Packages - Bug #2973

[linux-libre] Critical hangs with nouveau 2D acceleration under 5.10

2021-01-31 05:08 PM - biovoid

Status:	unconfirmed	**% Done:**	0%
Priority:	bug		
Assignee:	Megver83		
Category:			

Description

When 2D acceleration is enabled through nouveau (default), the computer freezes and becomes unresponsive. Occasionally remote access is still possible, but this is not always the case. In most cases, I wasn't doing anything particularly graphically intensive. In one instance, it locked up while I was away, simply streaming audio from the machine.

Adding the kernel parameter `nouveau.noaccel=1` prevents this failure.
Kernel logging on this failure is hit-or-miss... sometimes I get a `kmem_cache_alloc_trace` (as seen in the attached log) and others the logs stop after the line following `general protection fault, probably for non-canonical address`.

I spent some time running through some nouveau tweaks with a developer in #nouveau—they ultimately concluded that it is not a nouveau bug, and that nouveau is simply getting hosed by another problem (see a similar trace for btrfs in the kernel log).

I first encountered this at least a year ago, but at the time simply disabled 2d accel... as such, I am not sure what the first affected version is. Nonetheless, I have yet to run into this using the LTS kernel, thus my belief that this is a kernel issue.

My GPU is a GeForce GTX 670 (Kepler)

Attached are a (truncated; see note for full text) kernel log (with `drm.debug=14 log_buf_len=16M`, per nouveau debug suggestions), Xorg log, and my (minimal) Xorg config.

History

#1 - 2021-01-31 07:48 PM - biovoid

- File kernel.log.tar.gz added

Attaching full kernel log for reliability of access

#2 - 2021-02-01 03:02 AM - bill-auger

- Assignee set to Megver83

#3 - 2021-05-02 09:17 PM - GNUtoo

In the log there is the following kernel version: kernel: Linux version 5.10.6-gnu-1

In Linux v5.10.6 we have:

```
int
nouveau_fence_new(struct nouveau_channel *chan, bool sysmem,
                  struct nouveau_fence **pfence)
{
        struct nouveau_fence *fence;
        int ret = 0;

        if (unlikely(!chan->fence))
                return -ENODEV;

        fence = kzalloc(sizeof(*fence), GFP_KERNEL);
        if (!fence)
                return -ENOMEM;

        ret = nouveau_fence_emit(fence, chan);
        if (ret)
                nouveau_fence_unref(&fence);

        *pfence = fence;
        return ret;
}
```

and:

```
/* Update task and its cfs_rq load average */
static inline void update_load_avg(struct cfs_rq *cfs_rq, struct sched_entity *se, int flags)
{
        u64 now = cfs_rq_clock_pelt(cfs_rq);
        int decayed;

        /*
         * Track task load average for carrying it to new CPU after migrated, and
         * track group sched_entity load average for task_h_load calc in migration
         */
        if (se->avg.last_update_time && !(flags & SKIP_AGE_LOAD))
                __update_load_avg_se(now, cfs_rq, se);

        decayed  = update_cfs_rq_load_avg(now, cfs_rq);
        decayed |= propagate_entity_load_avg(se);

        if (!se->avg.last_update_time && (flags & DO_ATTACH)) {

                /*
                 * DO_ATTACH means we're here from enqueue_entity().
                 * !last_update_time means we've passed through
                 * migrate_task_rq_fair() indicating we migrated.
                 *
                 * IOW we're enqueueing a task on a new CPU.
                 */
                attach_entity_load_avg(cfs_rq, se);
                update_tg_load_avg(cfs_rq);

        } else if (decayed) {
                cfs_rq_util_change(cfs_rq, 0);

                if (flags & UPDATE_TG)
                        update_tg_load_avg(cfs_rq);
        }
}
```

So given that we have kernel: RIP: 0010:kmem_cache_alloc_trace+0xec/0x290 at least for nouveau we can strongly suspect that the issue appears somehow when doing the kzalloc.

I've a similar issue with my F2A85M-PRO and I also run Coreboot and Nouveau.

So I wonder if there is some issues with the RAM init code in Coreboot. Sadly the KCMA-D8 has been dropped from Coreboot so that doesn't make it easy.

I really wonder how to debug that. Do you know if there are ways to make the computer crash fast? In my case it can take a long time to crash and when it crash it tends to crash again very fast in the next boot. It also tends to crash more on heavy load like compilation (which is strange).

Maybe there are some things to try. Are you able to change Coreboot settings (with nvramcui or nvramtool) ?

From the log I've seen that you used Coreboot 4.10 or a deblobbed version.

Here's the cmos.default from src/mainboard/asus/kcma-d8 from Coreboot 4.10:

```
debug_level=Debug
multi_core=Enable
slow_cpu=off
compute_unit_siblings=Enable
iommu=Enable
nmi=Disable
hypertransport_speed_limit=Auto
max_mem_clock=DDR3-1600
minimum_memory_voltage=1.5V
dimm_spd_checksum=Enforce
ECC_memory=Enable
ECC_redirection=Enable
ecc_scrub_rate=1.28us
interleave_chip_selects=Enable
interleave_nodes=Disable
interleave_memory_channels=Enable
```

```
cpu_c_states=Enable
cpu_cc6_state=Enable
cpu_core_boost=Enable
sata_ahci_mode=Enable
sata_alpm=Disable
maximum_p_state_limit=0xf
probe_filter=Auto
l3_cache_partitioning=Disable
gart=Enable
ehci_async_data_cache=Enable
experimental_memory_speed_boost=Disable
power_on_after_fail=On
boot_option=Fallback
```

Maybe you could try to lower max_mem_clock somehow and see if that works better?

On my side I've been trying to find the time to fix my mainboard in upstream coreboot, so I didn't start tackling that issue yet.

Here's what I have on my computer instead:

```
# nvramtool -a
boot_option = Normal
reboot_counter = 0x0
hw_scrubber = Enable
interleave_chip_selects = Enable
max_mem_clock = 400Mhz
multi_core = Enable
power_on_after_fail = Disable
debug_level = Spew
slow_cpu = off
nmi = Enable
iommu = Enable
ECC_memory = Disable
nvramtool: Can not read coreboot parameter user_data because layout info specifies CMOS area that is too wide.
```

I could probably try the same thing but I've no idea how to test changes fast.

#4 - 2021-05-02 10:42 PM - bill-auger

> Sadly the KCMA-D8 has been dropped from Coreboot so that doesn't make it easy.

probably, it is still in libreboot

#5 - 2021-05-02 11:45 PM - biovoid

I have not found a way to reliably trigger the crash, but have also found it more likely under heavy load, especially compilation (as you found, GNUtoo).

Yes, I have self-built a blobless Coreboot 4.10 rom (most other defaults were kept). I am able to manipulate the configuration, though not trivially (but with preparation)—I can line up a few tests next time I go to schedule some downtime.

Here's my nvramtool output, for reference:

```
boot_option = Fallback
reboot_counter = 0x0
interleave_chip_selects = Enable
interleave_nodes = Disable
interleave_memory_channels = Enable
max_mem_clock = DDR3-1600
multi_core = Enable
debug_level = Debug
ecc_scrub_rate = 1.28us
slow_cpu = off
nmi = Disable
gart = Enable
power_on_after_fail = On
ECC_memory = Enable
ECC_redirection = Enable
hypertransport_speed_limit = Auto
minimum_memory_voltage = 1.5V
```

```
compute_unit_siblings = Enable
cpu_c_states = Enable
cpu_cc6_state = Enable
sata_ahci_mode = Enable
sata_alpm = Disable
dimm_spd_checksum = Enforce
probe_filter = Auto
l3_cache_partitioning = Disable
iommu = Enable
cpu_core_boost = Enable
ehci_async_data_cache = Enable
experimental_memory_speed_boost = Disable
maximum_p_state_limit = 0xf
nvramtool: Can not read coreboot parameter user_data because layout info specifies CMOS area that is too wide.
```

As you have mentioned, the dropping of support from Coreboot doesn't make this any easier.

I built my own Coreboot image because the current Libreboot release for this board is much older (I want to say around Coreboot 4.1, but don't remember exactly). A new release is due any day now—if anything like the osboot build (which I expect it is), it is based on Coreboot revision ad983eeec76ecdb2aff4fb47baeee95ade012225 (Nov 2019), the last commit to support the board. A configuration file is provided, but there do not appear to be substantial patches. That said, I have not tried using this version yet, as my last work done with flashing this machine preceded this commit. Maybe I'll give it a whirl, either through another self-compilation, or waiting for this Libreboot release to drop.

Leah (of Libreboot) has mentioned a desire to fork Coreboot (as Libreboot itself is merely a Coreboot distribution) and revive support for these boards, but who knows when that would come about.

#6 - 2021-05-06 08:55 PM - bill-auger

GNUtoo - can this ticket be marked as 'confirmed'? - or maybe just closed?, if there is nothing we could do about it directly - or marked 'forwarded-upstream'?, if there is some related work being done in coreboot/libreboot

biovoid - if you believe that coreboot is the problem, the first thing i would do, would be to flash the manufacturer's blob, to verify that the problem goes away

if the problem still exists, then you have at least determined that it is not related to coreboot - then perhaps there is something to be done in the parabola system

if the problem does not exist using the factory BIOS, then it is probably not related to parabola, and would behave the same with any distro - based on your description, i would contact leah rowe, and volunteer to experiment on your computer

#7 - 2021-08-11 12:33 AM - GNUtoo

I had hangs on my T400 that I managed to solve.

I think I solved it by cleaning the RAM as there was some dust on the RAM connector.

Though I don't remember who told me that trick, so I don't remember what is advised to use to clean that.

I've used 90% alcohol but that probably contains water in it which could then corrode the connector after some time.

I'll try that too on my F2A85M-pro to see if it was a software or hardware problem.

Files

kernel_trunc.log	10.6 KB	2021-01-31	biovoid
Xorg.0.log	63.2 KB	2021-01-31	biovoid
20-nouveau.conf	99 Bytes	2021-01-31	biovoid
kernel.log.tar.gz	93 KB	2021-01-31	biovoid