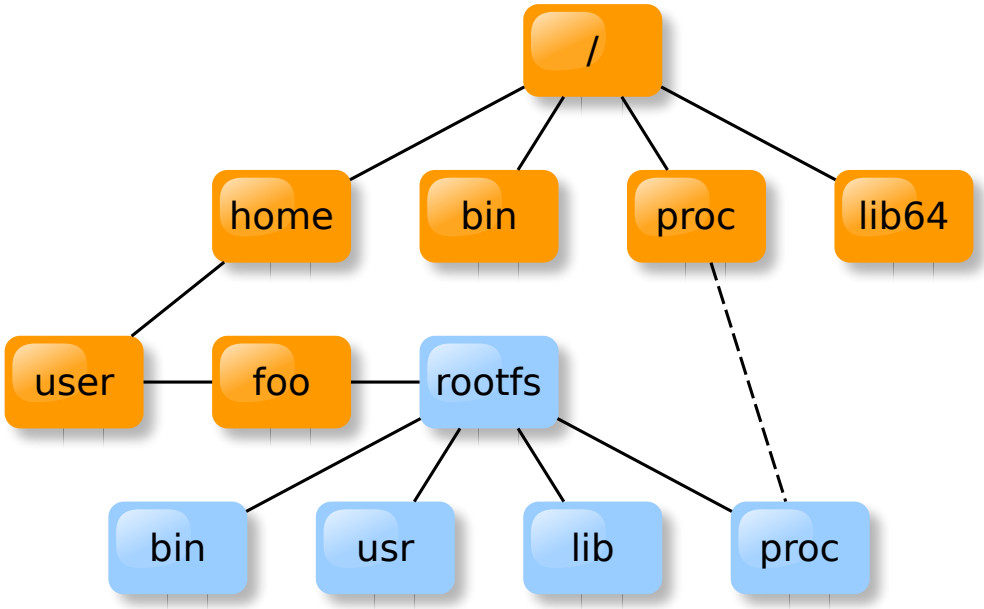
- a generic system call interposition engine (*ptrace syscall*),
- a path canonicalization engine à la *chroot* or *mount --bind*

CARE architecture (1/2)

$CARE = PRoot + Archiver$

PRoot: used for archive creation and re-execution

- a generic system call interposition engine (*ptrace syscall*),
- a path canonicalization engine à la *chroot* or *mount --bind*



CARE architecture (2/2)

$$CARE = PRoot + Archiver$$

PRoot advanced feature: syscall emulation

- Syscalls and syscall parameters can be modified,
- PRoot emulates syscalls to enhance kernel compatibility.

CARE architecture (2/2)

$$CARE = PRoot + Archiver$$

PRoot advanced feature: syscall emulation

- Syscalls and syscall parameters can be modified,
- PRoot emulates syscalls to enhance kernel compatibility.

Archiver

- decides which files should be inserted in archives,
- implements a new history-based algorithm,
- saves environment variables,
- adds the re-execution machinery to archives.

CARE and virtualization solutions

Archive	VMs	CARE
size (*typ.*)	in GBs	in tens or hundreds of MBs
content	arch. desc. + full OS	only files used by artifacts
format	VM-centric (*COW*,...)	generic (*cpio, tar, gzip,...*)

Table 1: Comparing archive attributes for VMs and CARE

CARE and virtualization solutions

Archive	VMs	CARE
size (*typ.*)	in GBs	in tens or hundreds of MBs
content	arch. desc. + full OS	only files used by artifacts
format	VM-centric (*COW*,...)	generic (*cpio, tar, gzip*,...)

Table 1: Comparing archive attributes for VMs and CARE

CARE and virtualization tools can be composed

- create CARE archives in a VM,
- or re-execute CARE archives in a VM,
- or re-execute CARE archives with *chroot* or *lxc*,
- or use emulation for archives built on embedded targets.

Replicating experiments on x86_64

Application	Initial	→	re-exec. hosts	Size
VLC-2.0.8	3.2.0	→	3.10.17	70 MB
MPlayer-1.1	3.10.17	→	2.6.18	43 MB
Wine-1.4	3.2.0	→	2.6.9	182 MB
Firefox-24.3.0	3.10.17	→	3.2.0	94 MB
Docutils-0.11	3.10.17	→	2.6.18	12 MB
MoarVM-2014.02	3.10.17	→	2.6.9	24 MB
Perl-5.18.2	3.10.17	→	2.6.9	42 MB

Table 2: Testing replicability. All archives compressed with *lzo*.

Enhanced kernel compatibility

Except VLC, all archives re-executed on older kernels.

Focusing on Perl 5.18.2 benchmark

Perl 5.8.12	syscalls			files	
	total	handled	emulated	seen	archi.
Build	7.3×10^6	3.0×10^6 *(40%)*	6.8×10^5 *(9%)*	47906	8082 *(17%)*

Table 3: CARE Perl 5.18.2 dynamic behavior: syscalls & files

- handled syscalls: nb of syscalls that are of interest to CARE,
- seen files: nb of unique files processed by CARE,
- archi. file: nb of files archived by CARE.

Noticeable achievements

- *Scaling up:* CARE was used to archive and re-execute the cross build of a complete embedded Linux distribution.

Noticeable achievements

- *Scaling up:* CARE was used to archive and re-execute the cross build of a complete embedded Linux distribution.
- *Preservability:* we validated that CARE archives created on a ten-years+ old system (Linux 2.6.7) can run on today's systems.

Noticeable achievements

- *Scaling up:* CARE was used to archive and re-execute the cross build of a complete embedded Linux distribution.
- *Preservability:* we validated that CARE archives created on a ten-years+ old system (Linux 2.6.7) can run on today's systems.
- *Observability on embedded targets:* a modified version of CARE enabled us to record all files accessed on an embedded ARM Cortex-A9 dual-core (@ 1GHz) Linux board, from start-up to shutdown.
  $\Longrightarrow$ *help prune embedded Linux distribution footprint*.

Conclusion

CARE can be tested freely

`http://reproducible.io`

- CARE is GPL v2+ licensed,
- Have a try, we would be interested in getting feedback!

See the demonstration video

`http://youtu.be/MvXhEgSEIs8`

- creation of an archive on an ARM Linux board,
- re-execution on another ARM Linux board (with a $\neq$ kernel),
- re-execution with emulation support on x86_64.

Demo application: the *links* web browser running in text mode.

Thank you for your attention!

CARE: `http://reproducible.io`

Demo: `http://youtu.be/MvXhEgSEIs8`

Annex A: CARE in practice - from ARM to X86_64

```
## Archive creation on ARM board
arm    $ care -o foo.tar.gz links

## Archive re-execution on a workstation
x86_64$ tar -zxf foo.tar.gz
x86_64$ PROOT="proot" ./foo/re-execute.sh \
                      -q qemu-arm links
```

Behind the scenes

- ISA & syscall emulation (here with QEMU user-mode).
- PRoot, a portable syscall interposition engine.