OPEN SOURCE FIRMWARE CONFERENCE 2020 DECEMBER 01 – 03

kexec based bootloaders/fast rebooting: Boon or Bane

SPEAKER

Bhupesh Sharma

Red Hat

<bhsharma@redhat.com>

$ whoami

- Part of Red Hat kernel team.
- Been hacking on bootloaders and kernel since past 14 years.
- Contribute to:
  - Linux,
  - EFI/u-boot bootloader, and
  - User-space utilities like:
    - kexec-tools, and
    - makedumpfile.
- Co-maintain crash-utility tool

Outline

- About kexec - What?
- About kexec - How?
- Linux booting Linux - kexec based bootloaders
- So everything works fine, or does it ..
- Pain Points
- Suggestions

About kexec - What?

- kexec enables you to load and boot into **another kernel** from the currently running kernel.

- **Standard** system boot v/s **kexec** boot:
  - **kexec** boot - skips hardware initialization performed by BIOS / firmware.

- So, overall **kexec** reboot time reduces 

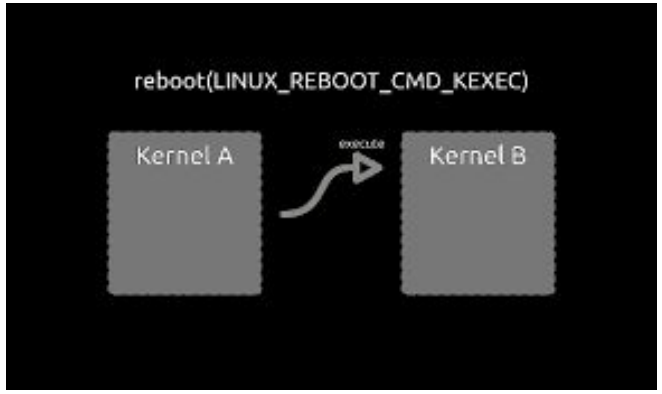
- Related **syscalls**
  - `kexec_load()`
  - `kexec_file_load()`

About kexec - What?

- Related kernel **CONFIG** options
  - `CONFIG_KEXEC`
  - `CONFIG_KEXEC_FILE`

- Supported **architectures**
  - `x86_64,ppc64/ppc64le,s390/s390x,arm,arm64`
    - `RISC-V: Work in progress`

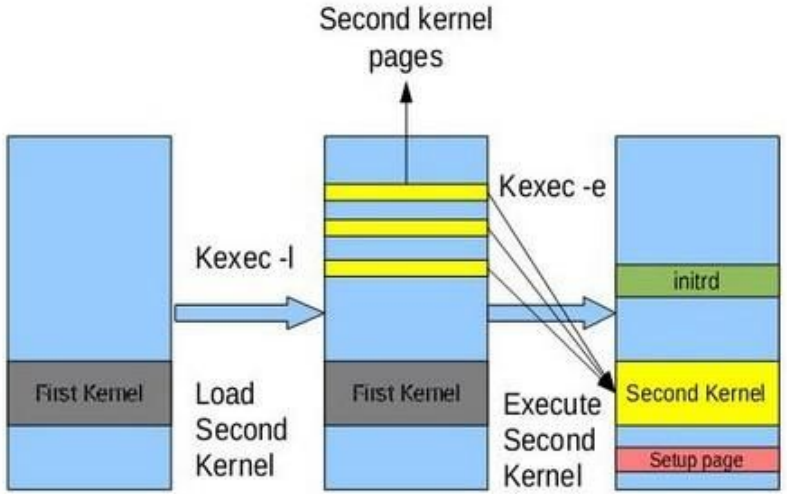
About kexec - How?



- **2-Step** process
  - **Kernel** space support
    - `kexec_load()` and `kexec_file_load()` syscall(s) loads a new kernel into memory.
    - `reboot(LINUX_REBOOT_CMD_KEXEC)` syscall reboots into the new kernel.
  - **User** space support

    `/usr/bin/kexec` – provided by **kexec-tools** package

Kexec design



About kexec - How?

from the context of running kernel

- **2-Step** process
  - **Load** a new kernel into the physical memory:

```
# kexec -l <kernel-image>  --initrd=<initramfs-image> --reuse-cmdline
```

  - **Boot** into the new kernel:

```
# kexec -e
```

  - **Unload** the loaded kernel *(if need be)*:

```
# kexec -u
```

Linux booting Linux - kexec based bootloaders

- Several `kexec` based open-source bootloaders are available:
  - LinuxBoot
  - Petitboot
  - kexecboot
  - kboot, several more ...

So everything works fine, or does it …

On Amazon EC2 Nitro plat ×

access.redhat.com/solutions/4466991

Red Hat Customer Portal | Products & Services | Tools | Security | Community

Products & Services › Knowledgebase › On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using "kexec -e " and "systemctl kexec" c...

On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using "kexec -e " and "systemctl kexec" causes the kernel to crash.

SOLUTION IN PROGRESS - Updated June 22 2020 at 8:29 AM - English

Issue

- On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using `kexec -e` and `systemctl kexec` causes the kernel to crash.
- The same kernel panic is appearing with `kexec -p` and `echo c > /proc/sysrq-trigger` as well.

Raw

```
[    3.327901] sched: Unexpected reschedule of offline CPU#1!
[    3.334363] WARNING: CPU: 0 PID: 1 at arch/x86/kernel/smp.c:128 native_smp_send_reschedule+0x34/0x40
[    3.346699] Modules linked in:
[    3.352238] CPU: 0 PID: 1 Comm: init Not tainted 4.18.0-80.1.2.el8_0.x86_64 #1
[    3.363631] Hardware name: Amazon EC2 t3.large/, BIOS 1.0 10/16/2017
[    3.370639] RIP: 0010:native_smp_send_reschedule+0x34/0x40
[    3.377330] Code: 05 21 90 3b 01 73 15 48 8b 05 78 af 10 01 be fd 00 00 00 48 8b 40 30 e9 9a 94 bb 00 89 fe 48 c7 c7 a8 68 e8
9e e8 b6 28 06 00 &lt;0f&gt; 0b c3 66 0f 1f 84 00 00 00 00 00 0f 1f 44 00 00 53 48 83 ec 20
[    3.398020] RSP: 0018:ffff98e9f9403e48 EFLAGS: 00010086
[    3.404505] RAX: 0000000000000000 RBX: ffff98e9f9523080 RCX: ffffffff9f059d28
[    3.411891] RDX: 0000000000000001 RSI: 0000000000000096 RDI: 0000000000000046
```

OOPS!

Bug #1869948 "Multiple K ×

bugs.launchpad.net/ubuntu/+source/linux/+bug/1869948

Ubuntu
linux package

Overview | Code | Bugs | Blueprints | Translations | Answers

Multiple Kexec in AWS Nitro instances fail

Bug #1869948 reported by Guilherme G. Piccoli on 2020-03-31

This bug affects 1 person

Affects	Status	Importance	Assigned to
linux (Ubuntu)	Fix Released	High	Guilherme G. Piccoli
Xenial	Fix Released	High	Guilherme G. Piccoli
Bionic	Fix Released	High	Guilherme G. Piccoli
Eoan	Fix Released	High	Guilherme G. Piccoli
Focal	Fix Released	High	Guilherme G. Piccoli

Also affects project | Also affects distribution/package | Nominate for series

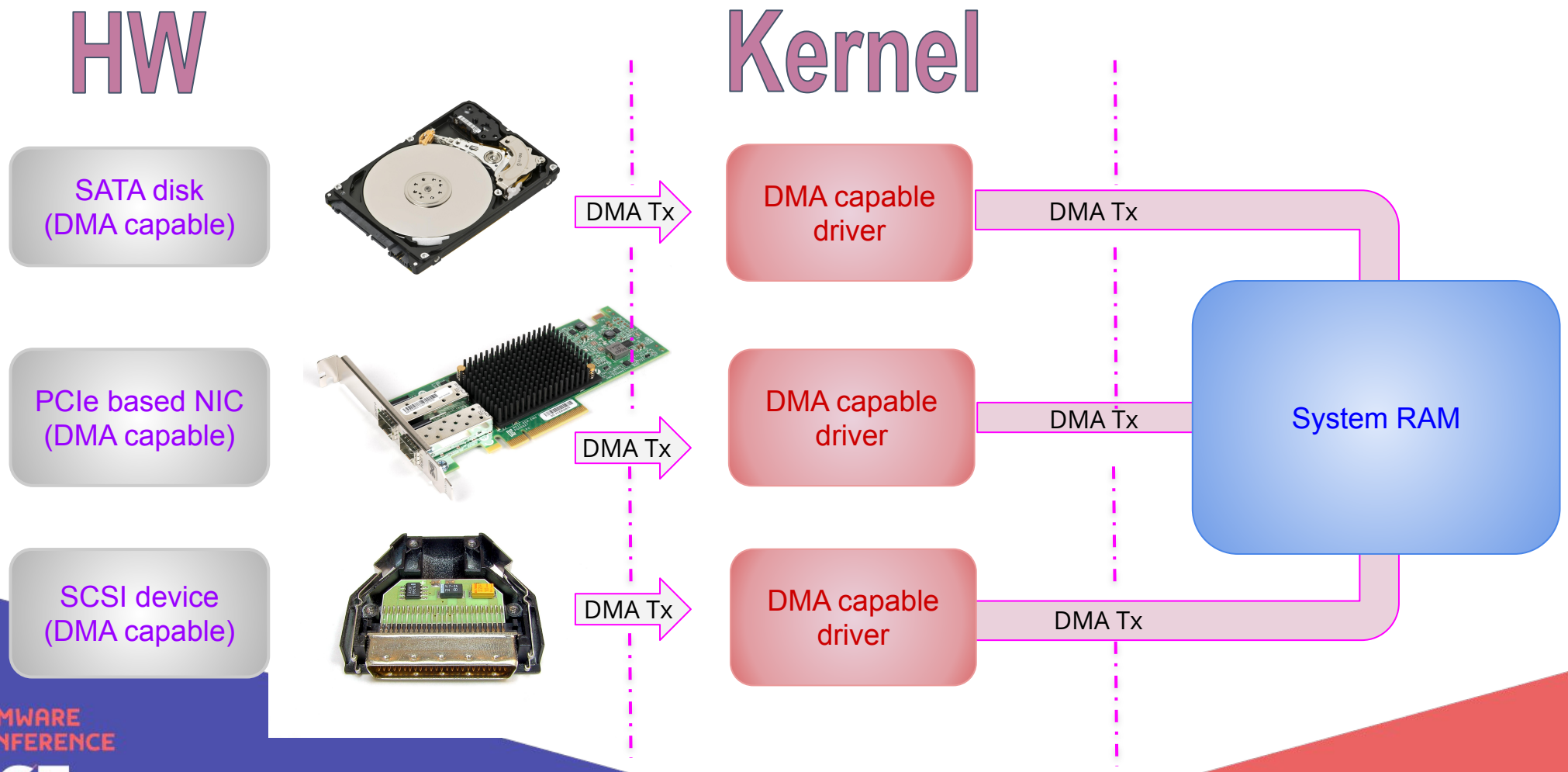
Bug Description

```
[Impact]
* Currently, users cannot perform multiple kernel kexec loads on AWS Nitro
instances (KVM-based); after the 2nd or 3rd kexec, an initrd corruption is
observed, with the following signature:

 Initramfs unpacking failed: junk within compressed archive
[...]
 Kernel panic - not syncing: No working init found.
Try passing init= option to kernel. See Linux Documentation/admin-guide/
init.rst for guidance.
CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.5.0-rc7-gpiccoli+ #26 Hardware
name: Amazon EC2 t3.large/, BIOS 1.0 10/16/2017
Call Trace:
  dump_stack+0x6d/0x9a
```

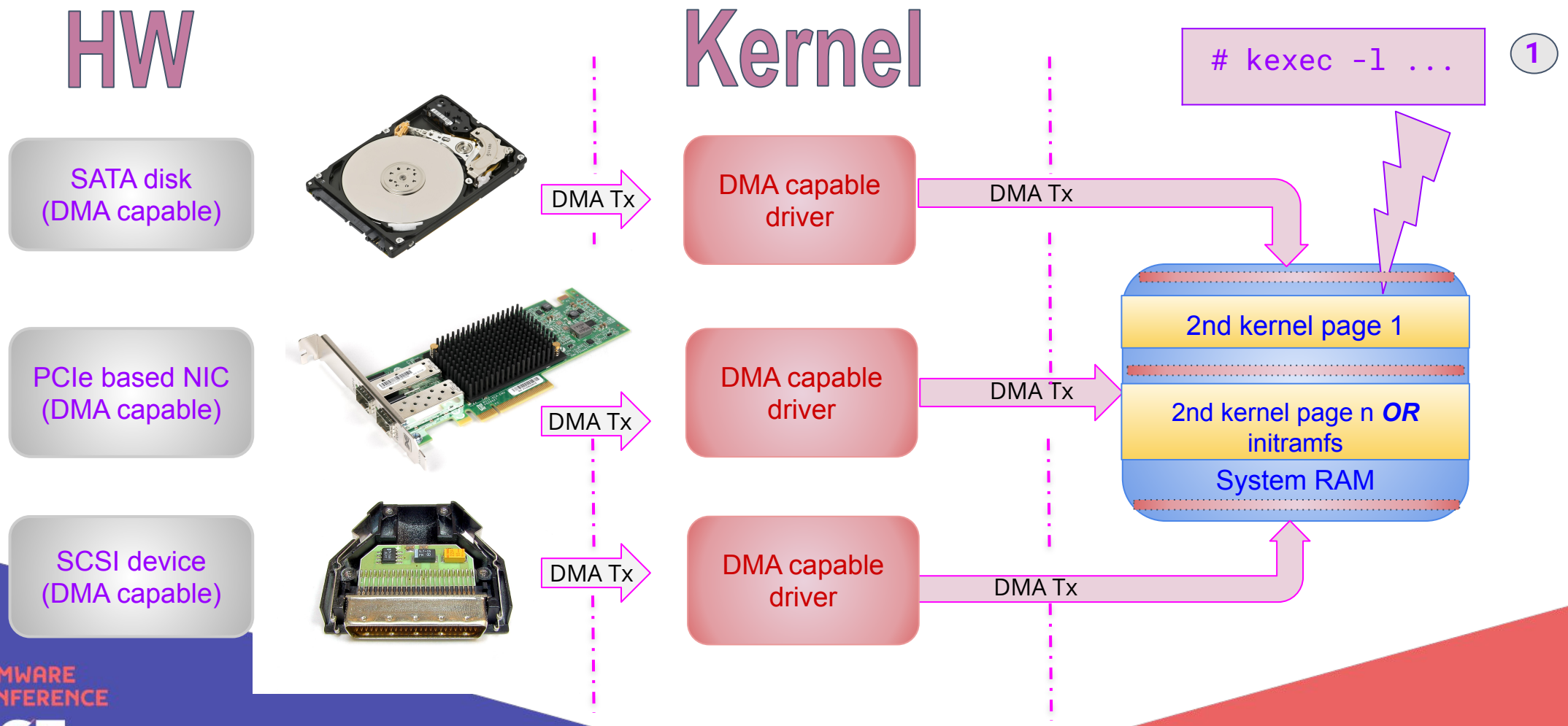
https://launchpad.net/ubuntu

OPEN SOURCE FIRMWARE CONFERENCE

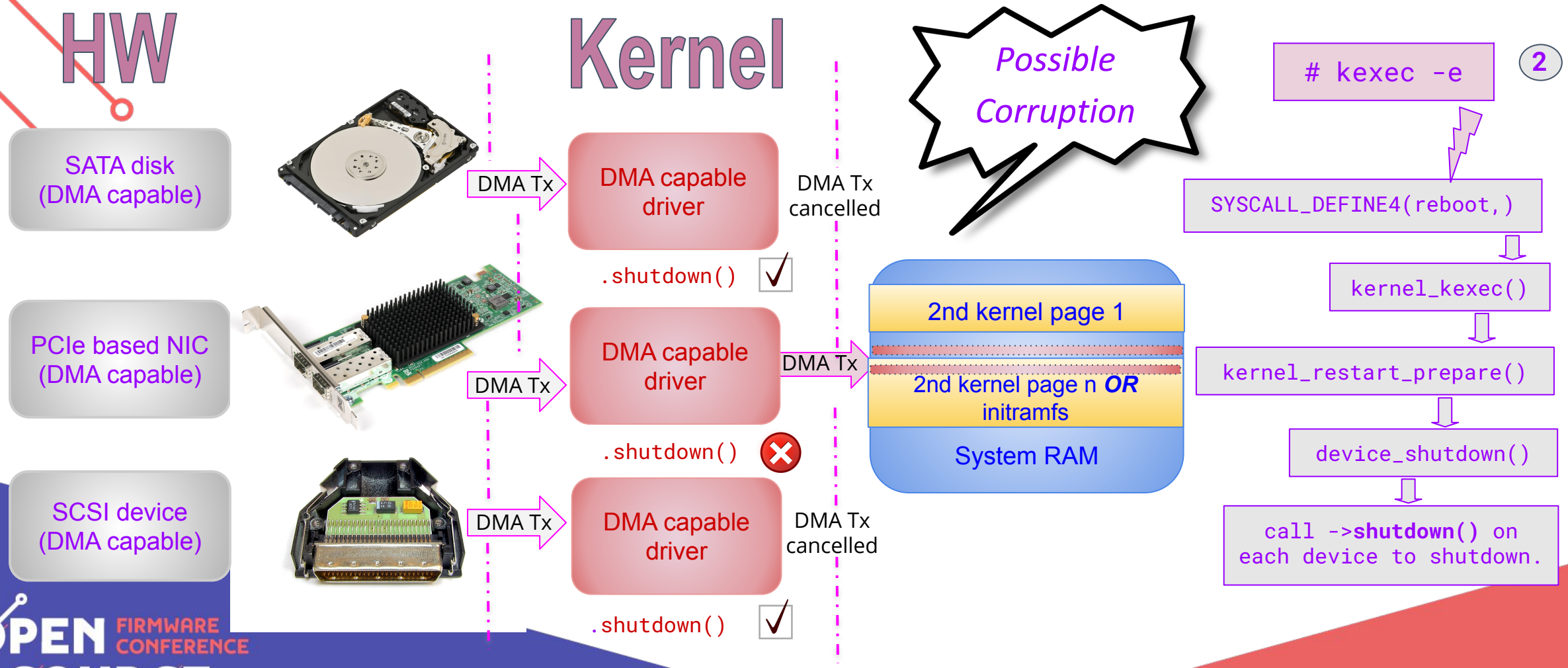
So everything works fine, or does it …



So everything works fine, or does it …



So everything works fine, or does it …



So everything **does not work** as intended

On Amazon EC2 Nitro plat ×

access.redhat.com/solutions/4466991

Red Hat Customer Portal | Products & Services | Tools | Security | Community

Products & Services › Knowledgebase › On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using "kexec -e " and "systemctl kexec" c...

kexec'ed kernel fails to boot

OOPS!

On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using "kexec -e " and "systemctl kexec" causes the kernel to crash.

SOLUTION IN PROGRESS - Updated June 22 2020 at 8:29 AM - English

Issue

- On Amazon EC2 Nitro platform running RHEL 8, the Fast boot using `kexec -e` and `systemctl kexec` causes the kernel to crash.
- The same kernel panic is appearing with `kexec -p` and `echo c > /proc/sysrq-trigger` as well.

Raw

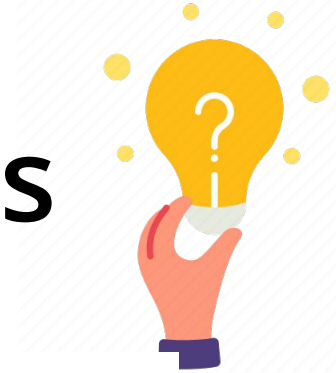
```
[    3.327901] sched: Unexpected reschedule of offline CPU#1!
[    3.334363] WARNING: CPU: 0 PID: 1 at arch/x86/kernel/smp.c:128 native_smp_send_reschedule+0x34/0x40
[    3.346699] Modules linked in:
[    3.352238] CPU: 0 PID: 1 Comm: init Not tainted 4.18.0-80.1.2.el8_0.x86_64 #1
[    3.363631] Hardware name: Amazon EC2 t3.large/, BIOS 1.0 10/16/2017
[    3.370639] RIP: 0010:native_smp_send_reschedule+0x34/0x40
[    3.377330] Code: 05 21 90 3b 01 73 15 48 8b 05 78 af 10 01 be fd 00 00 00 48 8b 40 30 e9 9a 94 bb 00 89 fe 48 c7 c7 a8 68 e8
9e e8 b6 28 06 00 &lt;0f&gt; 0b c3 66 0f 1f 84 00 00 00 00 00 0f 1f 44 00 00 53 48 83 ec 20
[    3.398020] RSP: 0018:ffff98e9f9403e48 EFLAGS: 00010086
[    3.404505] RAX: 0000000000000000 RBX: ffff98e9f9523080 RCX: ffffffff9f059d28
[    3.411891] RDX: 0000000000000001 RSI: 0000000000000096 RDI: 0000000000000046
```

OPEN SOURCE FIRMWARE CONFERENCE

Pain Points

- Linux based bootloaders use `kexec` **under the hood**.
- This means:
  - `kexec` is **not expected** to fail.
  - In case of a failure
    - machine is *no longer* boot'able.
    - several *painful* debug cycles.
    - most of such failures are *random* in nature.
- Common **cause** for `kexec` reboot failure:
  - **missing** `shutdown()` callback in driver code.

Suggestions



Add `shutdown()` callbacks in your driver code

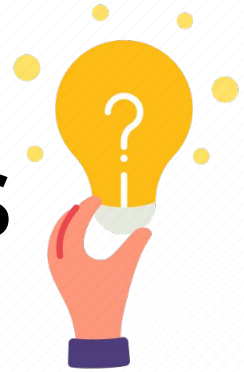
[PATCH] ata: ahci: Add shu ×

spinics.net/lists/kexec/msg24305.html

```
device_shutdown() called from reboot/power_shutdown expect all
devices to be shutdown. Same is true for ahci pci driver.
As no shutdown function was implemented ata subsystem remains
always alive and DMA/interrupt still active.

It creates problem during kexec, here "M" bit is cleared to stop
DMA usage. Any further DMA transaction may cause instability and
the hard-disk may even not get detected for second kernel.
One of possible case is periodic file system sync.

So defining ahci pci driver shutdown to freeze hardware (mask
interrupt, stop DMA engine and free DMA resources).

Signed-off-by: Prabhakar Kushwaha <pkushwaha@xxxxxxxxxxx>
---
 drivers/ata/ahci.c        |  8 ++++++++
 drivers/ata/libata-core.c | 21 +++++++++++++++++++++
 include/linux/libata.h    |  1 +
 3 files changed, 30 insertions(+)

diff --git a/drivers/ata/ahci.c b/drivers/ata/ahci.c
index 4bfd1b14b390..31fc934740b6 100644
--- a/drivers/ata/ahci.c
+++ b/drivers/ata/ahci.c
@@ -81,6 +81,7 @@ enum board_ids {

 static int ahci_init_one(struct pci_dev *pdev, const struct pci_device_id *ent);
 static void ahci_remove_one(struct pci_dev *dev);
+static void ahci_shutdown_one(struct pci_dev *dev);
 static int ahci_vt8251_hardreset(struct ata_link *link, unsigned int *class,
                                  unsigned long deadline);
 static int ahci_avn_hardreset(struct ata_link *link, unsigned int *class,
@@ -606,6 +607,7 @@ static struct pci_driver ahci_pci_driver = {
        .id_table               = ahci_pci_tbl,
        .probe                  = ahci_init_one,
        .remove                 = ahci_remove_one,
+       .shutdown               = ahci_shutdown_one,
        .driver = {
                .pm             = &ahci_pci_pm_ops,
        },
@@ -626,6 +628,7 @@ MODULE_PARM_DESC(mobile_lpm_policy, "Default LPM policy for mobile chipsets");
 static void ahci_pci_save_initial_config(struct pci_dev *pdev,
                                          struct ahci_host_priv *hpriv)
 {
```

[PATCH] ata: ahci: Add shutdown to freeze hardware resources of ahci

Suggestions



Add `shutdown()` callbacks in your driver code

[PATCH] scsi: lpfc: Add sh ×

spinics.net/lists/linux-scsi/msg105102.html

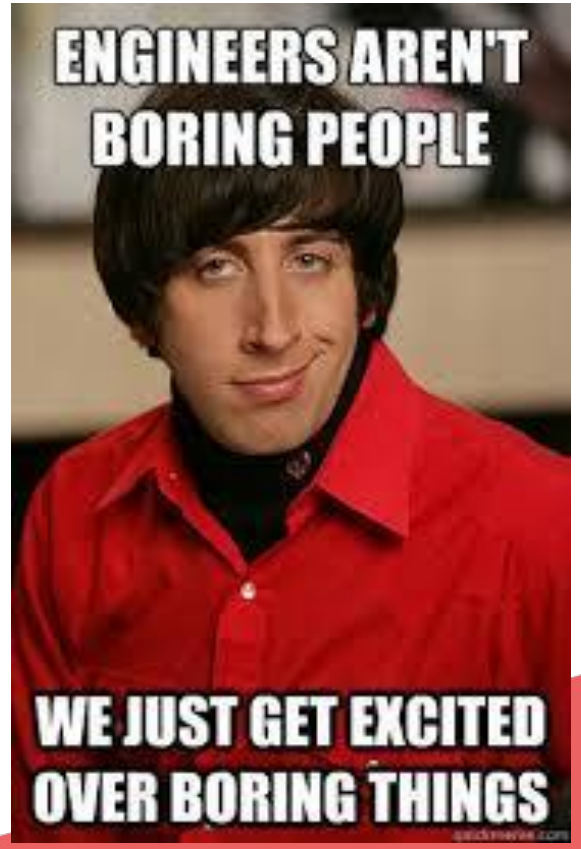
[PATCH] scsi: lpfc: Add shutdown method for kexec

From: Anton Blanchard <anton@xxxxxxxxx>

We see lpfc devices regularly fail during kexec. Fix this by adding a shutdown method which mirrors the remove method.

Next Steps

- Report `kexec` **reboot** failures @ `kexec@lists.infradead.org`
- `kexec` failures can be related to missing `shutdown()` callbacks in *DMA capable drivers*, e.g.
  - SATA, USB, NIC, PCIe driver
- Add more *debugging capabilities* to your `kexec` based bootloader.
  - console logs ⇨ pretty useful.



Questions

OPEN SOURCE FIRMWARE CONFERENCE

Thank you

OPEN SOURCE FIRMWARE CONFERENCE