CARE: the Comprehensive Archiver for Reproducible Execution

Yves Janin, Cédric Vincent, Rémi Duraffort

STMicroelectronics, Grenoble, France

TRUST’14
June 12th, 2014
1 Introducing CARE
   - The CARE model
   - CARE architecture

2 Experimental Results
   - Typical use cases
   - Noticeable achievements

3 Conclusion

4 Annex
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.
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

```bash
## 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

```bash
## 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.
CARE architecture (1/2)

\[
\text{CARE} = \text{PRoot} + \text{Archiver}
\]
CARE architecture (1/2)

\[ \text{CARE} = \text{PRoot} + \text{Archiver} \]

**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 -b bind*
CARE architecture (1/2)

\[ \text{CARE} = \text{PRoot} + \text{Archiver} \]

**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)

\[ \text{CARE} = \text{PRoot} + \text{Archiver} \]

**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

<table>
<thead>
<tr>
<th>Archive</th>
<th>VMs</th>
<th>CARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>size (typ.)</td>
<td>in GBs</td>
<td>in tens or hundreds of MBs</td>
</tr>
<tr>
<td>content</td>
<td>arch. desc. + full OS</td>
<td>only files used by artifacts</td>
</tr>
<tr>
<td>format</td>
<td>VM-centric (COW,...)</td>
<td>generic (cpio, tar, gzip,...)</td>
</tr>
</tbody>
</table>

Table 1: Comparing archive attributes for VMs and CARE
CARE and virtualization solutions

<table>
<thead>
<tr>
<th>Archive</th>
<th>VMs</th>
<th>CARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>size (typ.)</td>
<td>in GBs</td>
<td>in tens or hundreds of MBs</td>
</tr>
<tr>
<td>content</td>
<td>arch. desc. + full OS</td>
<td>only files used by artifacts</td>
</tr>
<tr>
<td>format</td>
<td>VM-centric (COW,...)</td>
<td>generic (cpio, tar, gzip,...)</td>
</tr>
</tbody>
</table>

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.
## Experimental Results

### Replicating experiments on x86_64

<table>
<thead>
<tr>
<th>Application</th>
<th>Initial</th>
<th>→</th>
<th>re-exec. hosts</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLC-2.0.8</td>
<td>3.2.0</td>
<td>→</td>
<td>3.10.17</td>
<td>70 MB</td>
</tr>
<tr>
<td>MPlayer-1.1</td>
<td>3.10.17</td>
<td>→</td>
<td>2.6.18</td>
<td>43 MB</td>
</tr>
<tr>
<td>Wine-1.4</td>
<td>3.2.0</td>
<td>→</td>
<td>2.6.9</td>
<td>182 MB</td>
</tr>
<tr>
<td>Firefox-24.3.0</td>
<td>3.10.17</td>
<td>→</td>
<td>3.2.0</td>
<td>94 MB</td>
</tr>
<tr>
<td>Docutils-0.11</td>
<td>3.10.17</td>
<td>→</td>
<td>2.6.18</td>
<td>12 MB</td>
</tr>
<tr>
<td>MoarVM-2014.02</td>
<td>3.10.17</td>
<td>→</td>
<td>2.6.9</td>
<td>24 MB</td>
</tr>
<tr>
<td>Perl-5.18.2</td>
<td>3.10.17</td>
<td>→</td>
<td>2.6.9</td>
<td>42 MB</td>
</tr>
</tbody>
</table>

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

### Enhanced kernel compatibility

Except VLC, all archives re-executed on older kernels.
### Table 3: CARE Perl 5.18.2 dynamic behavior: syscalls & files

<table>
<thead>
<tr>
<th>Perl 5.8.12</th>
<th>syscalls</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>handled</td>
</tr>
<tr>
<td></td>
<td>$7.3 \times 10^6$</td>
<td>$3.0 \times 10^6$ (40%)</td>
</tr>
</tbody>
</table>

- **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.

⇒ 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.