Transplantation of VirtualBox to the NOVA microhypervisor

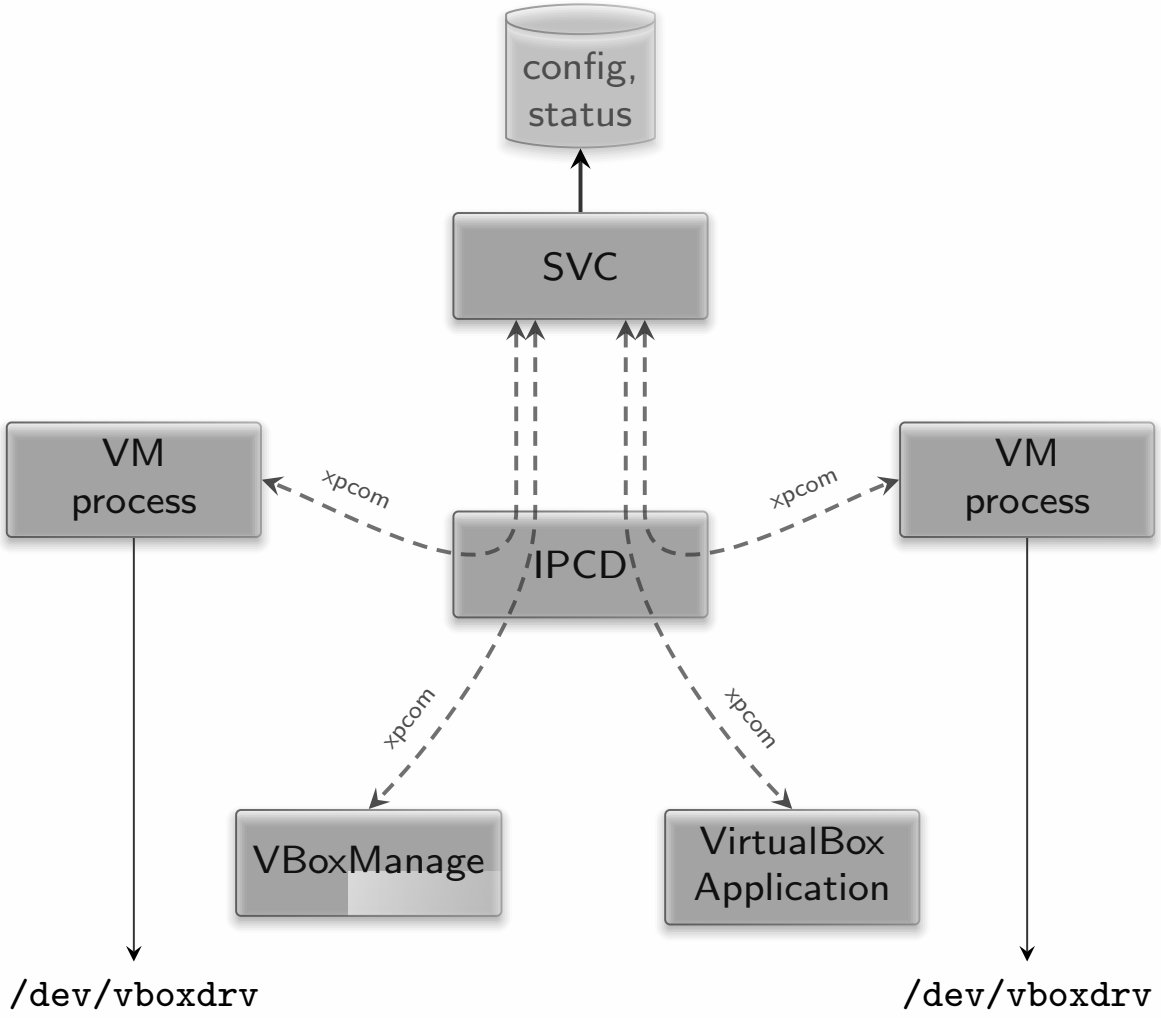
Norman Feske
<norman.feske@genode-labs.com>

Outline

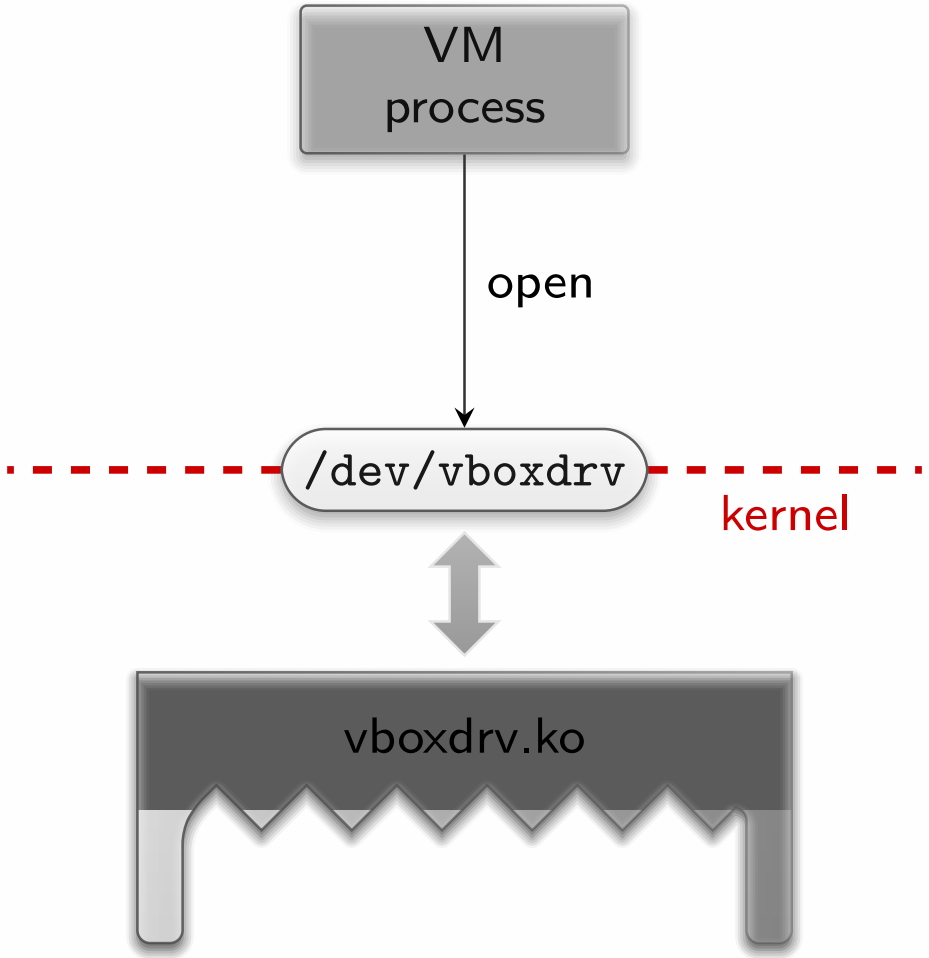
1. VirtualBox
2. NOVA microhypervisor and Genode
3. Steps
4. Demo + Outlook
5. War stories

Outline

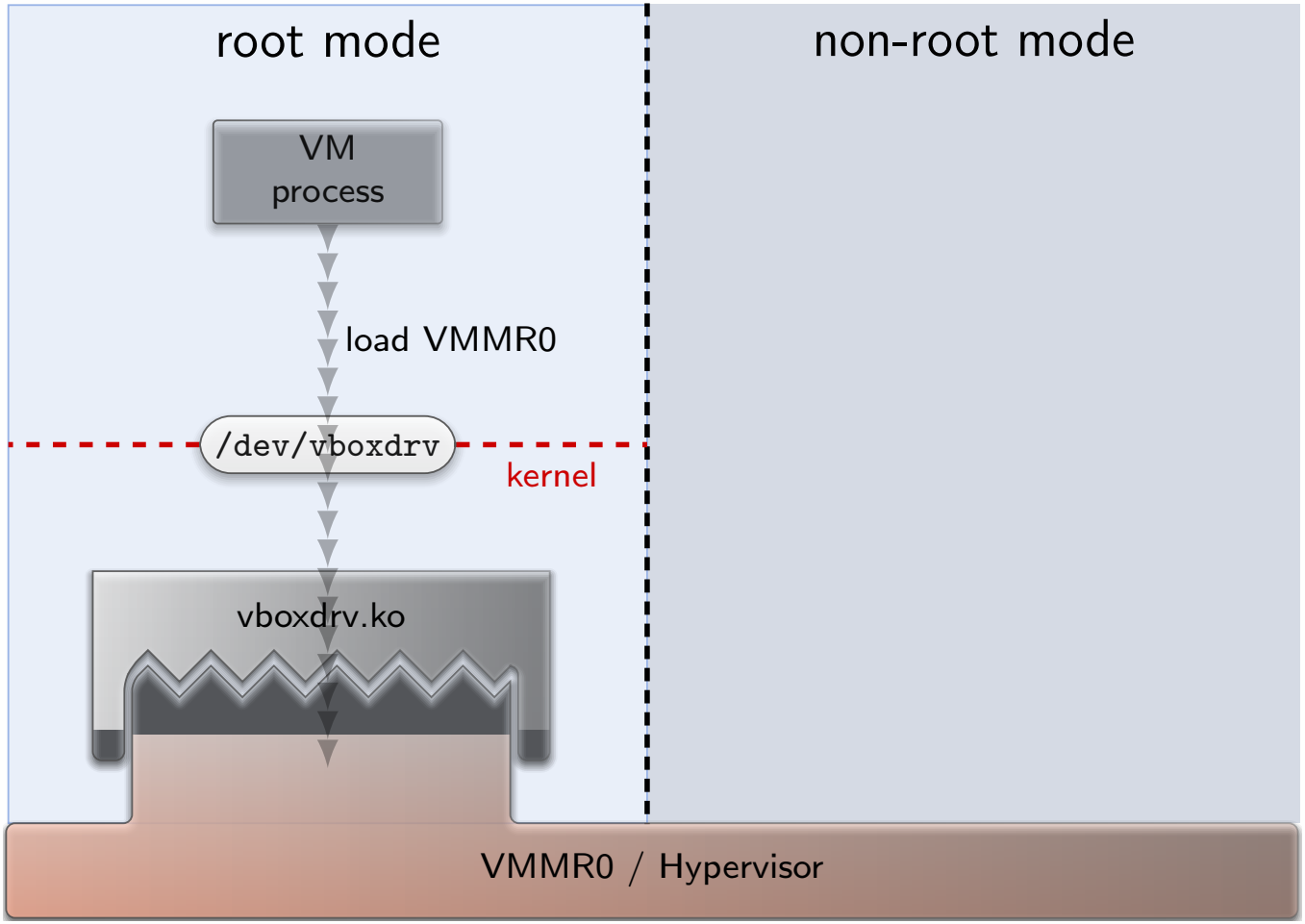
Architecture overview



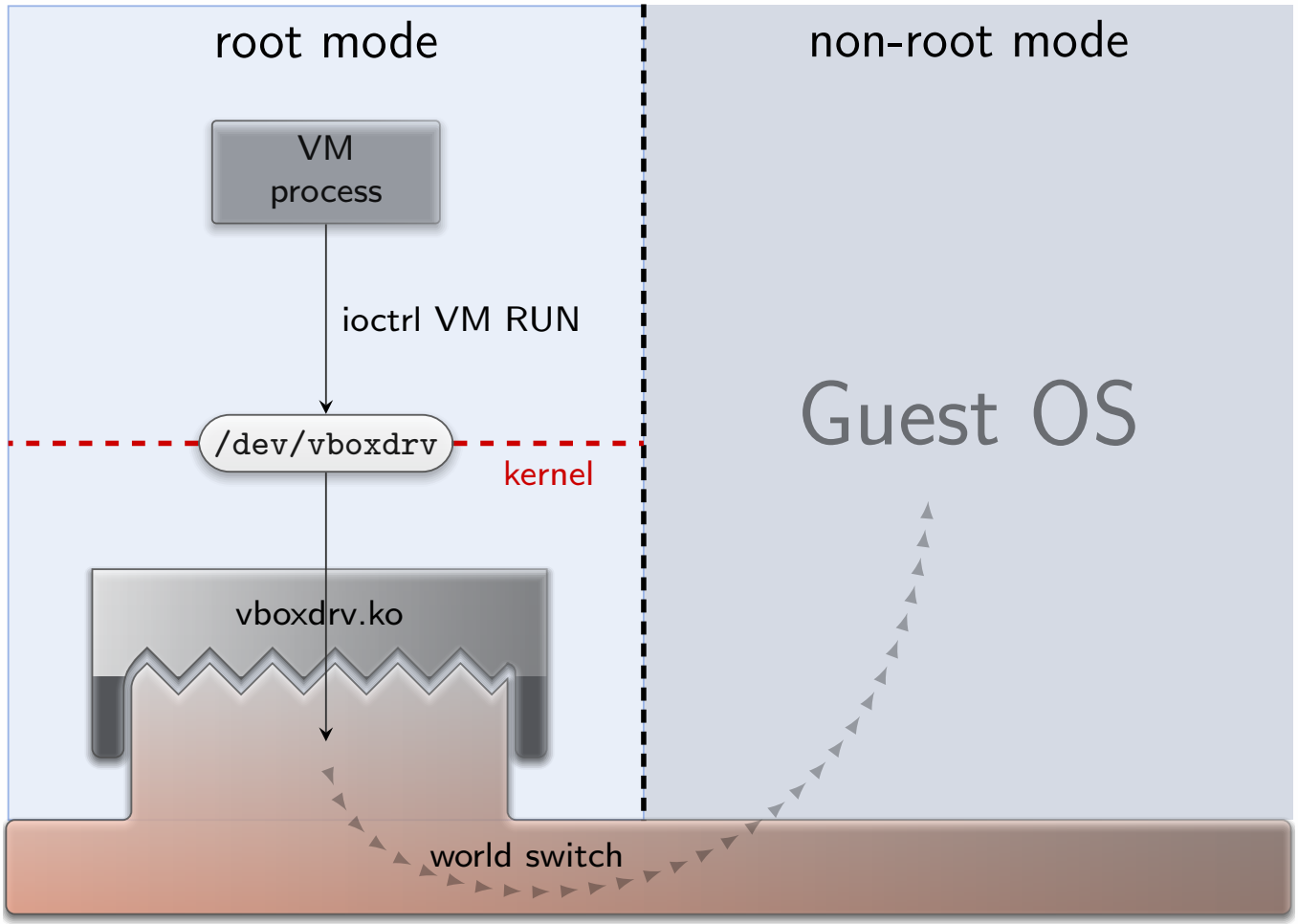
Starting up a VM process



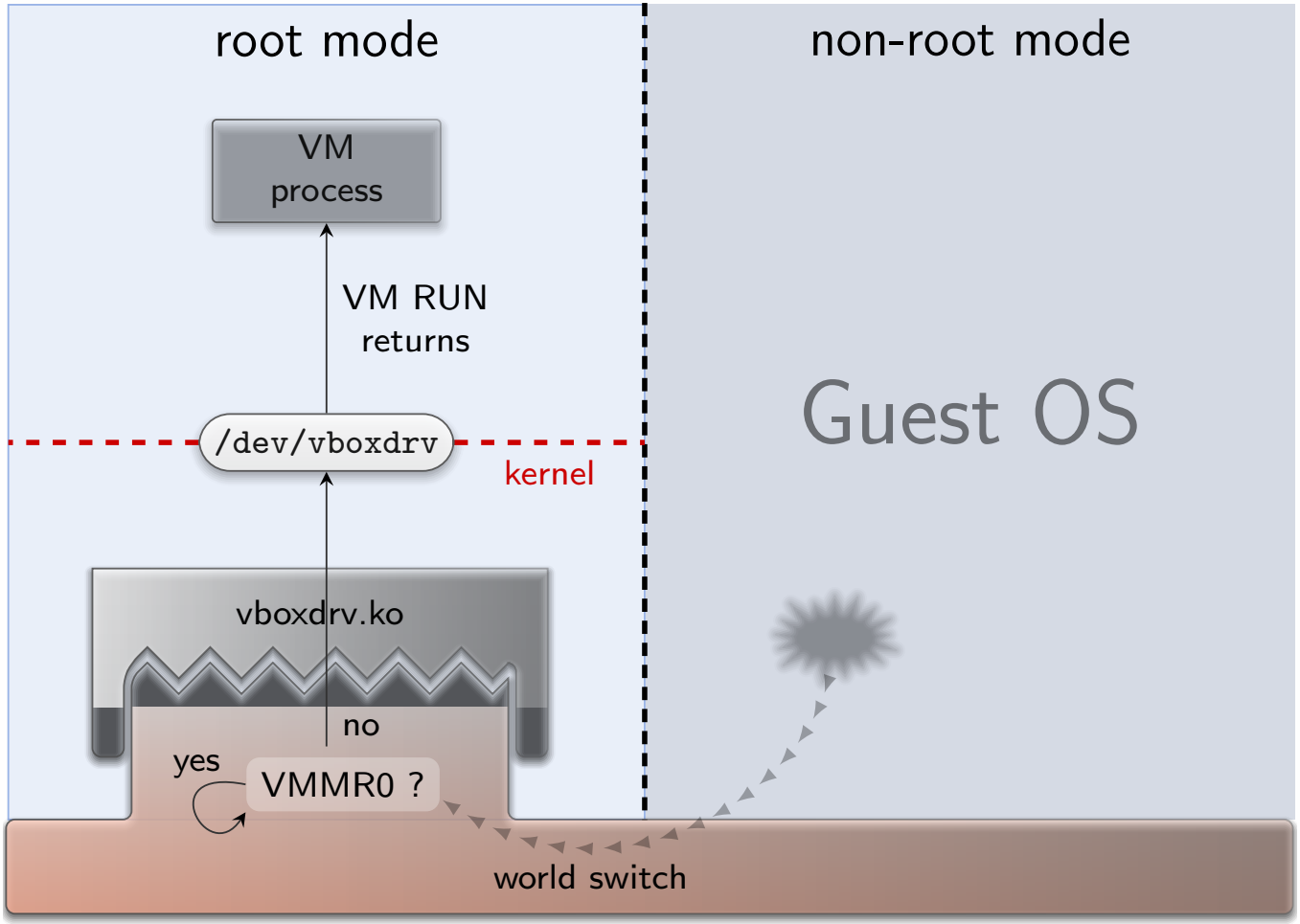
VM process running



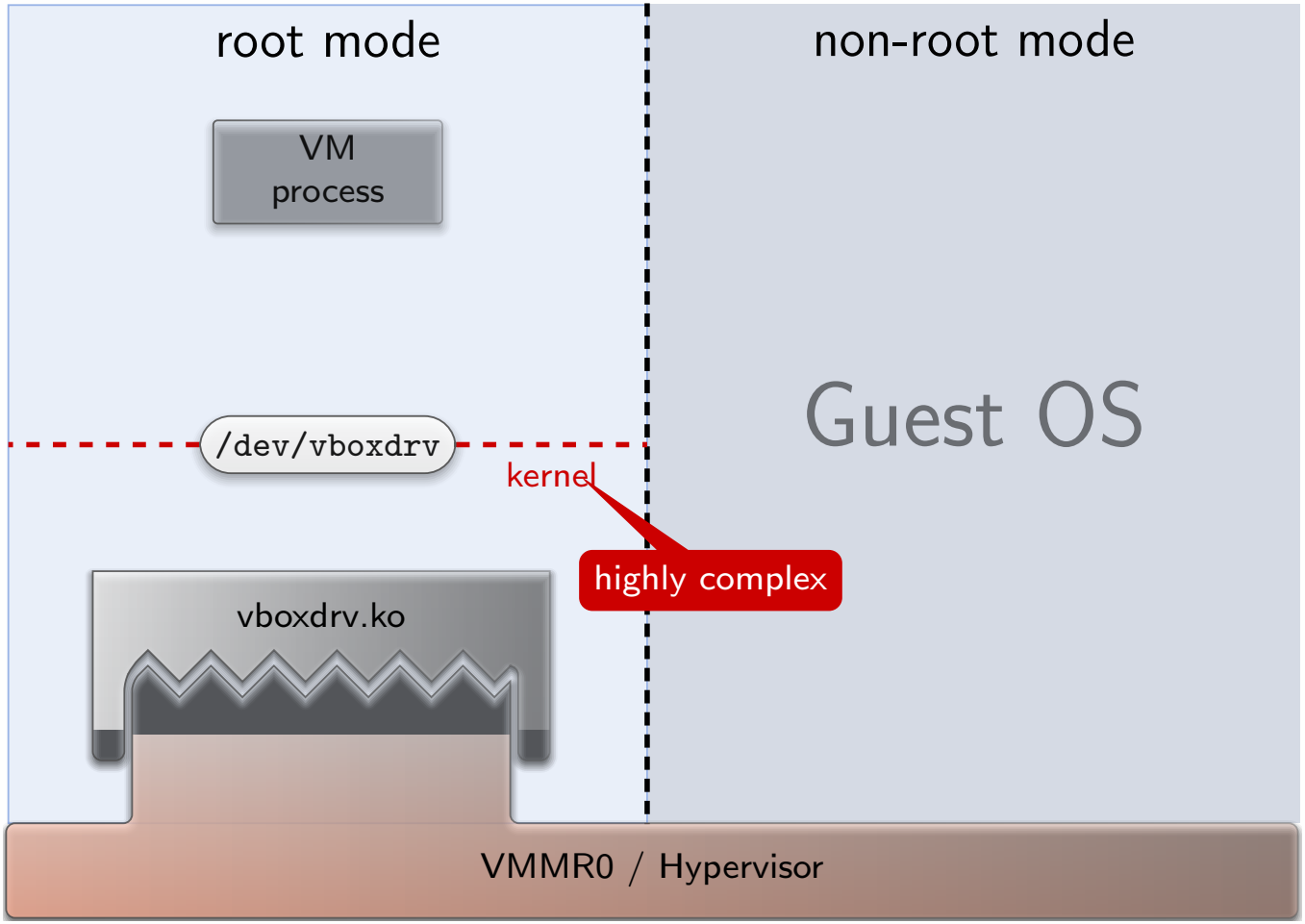
Entering the Guest OS



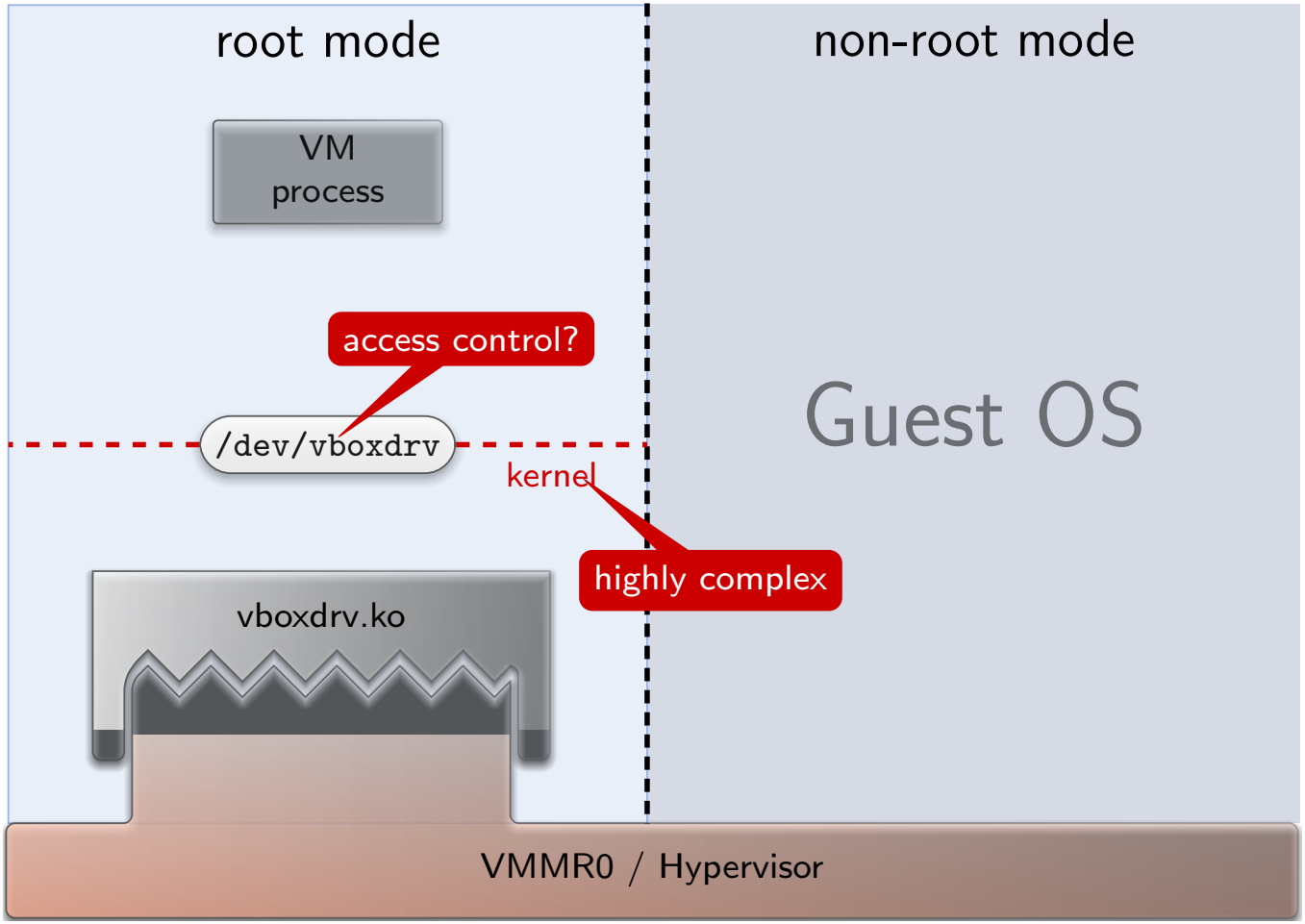
Flow of a virtualization event



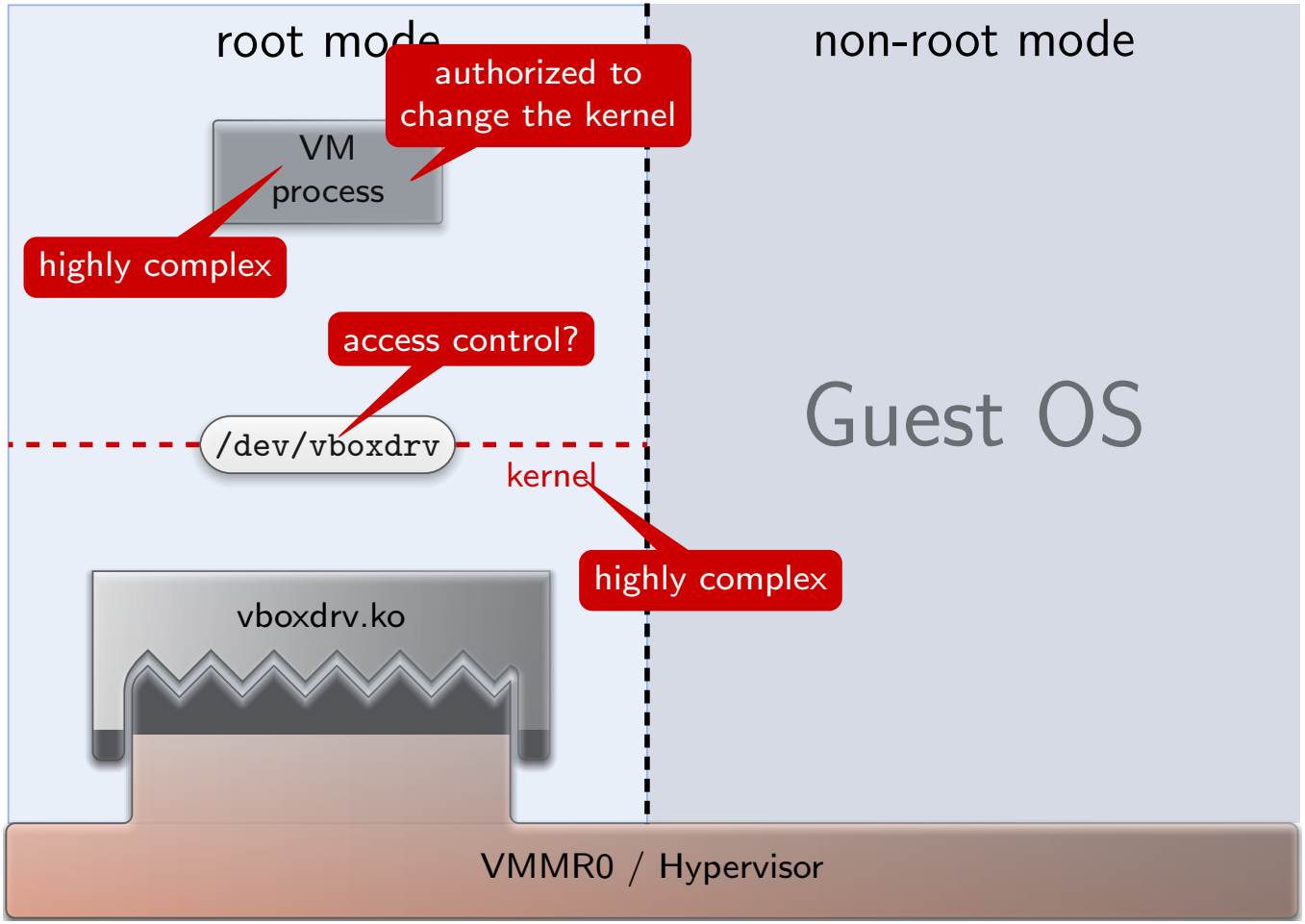
Risks for desktop virtualization



Risks for desktop virtualization



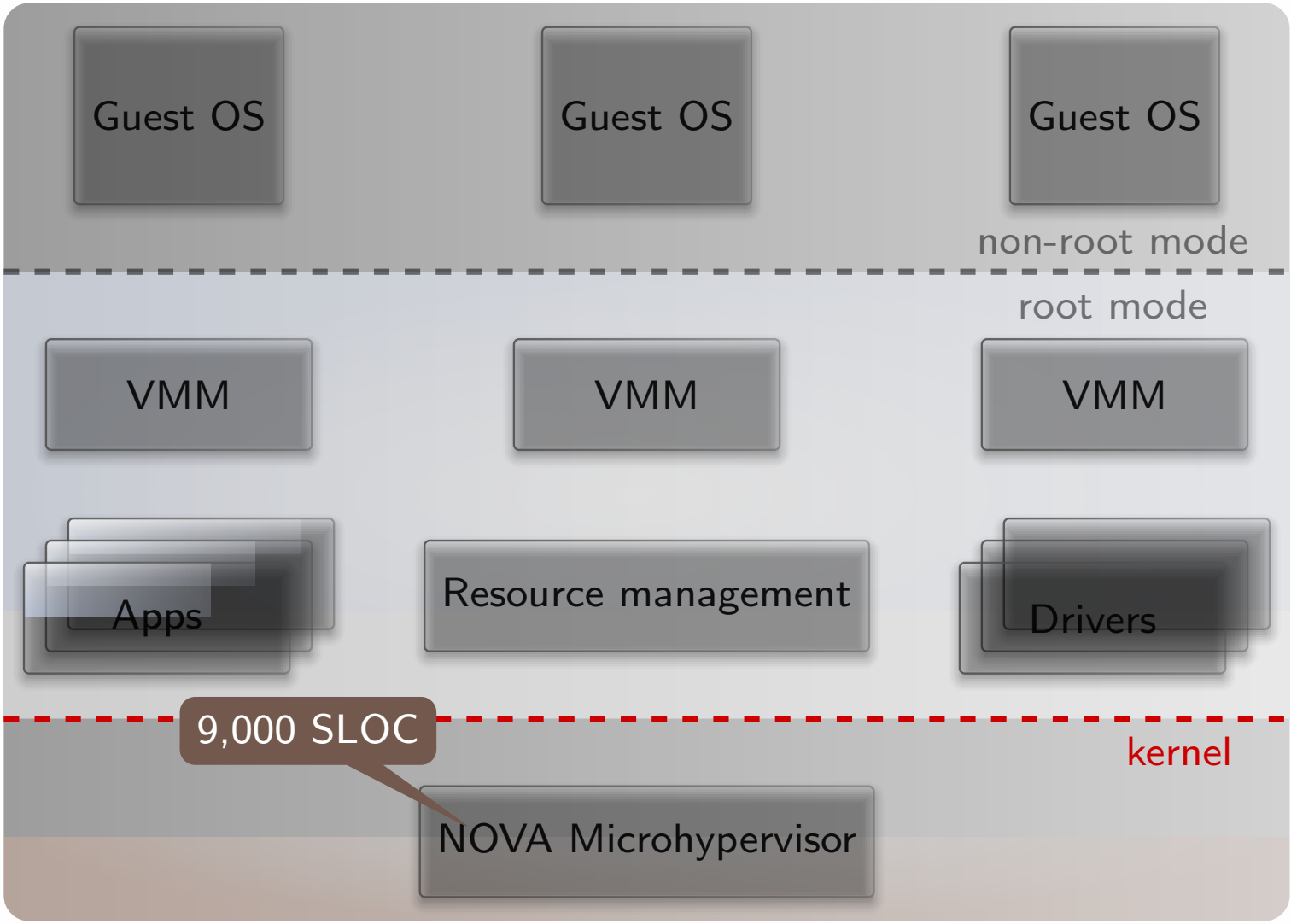
Risks for desktop virtualization



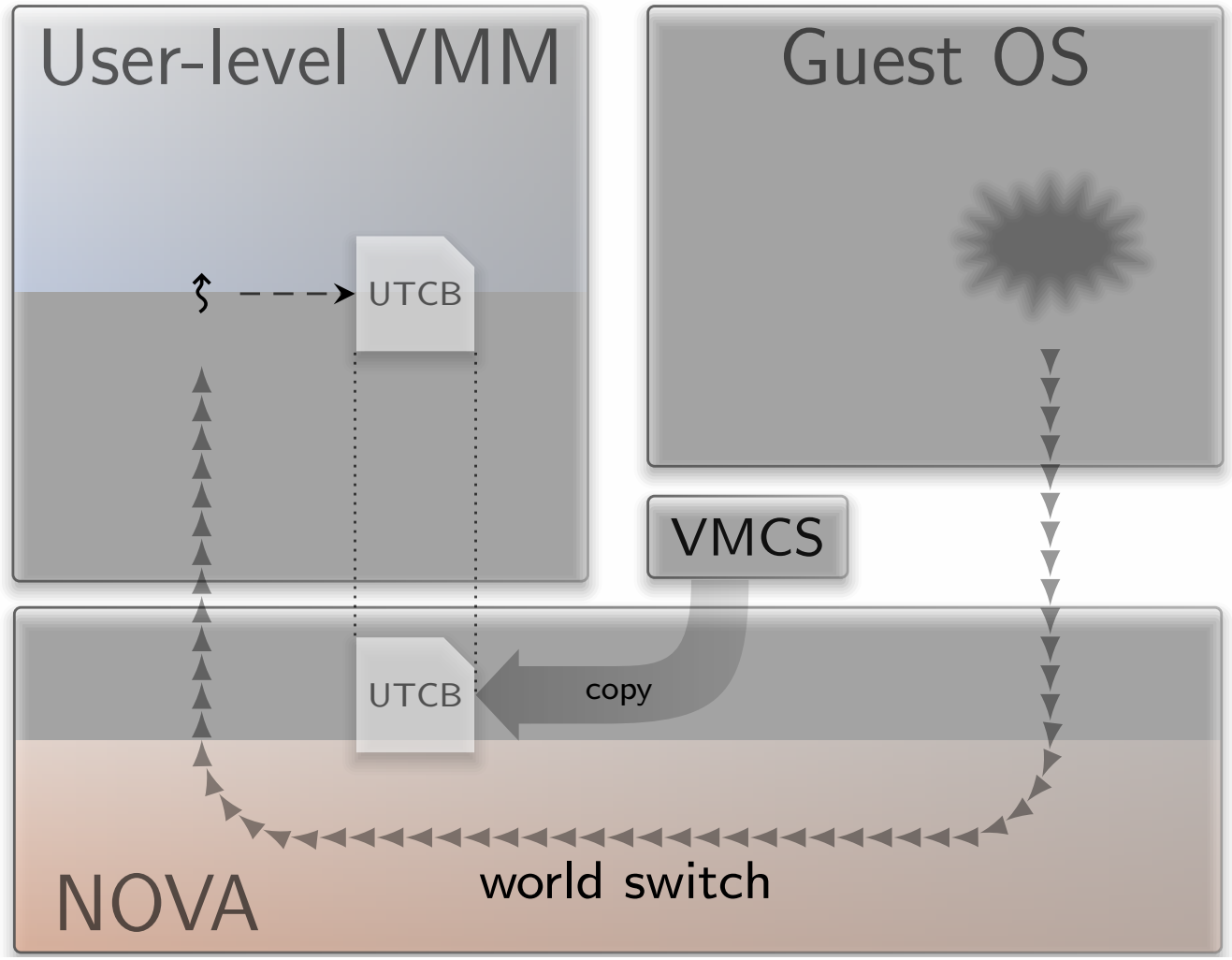
Outline

1. VirtualBox
2. NOVA microhypervisor and Genode
3. Steps
4. Demo + Outlook
5. War stories

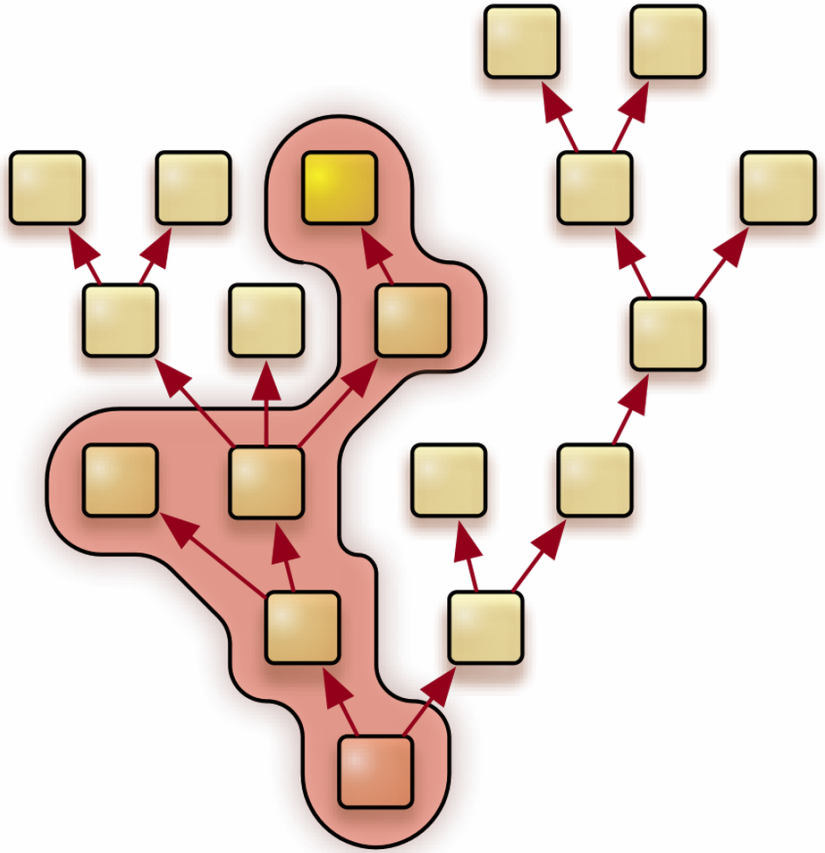
NOVA architecture



Flow of a virtualization event

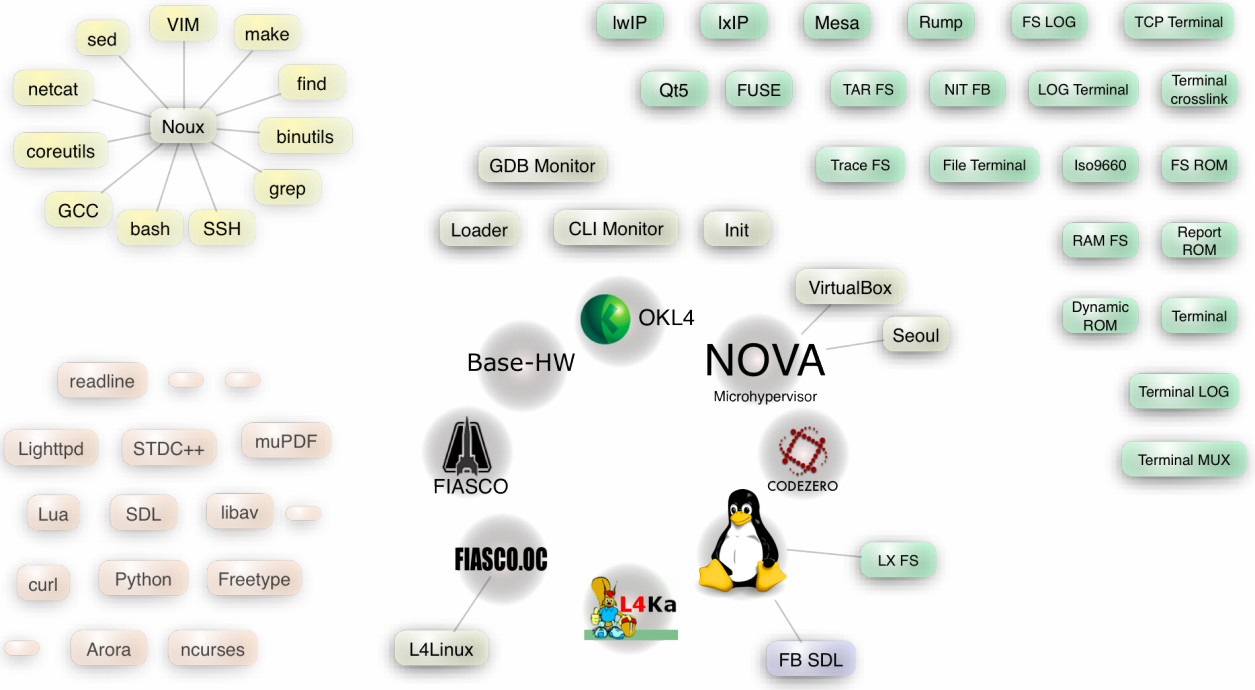


Genode OS architecture

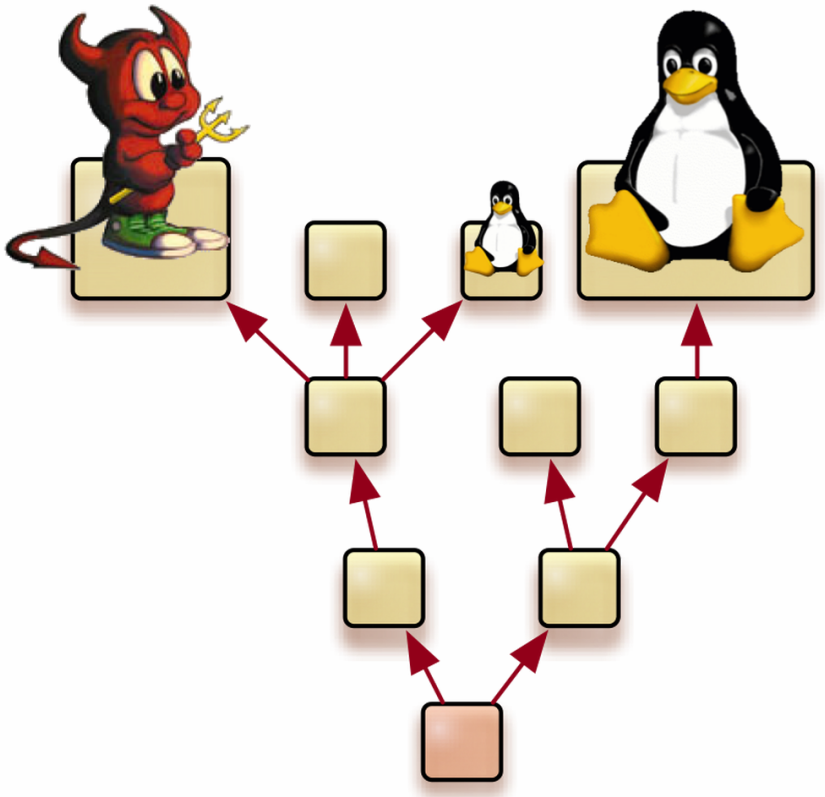


→ Application-specific TCB

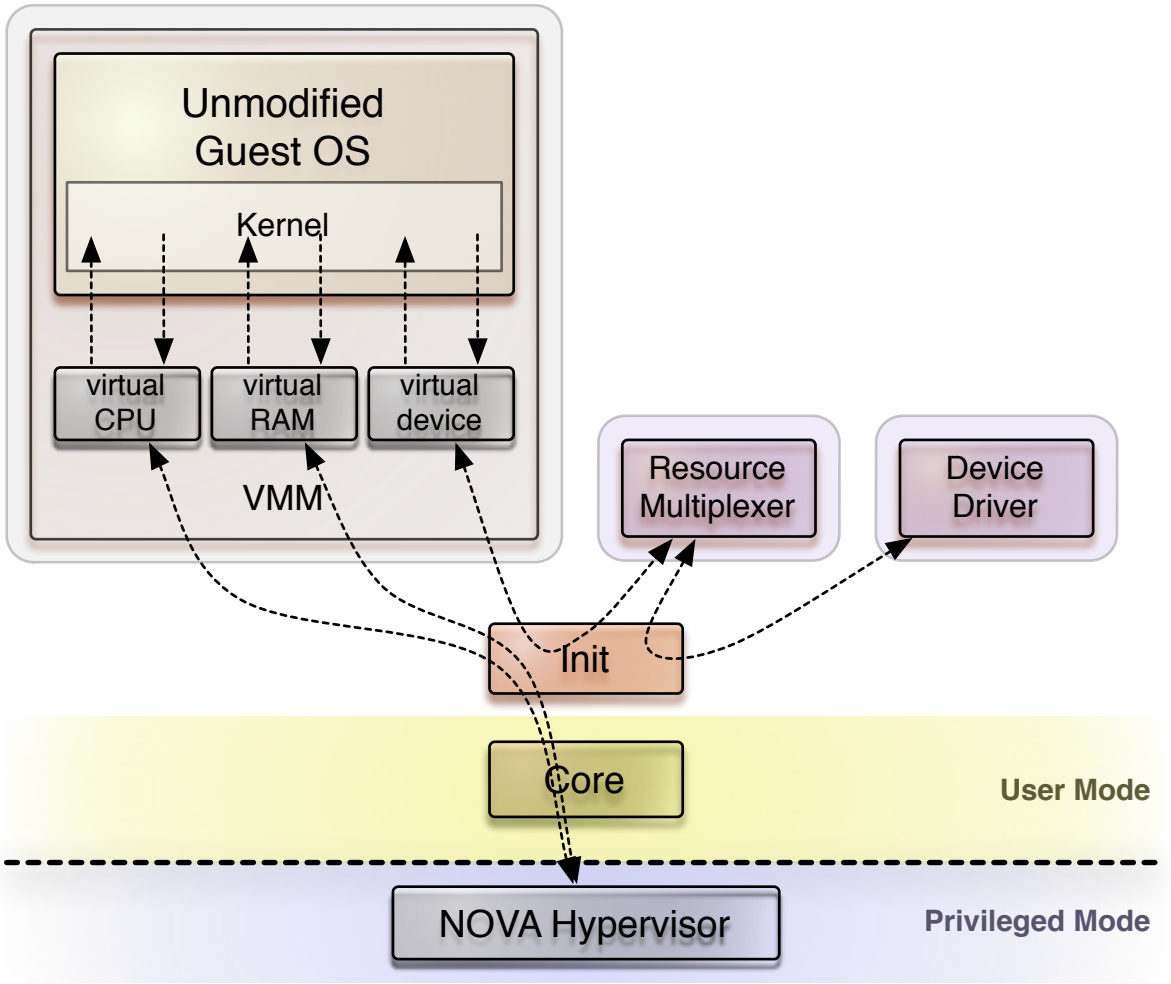
Genode OS framework



Genode combined with virtualization



Seoul VMM on top of Genode/NOVA



Idea

Device models and features of VirtualBox

+

Security of the Genode/NOVA architecture

Outline

Identify the interesting parts

Entire VirtualBox code base

$>$ 4 million lines of code (sloccount)

Narrowed to the interesting parts

$>$ 2 million lines of code

```
src/VBox/VMM
src/VBox/Main
src/VBox/Runtime
src/VBox/Devices
src/VBox/Storage
src/VBox/GuestHost
src/VBox/Disassembler
src/VBox/HostServices
src/recompiler
src/libs/liblzf-3.4
src/libs/liblzf-3.4/cs
src/libs/libxml2-2.6.31
src/libs/zlib-1.2.6
include/VBox
include/iprt
```

Porting the VirtualBox Runtime to Genode

- Facilitate Genode's existing infrastructure
  - 3rd-party software management tools
  - FreeBSD libc
  - Standard C++ library
  - POSIX threads

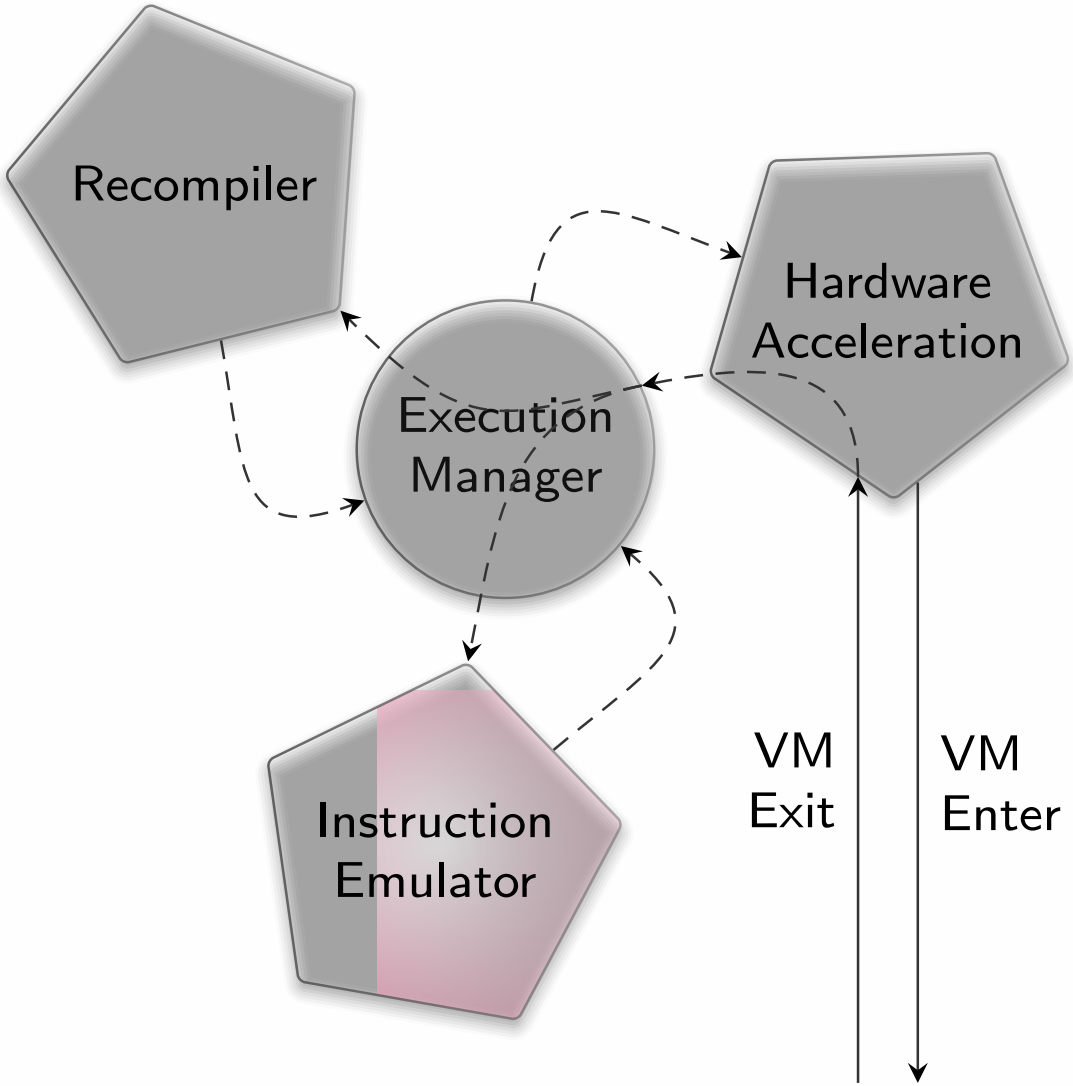
→ Most parts of the POSIX runtime could be reused

VM process initialization

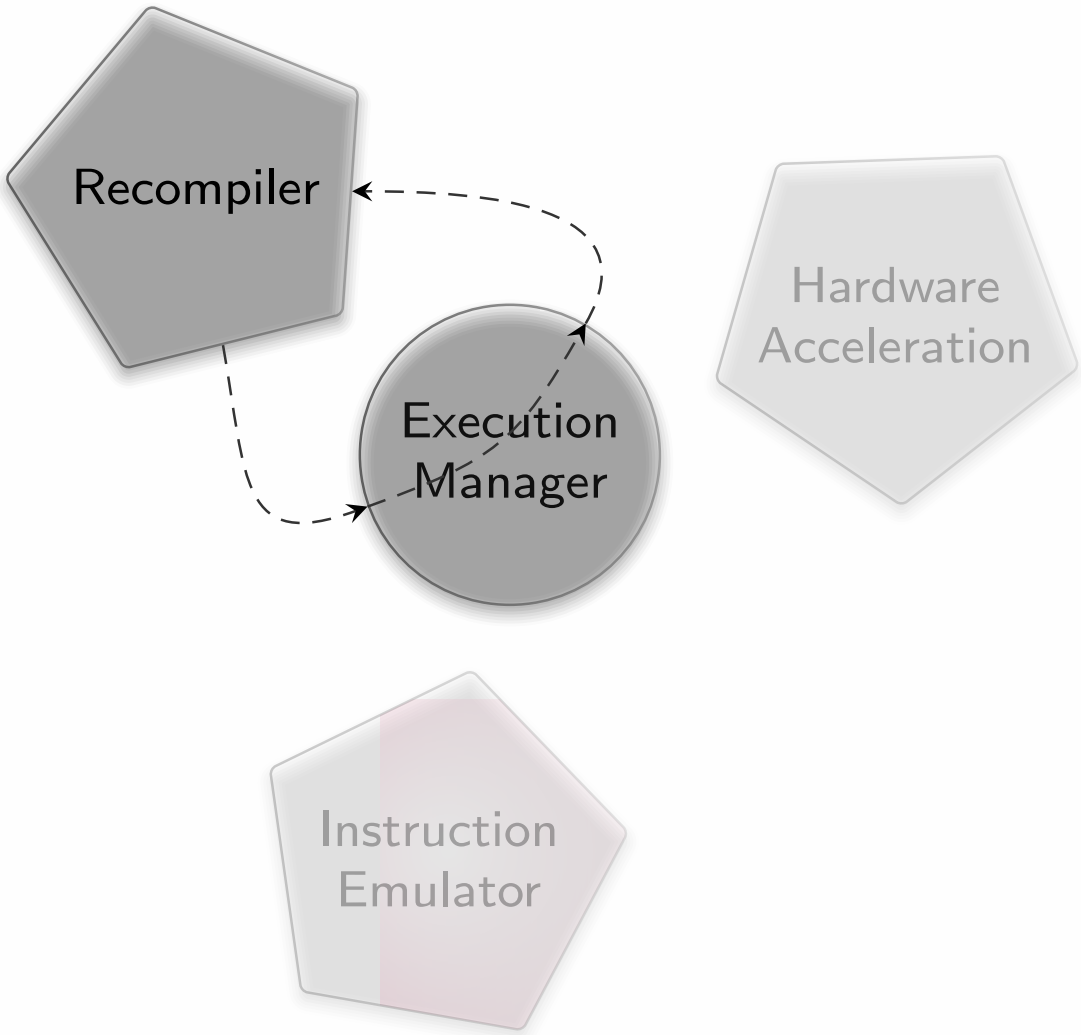
Enable subsystems one by one

- Guest memory (accessed by recompiler and device models)
  RAM, MMIO
- I/O-port handling
- PGM, HWACCM, TM
- Device models, PDM, BIOS
- Host drivers
  - Using the “Basic front end”
  - Reimplement `SDLConsole` interface

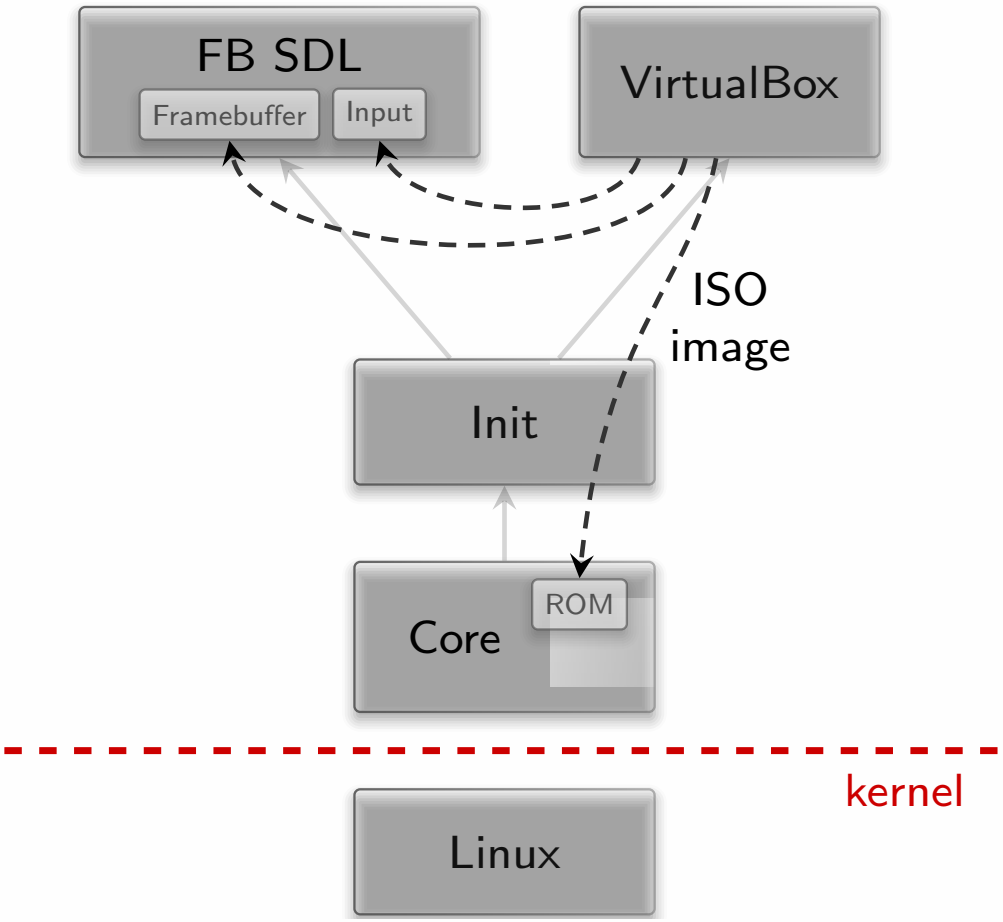
A look inside a VM process



Start with executing the recompiler only



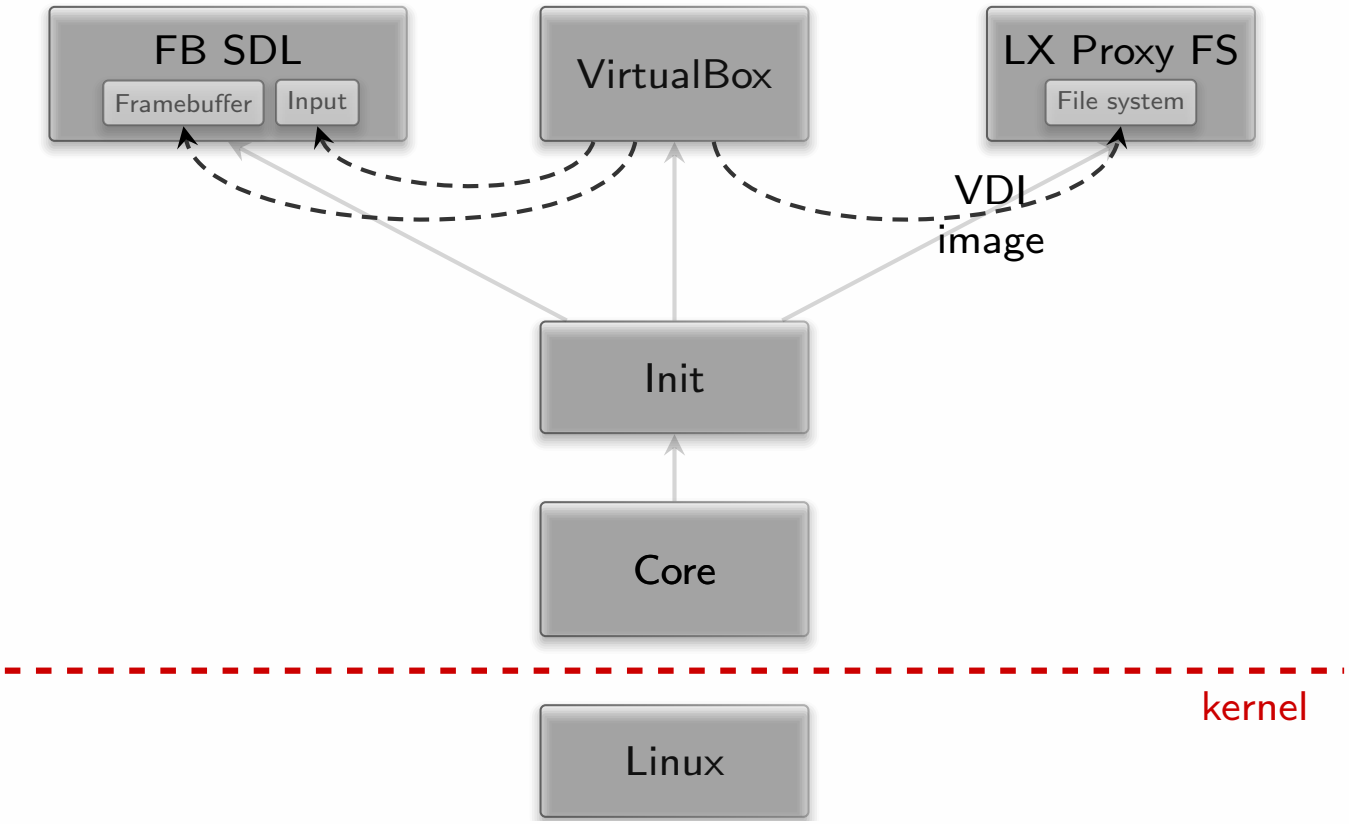
Simple test scenario



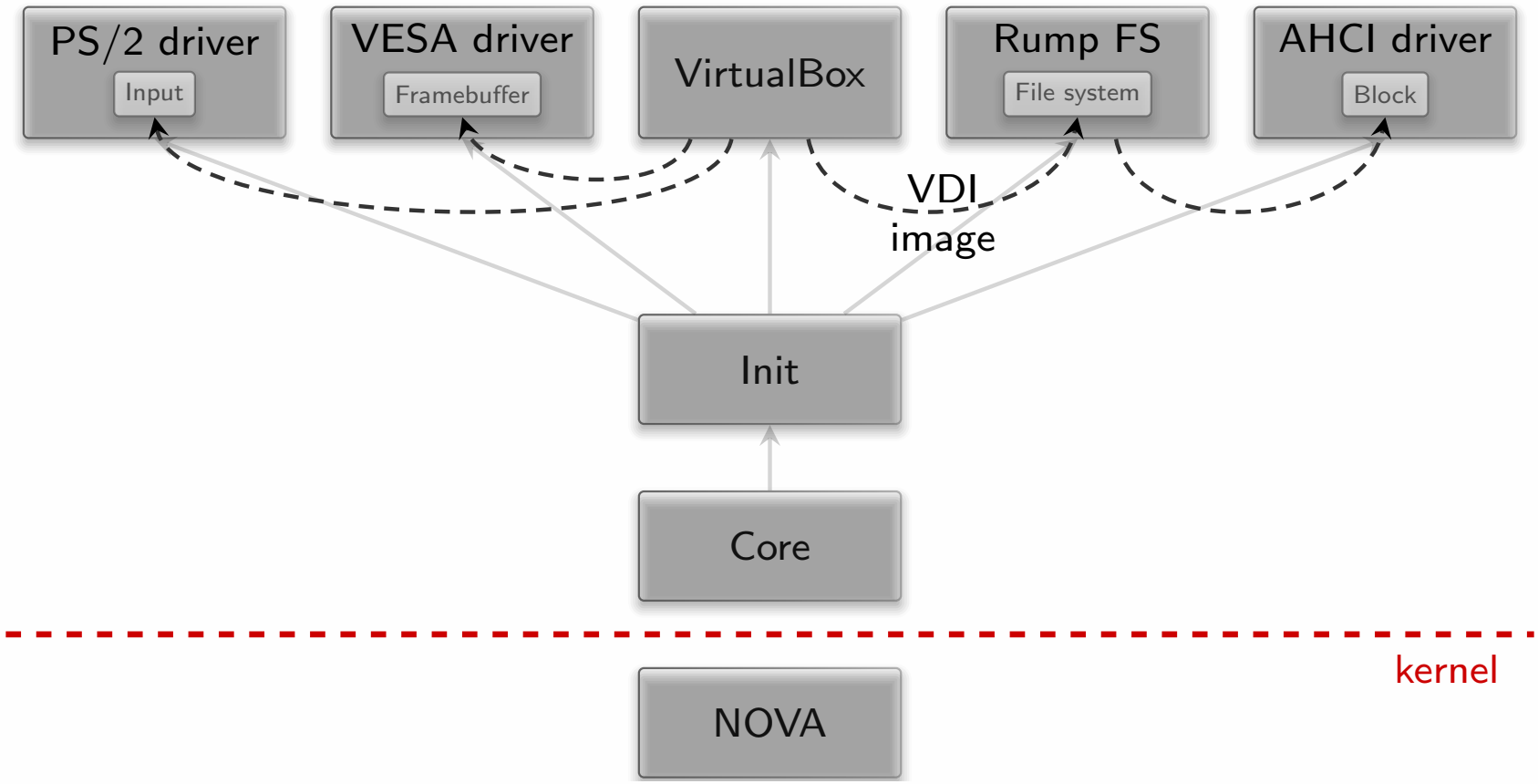
Increasing guest complexity

1. Custom-made Genode OS scenarios
2. Small Linux-based images (Tinycore, GRML)
3. Windows XP

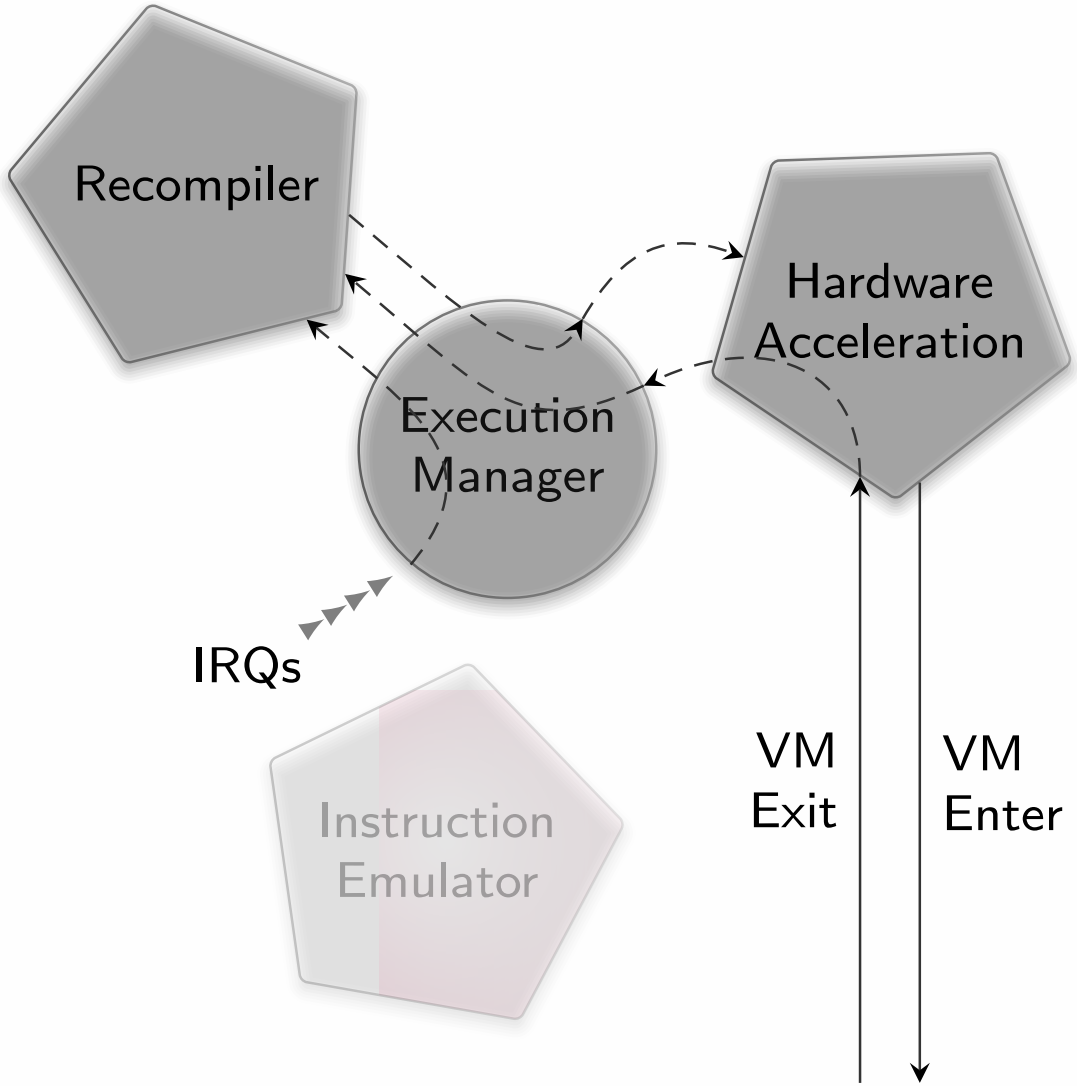
Windows XP as a guest



Move scenario to NOVA



Entering non-root mode



Entering non-root mode

- VBox VM state $\leftrightarrow$ NOVA UTCB state
- Virtualization of guest memory
  (EPT faults)
- Enter VT-x conservatively
  (if protected mode and paging enabled)
- Inject IRQs into recompiler
- Later: IRQ injection via NOVA into VT-X

Adding features

Additional drivers

- Networking

Guest tools

- Shared folders
- Host clock
- Mouse-pointer synchronization

Update to VirtualBox 4.3

- Basic front end no longer supported

- Use of main front end code to NOVA port
  - Custom console implementation
  - Shortcut XPCOM middleware

  → Support for using `.vbox` files

Outline

Demo

Windows 7 running in VirtualBox directly on top of NOVA

Adaptation of VirtualBox to Genode/NOVA

Ported code

- 400,000 lines of code (sloccount)

New code

- 6,200 lines (sloccount)
  hm, iommio, ioport, mm, pdm, pgm, sup

Modifications of the original code

- 510 lines added
- 120 lines removed

Current state and outlook

- Usable performance, optimization ongoing
- Focused on VT-X, SVM not regularly tested
- **Reduces TCB complexity to two orders of magnitude**
- Useful for building appliances in high-security computing
- Stepping stone for using Genode as a general-purpose OS

Outline

War stories

- Invalid guest state
- TLB consistency
- Interrupt handling
- Large files in shared folders

Thank you

Genode OS Framework

`http://genode.org`

Genode Labs GmbH

`http://www.genode-labs.com`

Source code at GitHub

`http://github.com/genodelabs/genode`