

Kexec/Kdump under the hood

Matthias Brugger
Linux Kernel Engineer
mbrugger@suse.com

Me...

- Software developer at SUSE
- Working on ARM64
- Maintainer of Mediatek SoCs in the Linux kernel
- ... and much more.
- Twitter @bruggems



Outline

- Use cases
- User-space internals
- Kernel internals
- Support in openSUSE
- Q&A



Use cases

The background features abstract geometric shapes in two shades of green. A large, dark teal shape occupies the left and top-left portions of the frame. To its right, a lighter green shape is partially visible, separated by a white diagonal line. The overall design is minimalist and modern.

Use cases

- Debug a system
 - No serial console
 - No reproducer inhouse
 - No good logs
- Boot a new kernel without rebooting machine
 - Faster machines have slower firmware
- Boot your system
 - That must be s390!

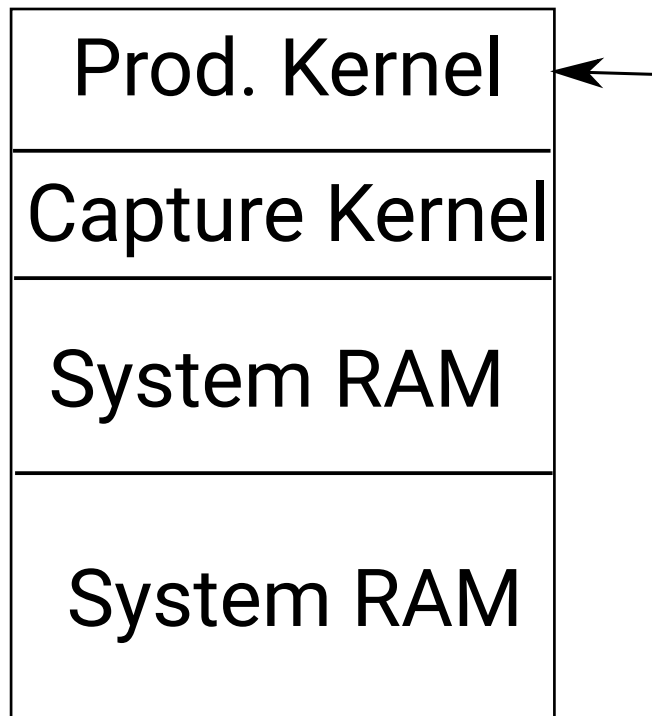


Some comments

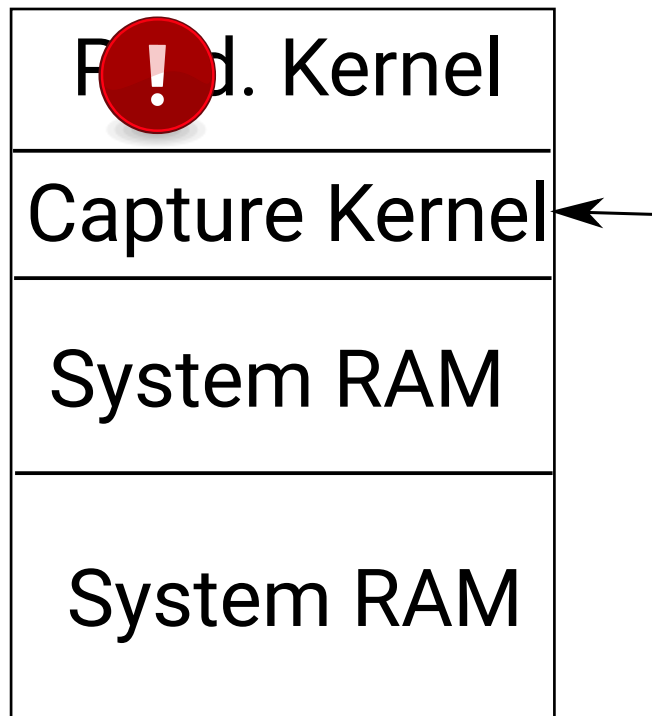
- Production- vs Capture-Kernel
- Capture kernel gets loaded when production kernel crashes
 - Creates a dump of memory
 - Save memory dump
 - The dump can later be inspected



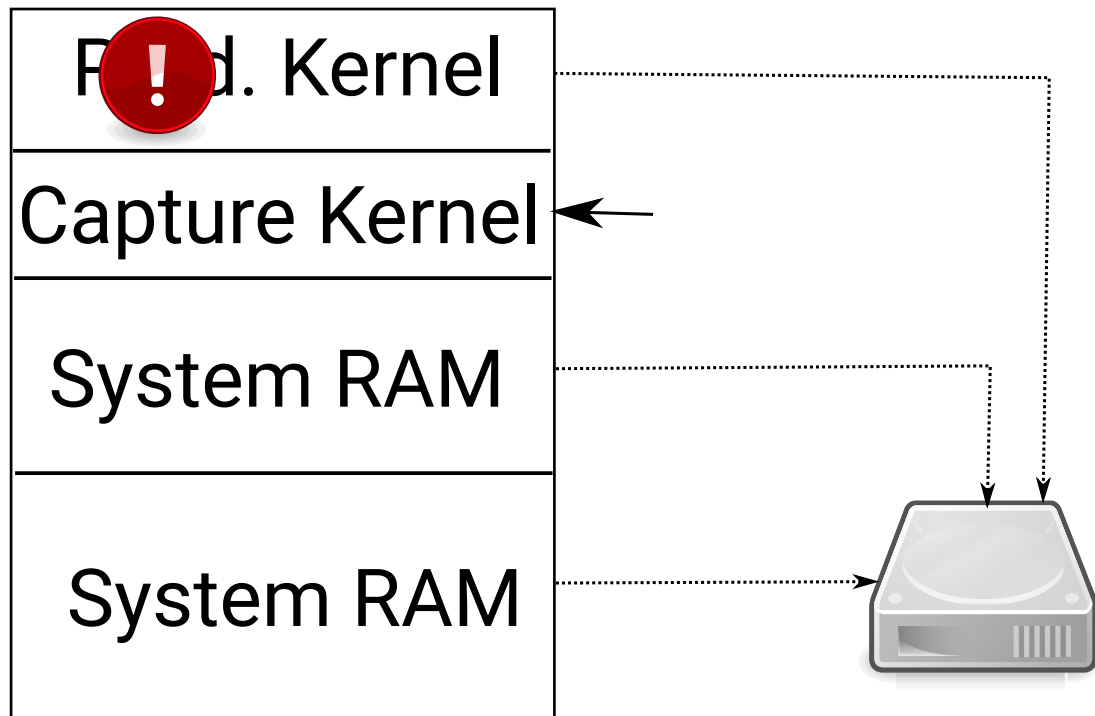
Use cases



Use cases



Use cases



Parts involved

Parts involved

- kexec-tools (user-space) – prepare capture system
- Kernel itself – executes capture kernel on crash
- Other userspace tools to inspect the dump
 - makedumpfile, crash and crash-python
- Distro programs to easier set things up
 - Kdump on openSUSE



kexec-tools

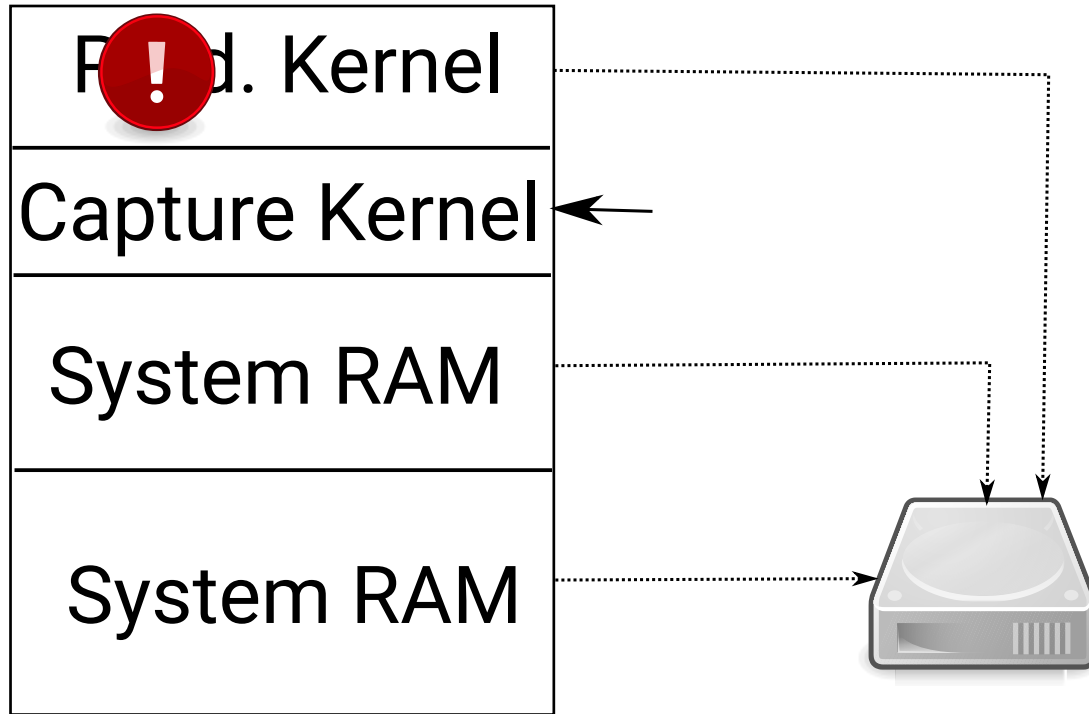
- `kexec -l /boot/vmlinuz --initrd=/boot/initrd --reuse-cmdline`
 - `-e` → execute (!) - reboot with magic value
 - `-p` → load capture kernel
 - `-l` → load kernel
 - `-u -up` → unload
 - Arch specific options
 - e.g. `--dtb`



The background features abstract geometric shapes in two shades of green. A large teal shape occupies the left and top portions, while a bright green shape is on the right. They are separated by a white diagonal line.

kexec-tools...
under the hood

Remember...



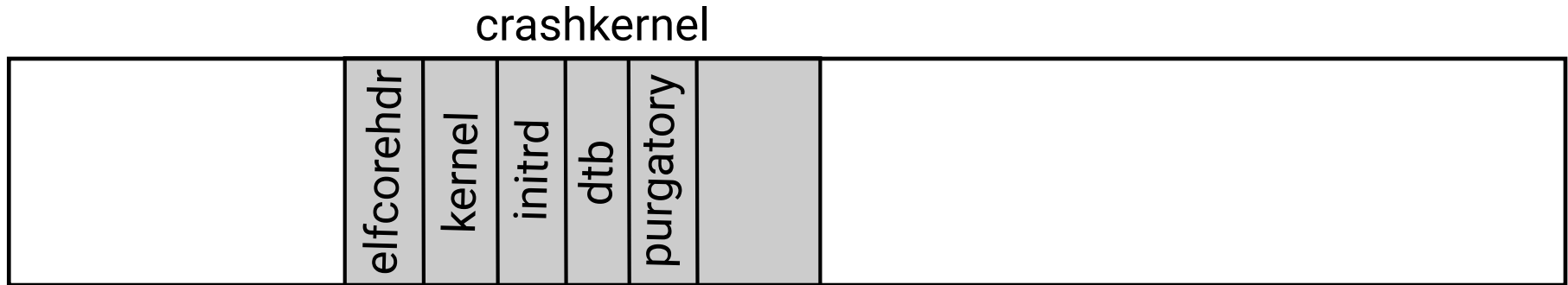
Questions

- When system crashes we need to know where is
 - Capture kernel
 - Usable memory for capture kernel
 - Capture kernel's initrd
 - Production kernel and memory (for the dump)



Memory in the production kernel

- Reserve memory for the capture kernel et. al.
- Production-Kernel boot parameter `crashkernel=`
 - Can be tricky to do



Memory in the production kernel

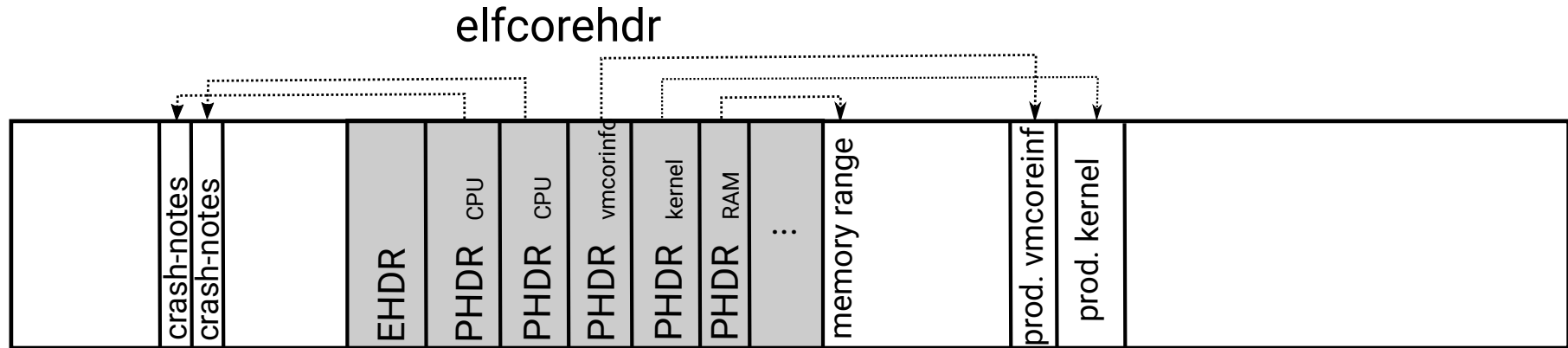
```
struct kexec_segment {  
    const void *buf;  
    size_t bufsz;  
    const void *mem;  
    size_t memsz;  
};
```

crashkernel



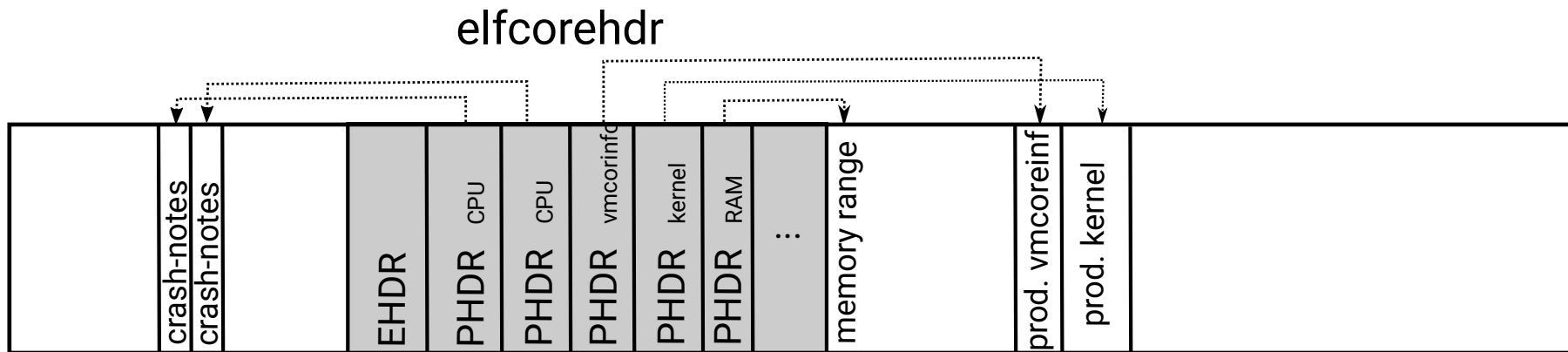
elfcorehdr

- Elf header information about production memory
- Capture kernel creates /proc/vmcore out of it
- Information is collected by kexec-tools



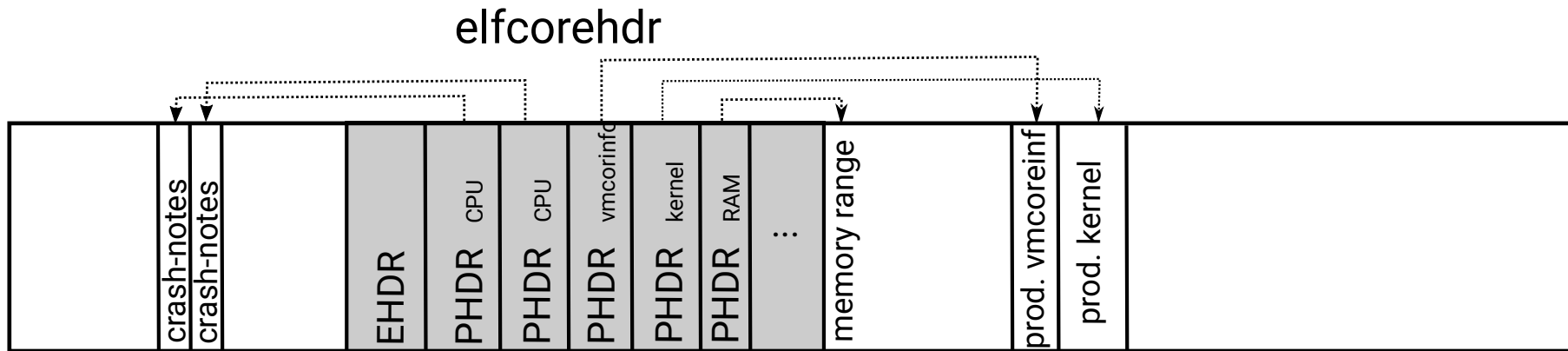
elfcorehdr

- Crashnotes
 - per-CPU area for storing CPU states, PID, CPU registers
 - `/sys/devices/system/cpu/cpu%d/crash_note`



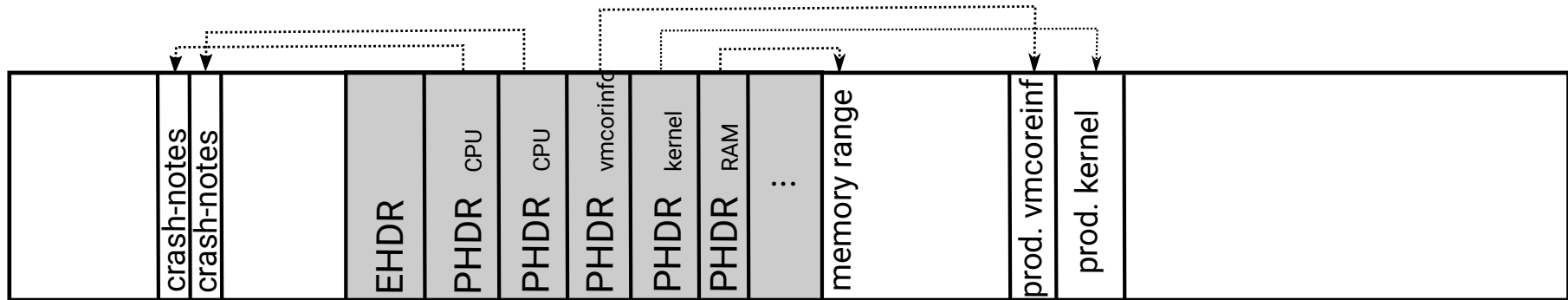
elfcorehdr

- vmcoreinfo
 - Kernel debug information
 - Size of a page, offset of flags in struct page
 - /sys/kernel/vmcoreinfo



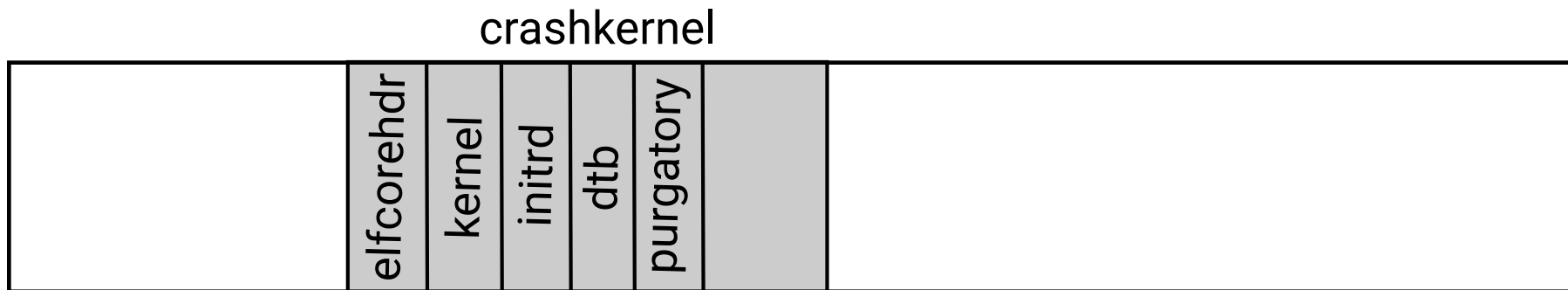
elfcorehdr

- Memory ranges
 - PT_LOAD
 - /proc/iomem
- Used to create /proc/vmcore dump file
elfcorehdr



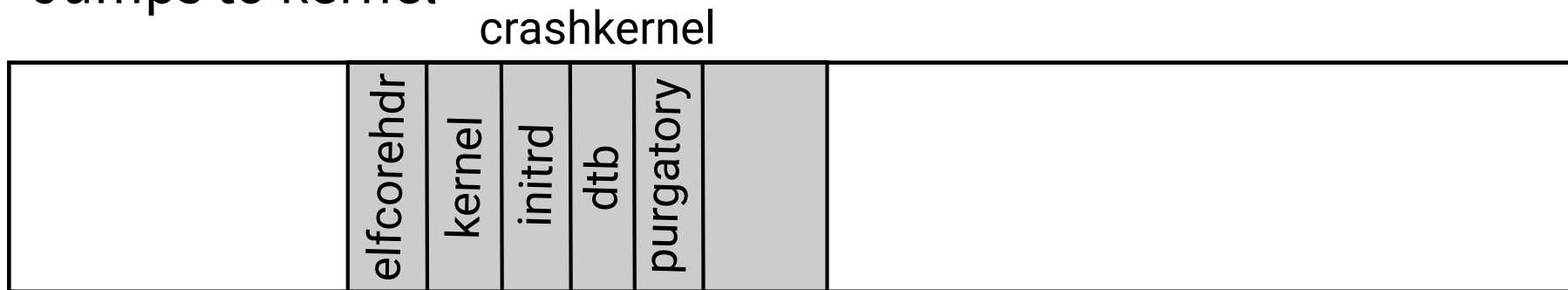
Device tree

- Created from /sys/firmware/fdt (even on ACPI only)
- Updated with information about
 - initrd, elfcorehdr, usable-memory-range



Purgatory

- He decides over heaven and hell
- Checks SHA265 of all segments but itself
- Loads kernel and device tree into registers
- Jumps to kernel



Purgatory - arm64

```
.globl purgatory_start
purgatory_start:

    adr    x19, .Lstack
    mov    sp, x19

    bl     purgatory

    /* Start new image. */
    ldr    x17, arm64_kernel_entry
    ldr    x0, arm64_dtb_addr
    mov    x1, xzr
    mov    x2, xzr
    mov    x3, xzr
    br     x17

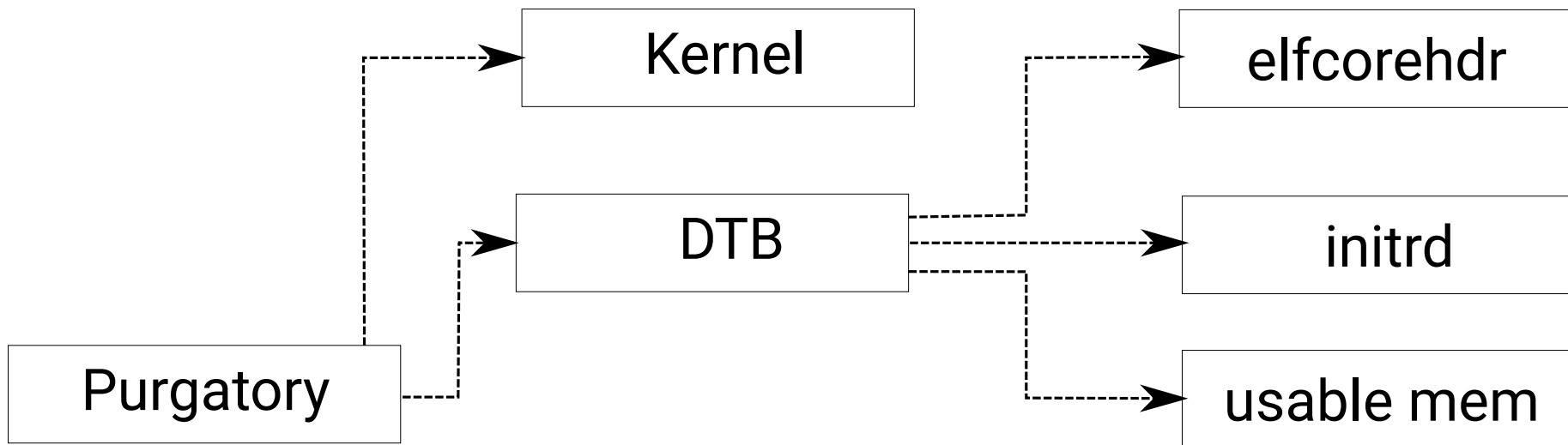
.data

.globl arm64_kernel_entry

.globl arm64_dtb_addr

.end
```


Overview - arm64



kexec-tools

- kexec_load and kexec_file_load
- In kexec_load case information passed to the kernel
 - Purgatory entry point
 - Number and address of the segments

```
struct kexec_segment {  
    const void *buf;  
    size_t bufsz;  
    const void *mem;  
    size_t memsz;  
};
```



The background features abstract geometric shapes in two shades of green. On the left, a large teal shape with a hexagonal-like form contains the text. To its right, a darker green shape also has a hexagonal-like form. These shapes are separated by white space, creating a modern, geometric aesthetic.

Kernel part...
under the hood

Kernel internals

- Production kernel prepares capture kernel
 - `kexec_load` syscall
- Production kernel crashes
- Capture kernel boots up



Loading capture system

- Check we are root, flags and segment number
- Create kimage which holds
 - kexec_segments info from userspace
 - Purgatory entry point (image->start)
 - Memory for control page, allocated from reserved memory
 - Memory for data copy of vmcoreinfo



Checks (no one told you about...)

- Check sanity of segments
 - No overlap, page aligned, are in reserved memory area
 - `segment.memsz >= segment.bufsz`
- But also:
 - `nr. pages of all segments.mem <= totalram_pages/2`

```
struct kexec_segment {  
    const void *buf;  
    size_t bufsz;  
    const void *mem;  
    size_t memsz;  
};
```

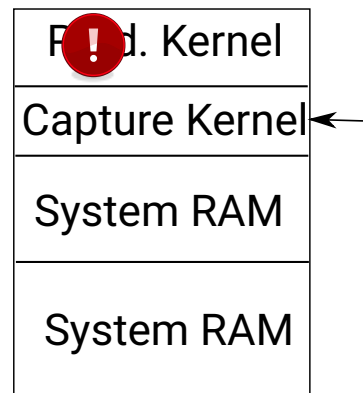
Loading capture system

- copy_from_user:
 - segment.buf to segment.mem (= reserved memory area)
- Protect segment.mem pages
 - Clear PTE_VALID bit for segment pages

```
struct kexec_segment {  
    const void *buf;  
    size_t bufsz;  
    const void *mem;  
    size_t memsz;  
};
```

Kernel crashes

- Disable local IRQs, save CPU registers
- Write time of crash to (restored) vmcoreinfo
- Stop all other CPUs (IPI_CPU_CRASH_STOP)
 - Save CPU registers (crash_notes), disable local IRQs
 - Call PSCI cpu_die
- Check if all CPUs down
- Copy relocation code to control page



Kernel crashes

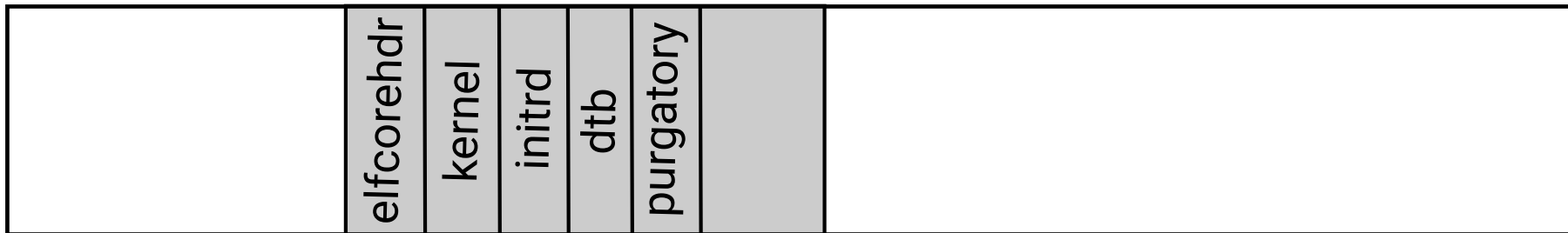
- Shutdown MMU, disable caches
- `arm64_reloacte_new_kernel`
 - Check if relocation needed
 - Jumps to purgatory (directly or through EL2)



Capture kernel boot

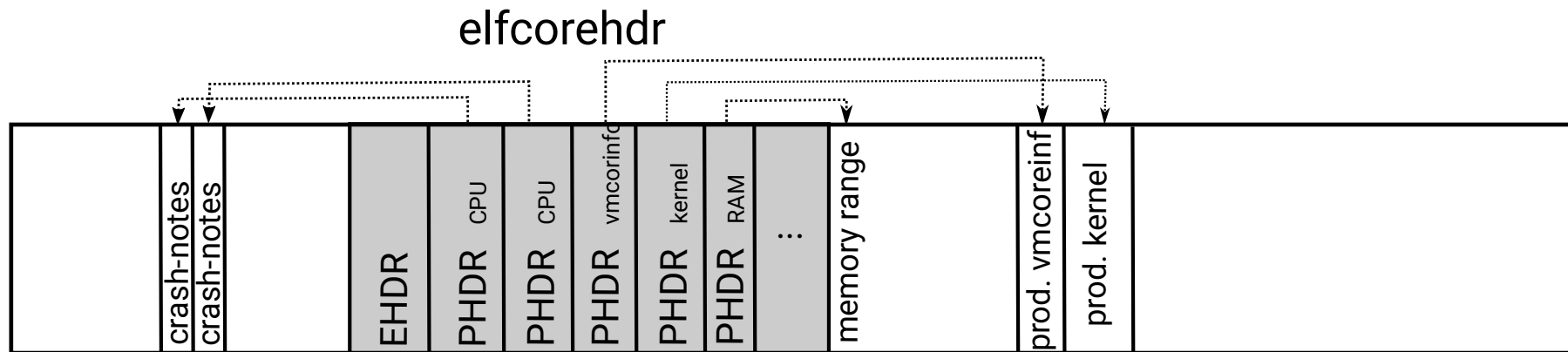
- Special device tree includes
 - linux,elfcorehdr
 - linux,usable-memory-range
 - linux,initrd-start, linux,initrd-end

crashkernel



Capture kernel boot

- Reserves memory and copys content from elfcorehdr into elfcorehdr_buf (from capture kernel)
- When reading /proc/vmcore copy production kernel memory



Distribution parts

Distribution parts

- Set up can be difficult
 - Reserved memory needed depends on system RAM + initrd size
 - Capture initrd should not be too big
 - ...but should have all the tools
 - Automatic storage of dump
 - Want to reboot to production system after crash?



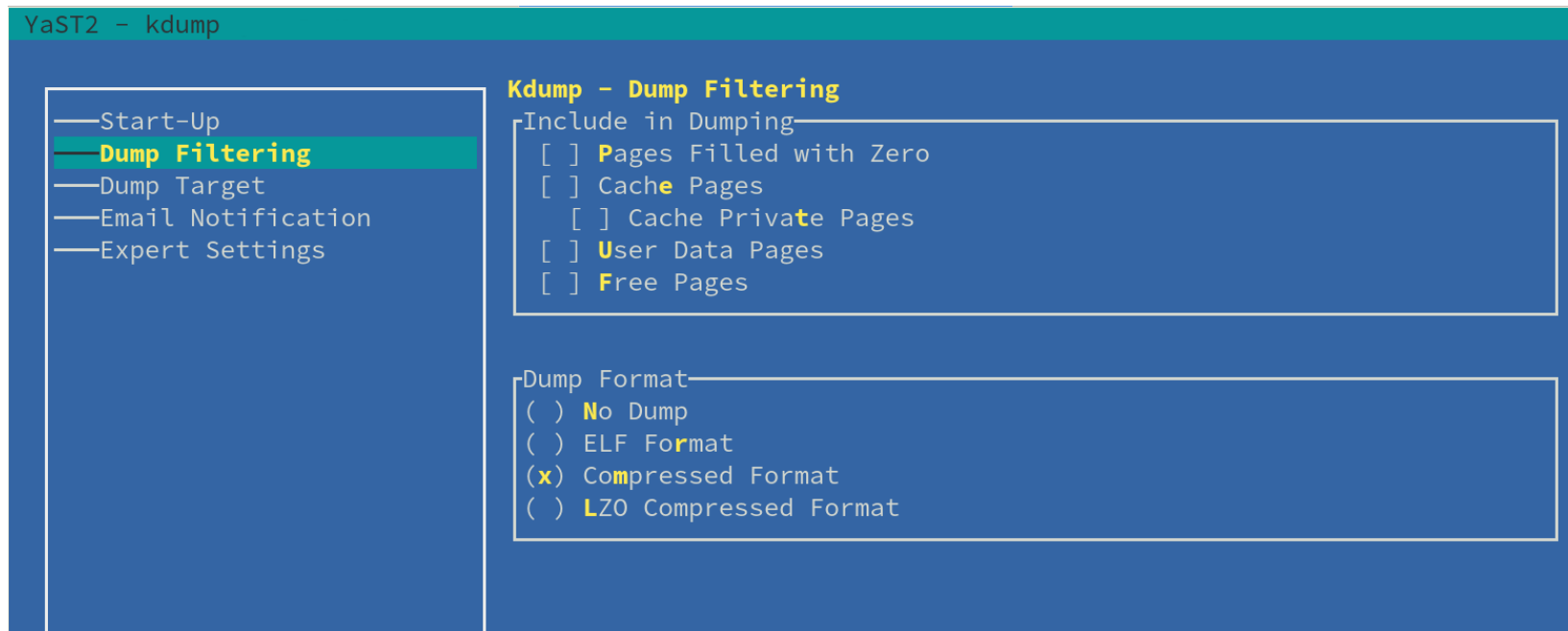
Distribution parts

- SUSE Kdump – swissarmy knife for setting up kdump
 - Production system
 - Dracut scripts to create initrd
 - Bash scripts to load capture system
 - Tool to approximate size of reserved memory
 - Capture system
 - Configuration of dump creation
 - Where the dump gets stored



Distribution parts

- yast2 kdump is your friend!



The background features abstract geometric shapes in two shades of green. A large teal shape occupies the left and top portions, while a bright green shape is on the right. They are separated by a white diagonal line.

Quick demo

References

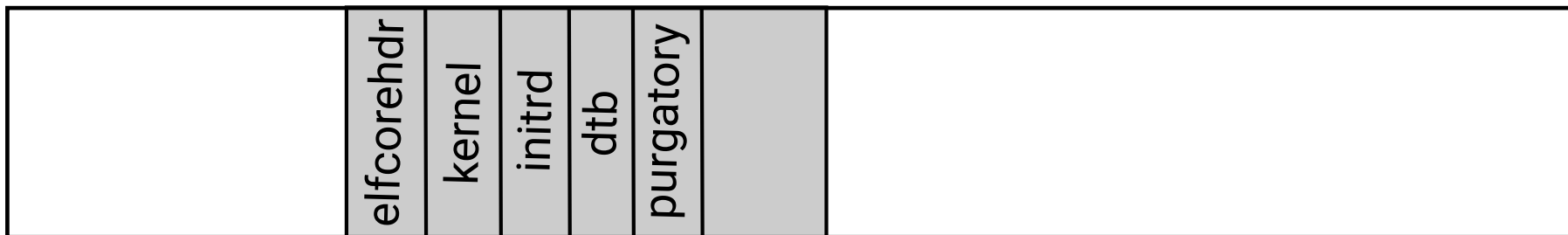
- kexec-tools source code
 - <https://git.kernel.org/pub/scm/utils/kernel/kexec/kexec-tools.git/>
- SUSE documentation
 - <https://doc.opensuse.org/documentation/leap/tuning/html/book.sle.tuning/cha.tuning.kexec.html>
- openSUSE Kdump
 - <https://github.com/openSUSE/kdump/>
- Blog explaining kexec/kdump
 - <https://opensource.com/article/17/6/kdump-usage-and-internals>



Take aways

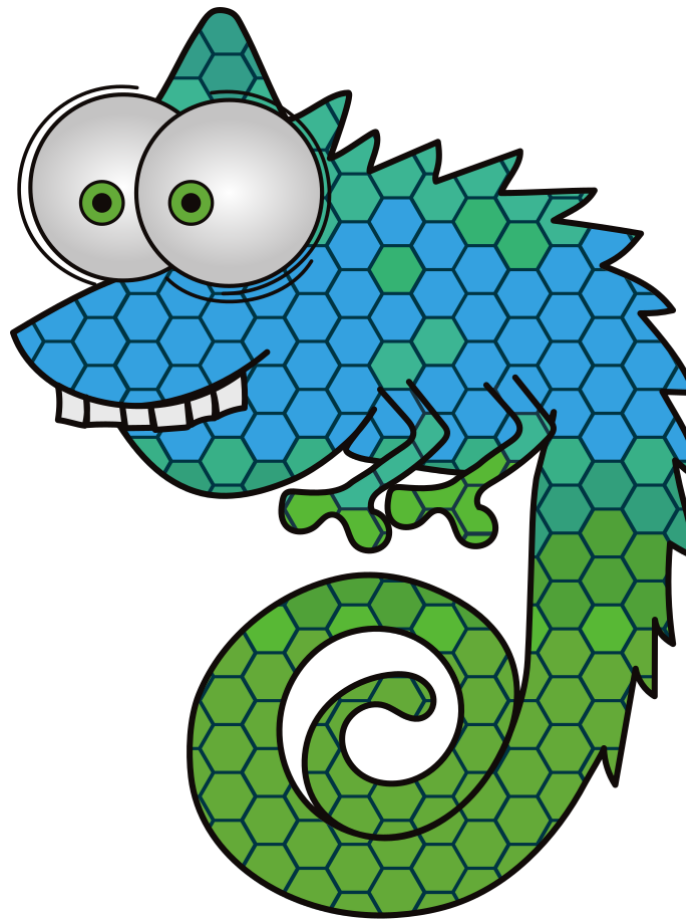
- Production system has reserved memory area
- Capture system gets saved in this area
- Segment elfcoreheader points to the different physical memory location of the production system
- Capture system uses this information to create a dump

crashkernel





Questions?!



Join Us at www.opensuse.org



License

This slide deck is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license. It can be shared and adapted for any purpose (even commercially) as long as Attribution is given and any derivative work is distributed under the same license.

Details can be found at <https://creativecommons.org/licenses/by-sa/4.0/>

General Disclaimer

This document is not to be construed as a promise by any participating organisation to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. openSUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for openSUSE products remains at the sole discretion of openSUSE. Further, openSUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All openSUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC, in the United States and other countries. All third-party trademarks are the property of their respective owners.

Credits

Template

Richard Brown
rbrown@opensuse.org

Design & Inspiration

openSUSE Design Team
<http://opensuse.github.io/branding-guidelines/>