XDC 2021

Improvements to the VKMS Driver

Sumera Priyadarsini

Mentors:
Melissa Wen and Daniel Vetter

September 15, 2021

About me

- Junior Software Engineer, Polar Signals
- Interests: *Low level systems, kernel development, programming languages*
- Linux Kernel Contributor
- Outreachy Dec’20 Intern for Linux kernel - dri-devel subsystem
- Mentors: Daniel Vetter, Melissa Wen

What is the VKMS driver?

- VKMS driver: *Virtual Kernel Mode Setting driver*
- Software only model of the KMS driver
- Can enable a virtual display without hardware support
- Useful for running tests and X on headless machines, like CI
- Involved two code areas:
  - **VKMS driver**: drivers/gpu/drm/vkms
  - **IGT test suite**: test suite for DRM driver

Internship: Goals

- Configfs interface: Expose VKMS through configfs
  - Enables creating/managing/destroying kernel objects from user space
  - Useful for testing multiple instances of VKMS simultaneously

- Virtual hardware/vblank-less mode
  - VKMS mimics actual hardware using vertical blanking(vblank) interrupts
  - Introduce a mode where it can also emulate virtual hardware, i.e, a vblank-less mode

Internship: Revised Goals

- Emulate Virtual Hardware
- Implement vblankless mode
- Preserve tests except those which depend on vblank interrupts

Implementation

```
> ---
>  drivers/gpu/drm/vkms/vkms_composer.c | 88 ++++++++++++++++++-----------
>  drivers/gpu/drm/vkms/vkms_drv.h      |  3 +
>  2 files changed, 58 insertions(+), 33 deletions(-)
>
> diff --git a/drivers/gpu/drm/vkms/vkms_composer.c b/drivers/gpu/drm/vkms/vkms_composer.c
> index 66c6842d70db..0d2bad3ff849 100644
> --- a/drivers/gpu/drm/vkms/vkms_composer.c
> +++ b/drivers/gpu/drm/vkms/vkms_composer.c
> @@ -169,6 +169,44 @@ static int compose_planes(void **vaddr_out,
>   return 0;
>  }
>
> +int vkms_composer_common(struct vkms_crtc_state *crtc_state,
> +          struct vkms_output *out, bool wb_pending, uint32_t *crc32)
> +{
> + struct vkms_composer *primary_composer = NULL;
> + struct vkms_composer *cursor_composer = NULL;
> + void *vaddr_out = NULL;
> + int ret;
> +
> + if (crtc_state->num_active_planes >= 1)
> +     primary_composer = crtc_state->active_planes[0]->composer;
> + if (crtc_state->num_active_planes == 2)
> +     cursor_composer = crtc_state->active_planes[1]->composer;
> + if (!primary_composer)
> +     return -EINVAL;
> + if (wb_pending)
> +     vaddr_out = crtc_state->active_writeback;
> +
> + ret = compose_planes(&vaddr_out, primary_composer, cursor_composer);
> + if (ret) {
> +     if (ret == -EINVAL && !wb_pending)
> +         kfree(vaddr_out);
> +     return -EINVAL;
> + }
> +
> + *crc32 = compute_crc(vaddr_out, primary_composer);
> +
> + if (wb_pending) {
> +     drm_writeback_signal_completion(&out->wb_connector, 0);
> +     spin_lock_irq(&out->composer_lock);
> +     crtc_state->wb_pending = false;
> +     spin_unlock_irq(&out->composer_lock);
> + } else {
> +     kfree(vaddr_out);
> + }
> +
> + return 0;
> +}
> +
```

- Add a helper function *vkms_composer_common()*
- Introduce *vkms_crtc_composer()*

```
> @@ -247,6 +255,20 @@  void vkms_composer_worker(struct work_struct *work)
>       drm_crtc_add_crc_entry(crtc, true, frame_start++, &crc32);
>  }
>
> +void vkms_crtc_composer(struct vkms_crtc_state *crtc_state)
> +{
> + struct drm_crtc *crtc = crtc_state->base.crtc;
> + struct vkms_output *out = drm_crtc_to_vkms_output(crtc);
> + u32 crc32 = 0;
> + int ret;
> +
> + ret = vkms_composer_common(crtc_state, out, crtc_state->wb_pending, &crc32);
> + if (ret == -EINVAL)
> +     return;
> +
> + drm_crtc_add_crc_entry(crtc, true, 0, &crc32);
> +}
> +
```

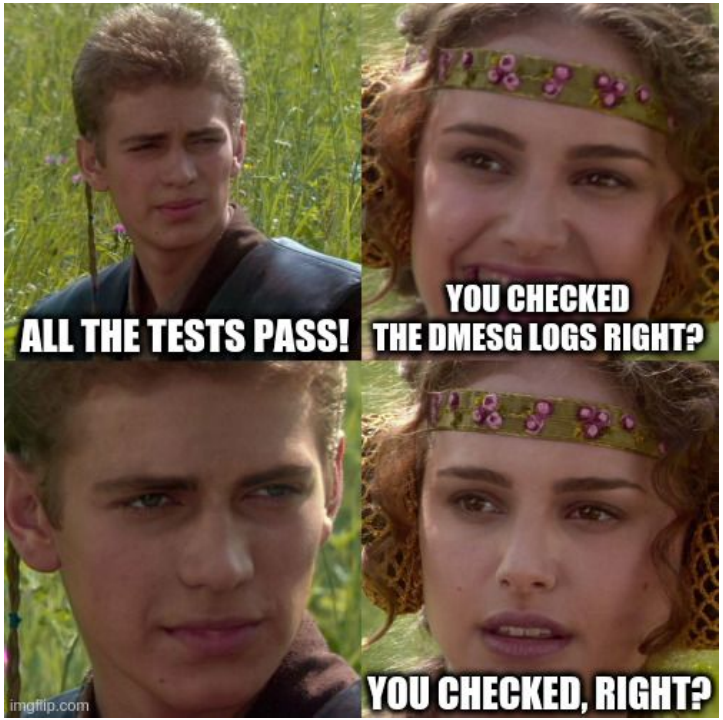
Implementation

- Add atomic helpers for virtual_hw mode in *vkms_virtual_crtc_helper_funcs*

```
>  static void vkms_release(struct drm_device *dev)
> @@ -159,12 +164,14 @@ static int vkms_create(struct vkms_config *config)
>       goto out_devres;
>    }
>
>
> - ret = drm_vblank_init(&vkms_device->drm, 1);
> - if (ret) {
> -     DRM_ERROR("Failed to vblank\n");
> -     goto out_devres;
> + if (!vkms_device->config->virtual_hw) {
> +     ret = drm_vblank_init(&vkms_device->drm, 1);
> +     if (ret) {
> +         DRM_ERROR("Failed to vblank\n");
> +         goto out_devres;
> +     }
>    }
```

```
> +
> +static const struct drm_crtc_helper_funcs vkms_virtual_crtc_helper_funcs = {
> + .atomic_check   = vkms_crtc_atomic_check,
> + .atomic_flush   = vkms_virtual_crtc_atomic_flush,
>  };
>
>  int vkms_crtc_init(struct drm_device *dev, struct drm_crtc *crtc,
>          struct drm_plane *primary, struct drm_plane *cursor)
>  {
>   struct vkms_output *vkms_out = drm_crtc_to_vkms_output(crtc);
> + struct vkms_device *vkmsdev = drm_device_to_vkms_device(dev);
>   int ret;
>
>   ret = drm_crtc_init_with_planes(dev, crtc, primary, cursor,
> @@ -289,7 +309,10 @@ int vkms_crtc_init(struct drm_device *dev, struct drm_crtc *crtc,
>       return ret;
>   }
>
> - drm_crtc_helper_add(crtc, &vkms_crtc_helper_funcs);
> + if (vkmsdev->config->virtual_hw)
> +     drm_crtc_helper_add(crtc, &vkms_virtual_crtc_helper_funcs);
> + else
> +     drm_crtc_helper_add(crtc, &vkms_vblank_crtc_helper_funcs);
>
>   spin_lock_init(&vkms_out->lock);
>   spin_lock_init(&vkms_out->composer_lock);
> diff --git a/drivers/gpu/drm/vkms/vkms_drv.c b/drivers/gpu/drm/vkms/vkms_drv.c
> index 2173b82606f6..945c4495d62a 100644
> --- a/drivers/gpu/drm/vkms/vkms_drv.c
> +++ b/drivers/gpu/drm/vkms/vkms_drv.c
> @@ -44,6 +44,11
> +static bool enable_virtual_hw = false;
> +module_param_named(enable_virtual_hw, enable_virtual_hw, bool, 0444);
> +MODULE_PARM_DESC(enable_virtual_hw, "Enable/Disable virtual hardware mode(virtual \
> +hardware mode disables vblank interrupts)");
> +
```

Implementation



dmesg logs for *kms_flip.c*

1. *Memory allocation errors*
2. *Kernel panic*

Implementation

```
[drm:vkms_composer_common.cold [vkms]] *ERROR* Cannot allocate memory for output frame.
kzalloc size is 6291456
[drm:vkms_composer_common.cold [vkms]] *ERROR* Cannot allocate memory for output frame.
kzalloc size is 6291456
[drm:vkms_composer_common.cold [vkms]] *ERROR* Cannot allocate memory for output frame.
kzalloc size is 6291456
[drm:vkms_composer_common.cold [vkms]] *ERROR* Cannot allocate memory for output frame.
```

Problem allocating for two frames during *kms_flip* test, used *kvmalloc* instead of *kzalloc*

Flaky vmap allocation error, handled and fixed in a patch sent by Thomas Zimmermann

```
[  +0.009109] [IGT] kms_flip: starting subtest dpms-off-confusion-interruptible
[  +0.000227] [IGT] kms_flip: starting dynamic subtest A-Virtual2
[  +0.001448] [drm:vkms_prepare_fb [vkms]] *ERROR* vmap failed: -4
[  +0.000096] wb_pending in crtc_composer is 0
[  +0.000002] kzalloc size(outta if) is 3145728
[  +0.000002] kzalloc size(outta if, after) is 3145728
[  +0.000002] kzalloc size is 3145728
[  +0.001648] BUG: unable to handle page fault for address: ffffb54d02b01000
[  +0.000029] #PF: supervisor read access in kernel mode
[  +0.000002] #PF: error_code(0x0000) - not-present page
[  +0.000002] PGD 1000067 P4D 1000067 PUD 11b2067 PMD 3579067 PTE 0
[  +0.000007] Oops: 0000 [#1] SMP PTI
[  +0.000004] CPU: 1 PID: 59 Comm: kworker/u4:1 Tainted: G           OE     5.13.0-rc3sumov
[  +0.000007] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS ArchLinux 1.14.0-
[  +0.000003] Workqueue: events_unbound commit_work [drm_kms_helper]
[  +0.000002] RIP: 0010:memcpy_erms+0x6/0x10
[  +0.000013] Code: c3 cc cc cc cc eb 1e 0f 1f 00 48 89 f8 48 89 d1 48 c1 e9 03 83 e2 07 f3 48
[  +0.000003] RSP: 0018:ffffb54d001e7d38 EFLAGS: 00010286
[  +0.000002] RAX: ffff921193c00000 RBX: ffff921181837400 RCX: 0000000000300000
[  +0.000001] RDX: 0000000000300000 RSI: ffffb54d02b01000 RDI: ffff921193c00000
[  +0.000002] RBP: ffff92118fa4c610 R08: 0000000000000000 R09: 0000000000000006
[  +0.000001] R10: 0000000000000001 R11: 0000000000000000 R12: 0000000000000000
[  +0.000001] R13: ffff92118429bf00 R14: ffff9211915c4e00 R15: ffff921193c00000
[  +0.000002] FS:  0000000000000000(0000) GS:ffff9211efd00000(0000) knlGS:0000000000000000
[  +0.000001] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[  +0.000001] CR2: ffffb54d02b01000 CR3: 0000000036d2e006 CR4: 0000000000370ee0
[  +0.000004] Call Trace:
[  +0.000017]  vkms_composer_common.cold+0x9c/0x46e [vkms]
[  +0.000008]  ? printk+0x68/0x7f
[  +0.000007]  vkms_crtc_composer+0x54/0x5b9 [vkms]
[  +0.000003]  vkms_virtual_crtc_atomic_flush+0x19/0x40 [vkms]
[  +0.000002]  drm_atomic_helper_commit_planes+0x1c4/0x220 [drm_kms_helper]
[  +0.000019]  vkms_atomic_commit_tail+0x28/0xb0 [vkms]
[  +0.000003]  commit_tail+0x94/0x130 [drm_kms_helper]
[  +0.000050]  process_one_work+0x21d/0x3b0
[  +0.000007]  worker_thread+0x50/0x400
[  +0.000002]  ? process_one_work+0x3b0/0x3b0
[  +0.000001]  kthread+0x11b/0x140
[  +0.000006]  ? __kthread_bind_mask+0x60/0x60
[  +0.000002]  ret_from_fork+0x22/0x30
[  +0.000009] Modules linked in: vkms(OE) md4(E) cmac(E) nls_utf8(E) cifs(E) libarc4(E) dns
(E) mousedev(E) bochs_drm(E) drm_vram_helper(E) drm_ttm_helper(E) ttm(E) drm_kms_helper(E)
crc32c_generic(E) crc16(E) mbcache(E) jbd2(E) sr_mod(E) cdrom(E) ata_generic(E) pata_acpi(E
[  +0.000140] CR2: ffffb54d02b01000
[  +0.000003] ---[ end trace 153dcc6bb3f6a5b4 ]---
[  +0.000002] RIP: 0010:memcpy_erms+0x6/0x10
```

Implementation

- **CRC tests** still fail
- CRC: *Cyclic Redundancy Check,* checks data loss in frames
- CRC needs to be implemented as an oneshot operation
- Modify atomic helpers to skip all CRC tests for now
- Tests skip/pass without any errors. Yay!!!

```
> diff --git a/drivers/gpu/drm/vkms/vkms_crtc.c b/drivers/gpu/drm/vkms/vkms_crtc.c
> index 57bbd32e9beb..4a933553e0e4 100644
> --- a/drivers/gpu/drm/vkms/vkms_crtc.c
> +++ b/drivers/gpu/drm/vkms/vkms_crtc.c
> @@ -174,6 +174,15 @@ static const struct drm_crtc_funcs vkms_crtc_funcs = {
>    .verify_crc_source  = vkms_verify_crc_source,
>  };
>
> +static const struct drm_crtc_funcs vkms_vblankless_crtc_funcs = {
> + .set_config             = drm_atomic_helper_set_config,
> + .destroy                = drm_crtc_cleanup,
> + .page_flip              = drm_atomic_helper_page_flip,
> + .reset                  = vkms_atomic_crtc_reset,
> + .atomic_duplicate_state = vkms_atomic_crtc_duplicate_state,
> + .atomic_destroy_state   = vkms_atomic_crtc_destroy_state,
> +};
> +
```

WHAT I LEARNED:

- Writing idiomatic C.
- *Tools*: ftrace, qemu, git
- Checking dmesg logs even if test results are preserved.
- Locks, interrupts, and race conditions
- Using atomic interface to write code
- Writing code that has clear code flow and is easy to test
- Software can be both extremely tough and rewarding.
- Planning ahead, researching potential roadblock.
- Working as part of an inclusive community makes a lot of difference- both in terms of motivation for and quality of work.
- Make sure to ask for help if you are stuck.
- Overcoming imposter syndrome to ask questions on forums

Side quests

- drm/vblank: Fix typo in docs
- drm/vkms: Add information about module options
- drm/vkms: Add support for writeback module
- drm/vkms: Add vkms_config type
- drm/vkms: Add setup and testing information

Future work

- Expose VKMS through configfs
- Implement CRC for virtual hardware/vblankless mode

- I am very grateful to my mentors, Daniel Vetter and Melissa Wen for being so patient and inspiring while teaching me so much.

- Shoutout to Sage Sharp, Anna e só, Vaishali Thakkar, Helen Koike and my fellow interns for making Outreachy such a great experience

Questions?

Thank you for your time!

- **RESOURCES**:
  - Blog: https://whimsicalspren.netlify.app/
  - IRC: sumera
  - Email: sylphrenadin@gmail.com
  - VKMS driver: https://dri.freedesktop.org/docs/drm/gpu/vkms.html
  - IGT test suite: https://gitlab.freedesktop.org/drm/igt-gpu-tools