PRoot: a Step Forward for QEMU User-Mode

Cédric Vincent Yves Janin

Compilation Expertise Center (CEC)
STMicroelectronics, Grenoble, France

1st International QEMU Users Forum (QUF'11)
March 18th 2011

Description

A QEMU Solution for Software Development

QEMU is the right solution for emulating embedded platforms.

- Runs fast,
- Supports several targets: ARM, ST40 (a.k.a. SH4), ...

Description

A QEMU Solution for Software Development

QEMU is the right solution for emulating embedded platforms.

- Runs fast,
- Supports several targets: ARM, ST40 (a.k.a. SH4), ...

QEMU user-mode

executes one *target* Linux application on a *host* Linux system.

Description

A QEMU Solution for Software Development

QEMU is the right solution for emulating embedded platforms.

- Runs fast,
- Supports several targets: ARM, ST40 (a.k.a. SH4), ...

QEMU user-mode

executes one *target* Linux application on a *host* Linux system.

PRoot

controls the execution of Linux processes.

Description

A QEMU Solution for Software Development

QEMU is the right solution for emulating embedded platforms.

- Runs fast,
- Supports several targets: ARM, ST40 (a.k.a. SH4), ...

QEMU user-mode

executes one *target* Linux application on a *host* Linux system.

PRoot

controls the execution of Linux processes.

Both combined

a lightweight emulation environment for Linux applications.

QEMUlating complex *embedded* Linux applications

A wide range of use cases

At STMicroelectronics, we use PRoot+QEMU user-mode to:

- build Linux packages for an embedded target;
- run test-suites and validations;
- and develop Rich Internet Applications (WebKit, FlashPlayer's VM).

QEMU system-mode *vs* user-mode

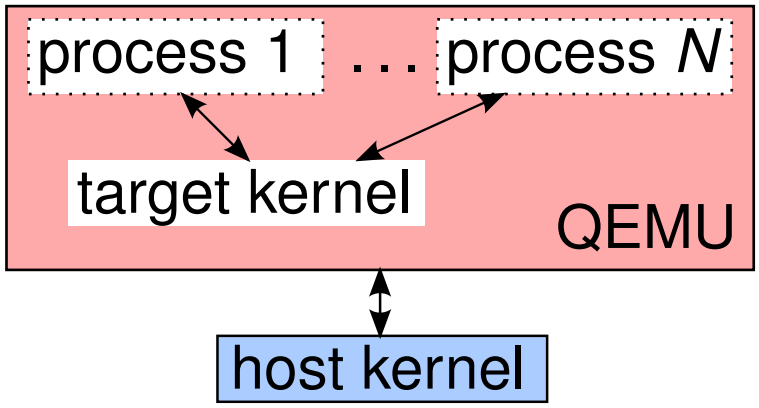

Figure: QEMU system-mode

QEMU system-mode *vs* user-mode

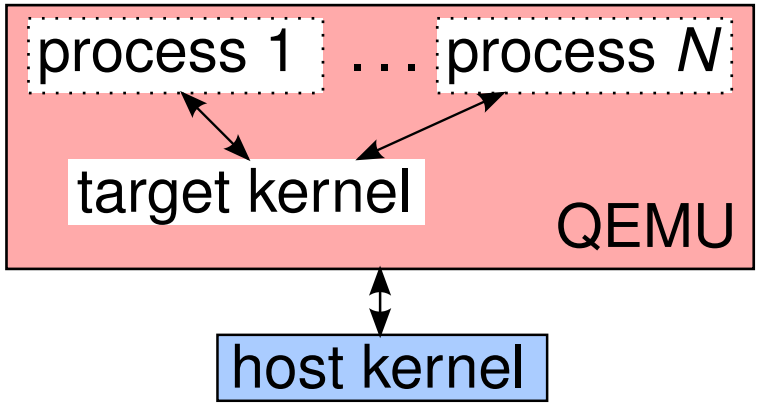

Figure: QEMU system-mode

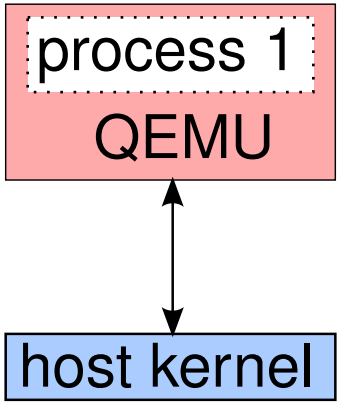

Figure: QEMU user-mode

Overcoming QEMU user-mode limitations with PRoot

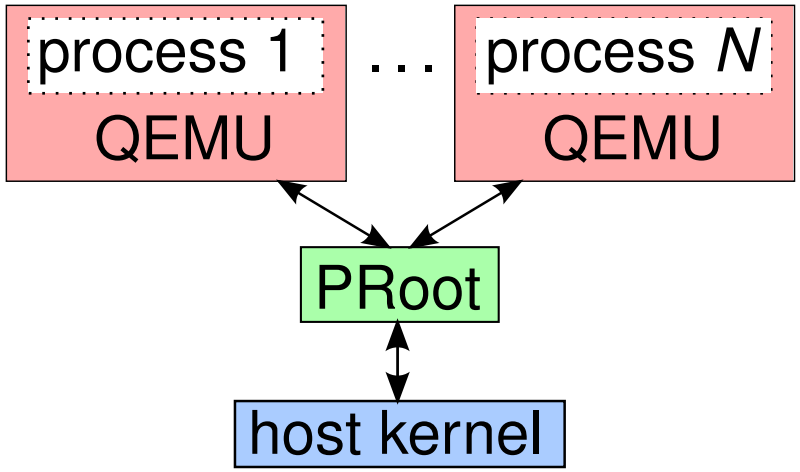

Figure: PRoot + QEMU user-mode

Overcoming QEMU user-mode limitations with PRoot

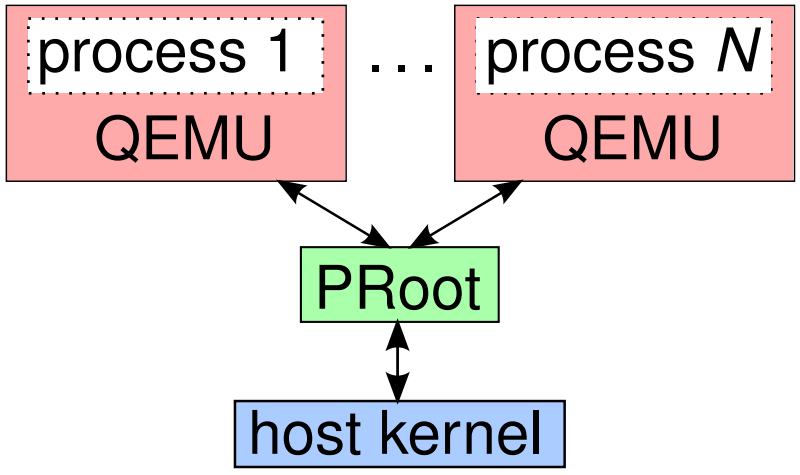

Figure: PRoot + QEMU user-mode

Defining two requirements for PRoot

R2: New target processes are kept under QEMU.

Overcoming QEMU user-mode limitations with PRoot

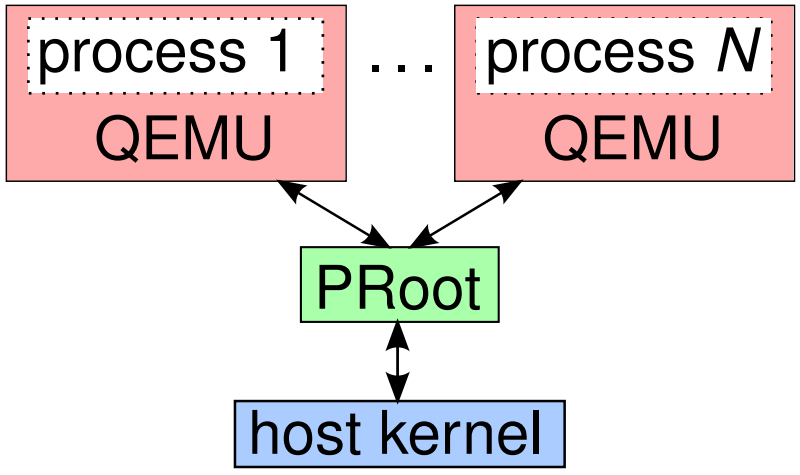

Figure: PRoot + QEMU user-mode

Defining two requirements for PRoot

R2: New target processes are kept under QEMU.

R1: Target processes are confined within the target *rootfs*.

The `ptrace` system call

`ptrace`

`ptrace` makes it possible for a process to control other processes. It has several applications:

debuggers: GDB, Strace, Ltrace, ...

kernel features: User-Mode Linux, Goanna FS, **PRoot**, ...

QEMU user-mode control flow

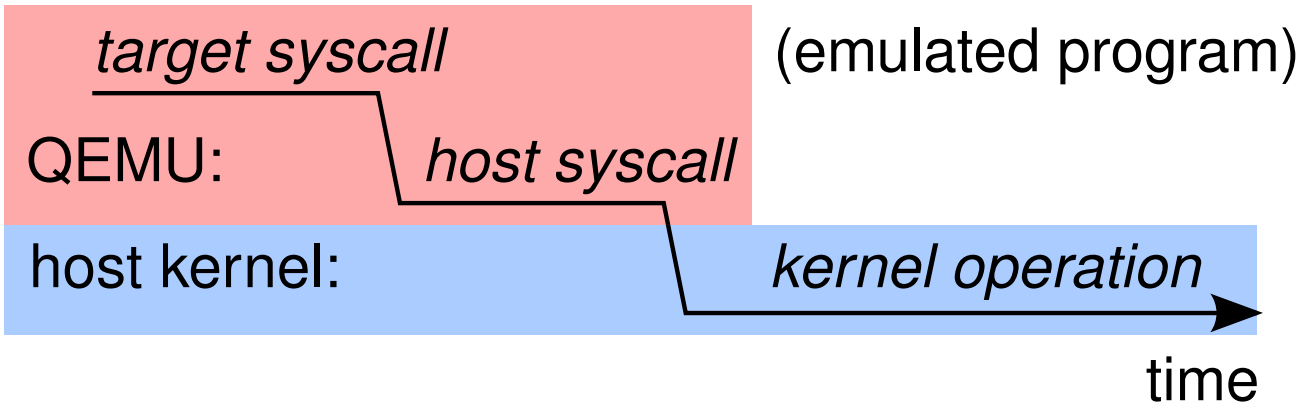

Figure: QEMU user-mode control flow

PRoot + QEMU user-mode control flow

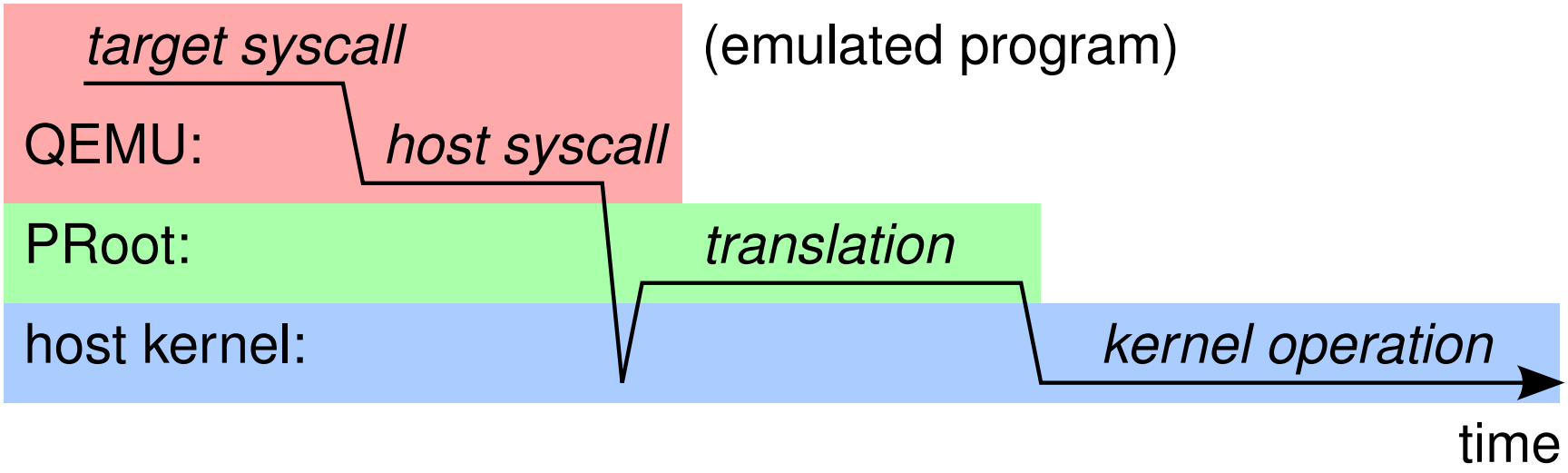

Figure: PRoot + QEMU user-mode control flow

PRoot + QEMU user-mode control flow

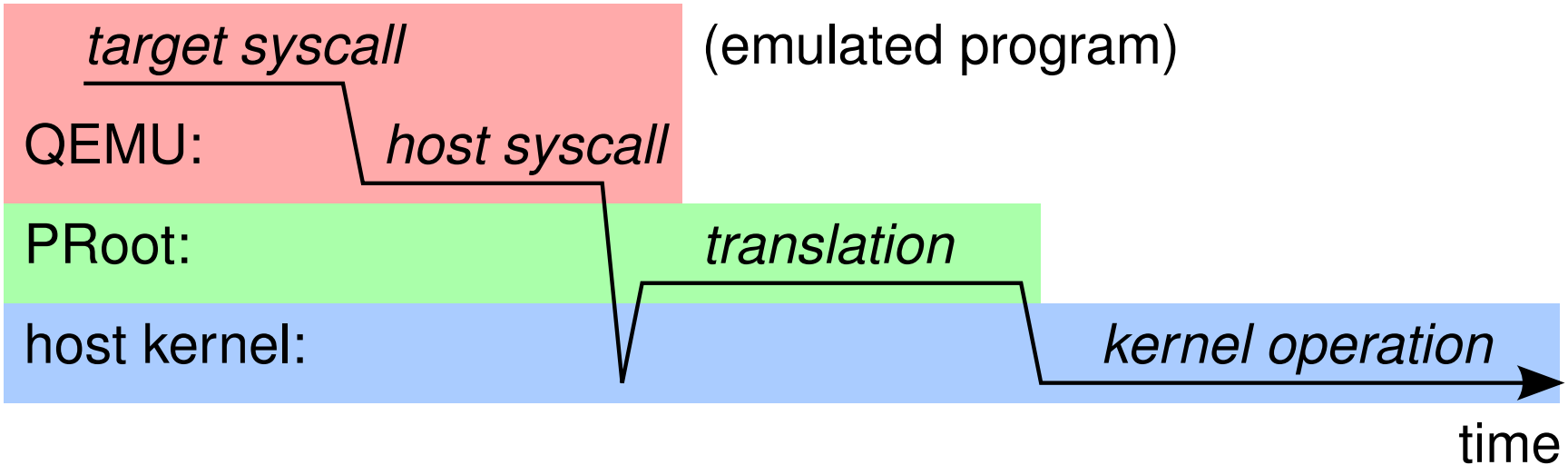

Figure: PRoot + QEMU user-mode control flow

```
open "/lib/ld-linux.so.2"
    -> open "${target_rootfs}/lib/ld-2.7.so"
```

PRoot + QEMU user-mode control flow

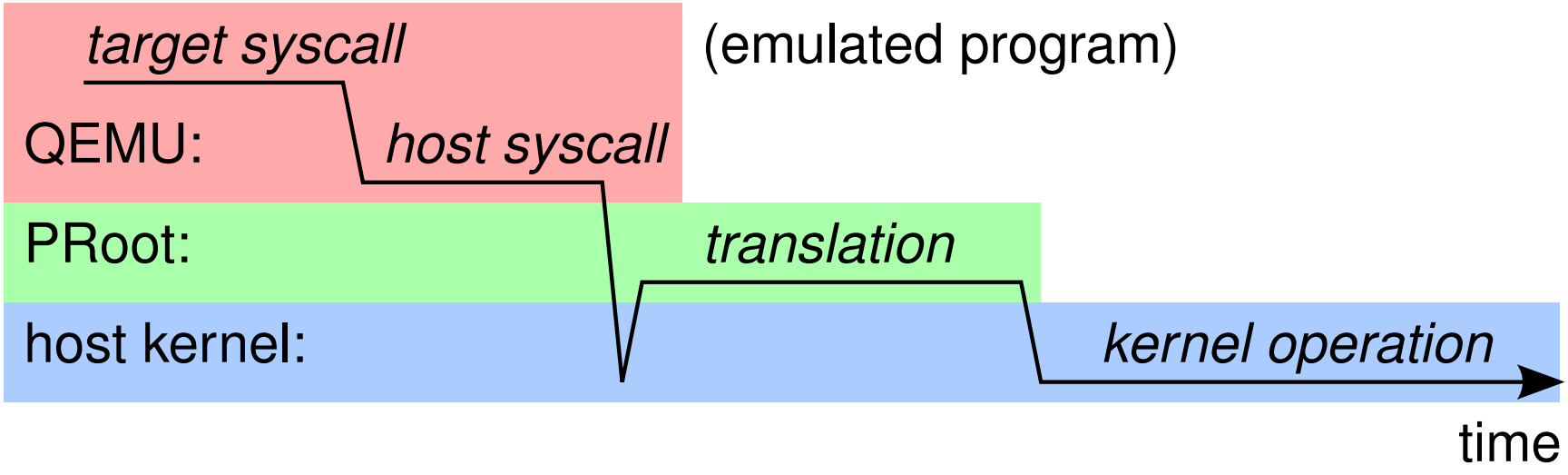

Figure: PRoot + QEMU user-mode control flow

```
open "/lib/ld-linux.so.2"
    -> open "${target_rootfs}/lib/ld-2.7.so"

exec "/bin/ls"
    -> exec "qemu-arm", "/bin/ls"
```

PRoot + QEMU user-mode *vs* QEMU system-mode

Table: PRoot + QEMU user-mode *vs* QEMU system-mode speed-up

Build step	Perl v5.10.0	Coreutils v6.12
archive extraction	3.6× faster	2.7× faster
configuration	2.0× faster	4.0× faster
build	2.9× faster	3.5× faster
validation	4.1× faster	3.6× faster

▸ Annex A

Typical example

You just have to specify the QEMU and the target rootfs:

```
$ proot -Q qemu-arm ./armedslack-12.2
```

Typical example

You just have to specify the QEMU and the target rootfs:

```
$ proot -Q qemu-arm ./armedslack-12.2
```

From that point every process is QEMUlated:

```
$ file `which file`
/usr/bin/file: ELF 32-bit LSB executable, ARM [...]
```

Typical example

You just have to specify the QEMU and the target rootfs:

```
$ proot -Q qemu-arm ./armedslack-12.2
```

From that point every process is QEMUlated:

```
$ file `which file`
/usr/bin/file: ELF 32-bit LSB executable, ARM [...]
```

The build and validation of a target package is straight forward, no need for cross-compilation support:

```
$ tar -xf perl-5.10.0.tar.gz; cd perl-5.10.0
$ ./Configure -de
$ make -j 4
$ make -j 4 test
```

Conclusion

- PRoot + QEMU user-mode: an extended user-mode;
- no setup, no configuration, no administrative privilege;
- and compatible with any version of QEMU.

PRoot soon to be published

- GPL v2+ license
- **`http://proot.me`**

Any feedback and suggestion welcomed! Thanks!

Annex A: Comparison with other tools

Back

Table: Testing SB2+QEMU user-mode *vs* PRoot+QEMU user-mode

Package	SB2 v2.2	PRoot v0.5	system-mode
Perl v5.10.0	99.6%	99.6%	99.8%
GNU Coreutils v6.12	94.9%	97.3%	96.7%

- We tested many tools, but only two survive the tests above.
- We did not consider solutions with administrative privilege.
- Both PRoot and SB2 have robust path canonicalization algorithms.
- PRoot and SB2 differ in many aspects (design, usage, . . .).