# Panic Attack

## Finding some order in the panic chaos

Kernel Recipes

Guilherme G. Piccoli (Igalia)

2023-09-26 / Kernel Recipes

# Context

- Interest in having a panic log collecting tool on Arch / SteamOS

- Analysis of kernel infra available - different use cases:
  - kdump == more data collected, heavier on resources
  - pstore == log collected on panic -> lightweight, but less data

- By playing with kdump/pstore, crossed paths with panic notifiers
  - Panic path is full of trade-offs / conflicting goals
  - Panic notifiers discussions, ideas and eventual refactor

- Some other orthogonal problems on panic time
  - Interrupt storms / Graphics on panic

# Disclaimer

- Feel free to interrupt with questions
  - Multiple concepts / dense topic
  - Risks of "assumed knowledge"

- kexec set of recent problems won't be addressed here
  - Memory preserving across kexec boots
  - SEV / TDX problems with kexec
  - Unikernels support, etc.

# Outline

- **The genesis of this work: SteamOS**
- Panic notifiers: discussion and refactor
- Chaos on kdump: a real case of interrupt storm
- Challenges of GFX on panic: dream or reality?

# Where all started: Steam Deck


- Steam Deck, from Valve
  - CPU/APU AMD Zen 2 (custom), 4-cores/8-threads
  - 16 GB of RAM / 7" display
  - 3 models of NVMe storage (64G, 256G, 512G)

# Deck's distro: SteamOS 3

- Arch Linux based distro with gamescope (games) and KDE Plasma (desktop)
- Sophisticated stack for games: Steam, Proton (Wine), DXVK, VKD3D, etc
- Arch Linux has no kdump official tool
- Steam Deck community would benefit of such tool for panic log collection!

# Requirements: what logs to collect?

- Collect the most logs we can: dmesg (call trace), tasks' state, memory info
  - Though being careful with size - should be easy to share
- Information that could be used for kernel/HW debugging
- Rely on in-kernel infrastructure for that - don't reinvent the wheel

# How to collect such logs? Kernel infra

- kdump: kexec-loaded crash kernel
  - kexec to a new kernel to collect info from the broken kernel
  - Requires pre-reserved memory (>200MB usually)
  - Collects a vmcore (full memory image) of the crashed kernel
  - Lots of information, but heavy / hard for users to share it
- pstore: persistent storage log saving
  - Save dmesg during panic time to some backend
  - Multiple backends (RAM, UEFI, ACPI, etc)
  - Also multiple frontends (oops, ftrace, console, etc)
  - Enough amount of information? (dmesg only)
- Both tools benefits from userspace counter-part
  - Kdump tooling common (Debian/Fedora), but not Arch

# Presenting kdumpst

- **kdumpst** is an Arch Linux kdump and pstore tool
- Available on **AUR**, supports GRUB and initcpio / dracut

- Defaults to pstore; currently only ramoops backend (UEFI plans)
- Used by default on Steam Deck, submits logs to Valve

- But how to improve the amount of logs on dmesg?
  `panic_print` FTW!

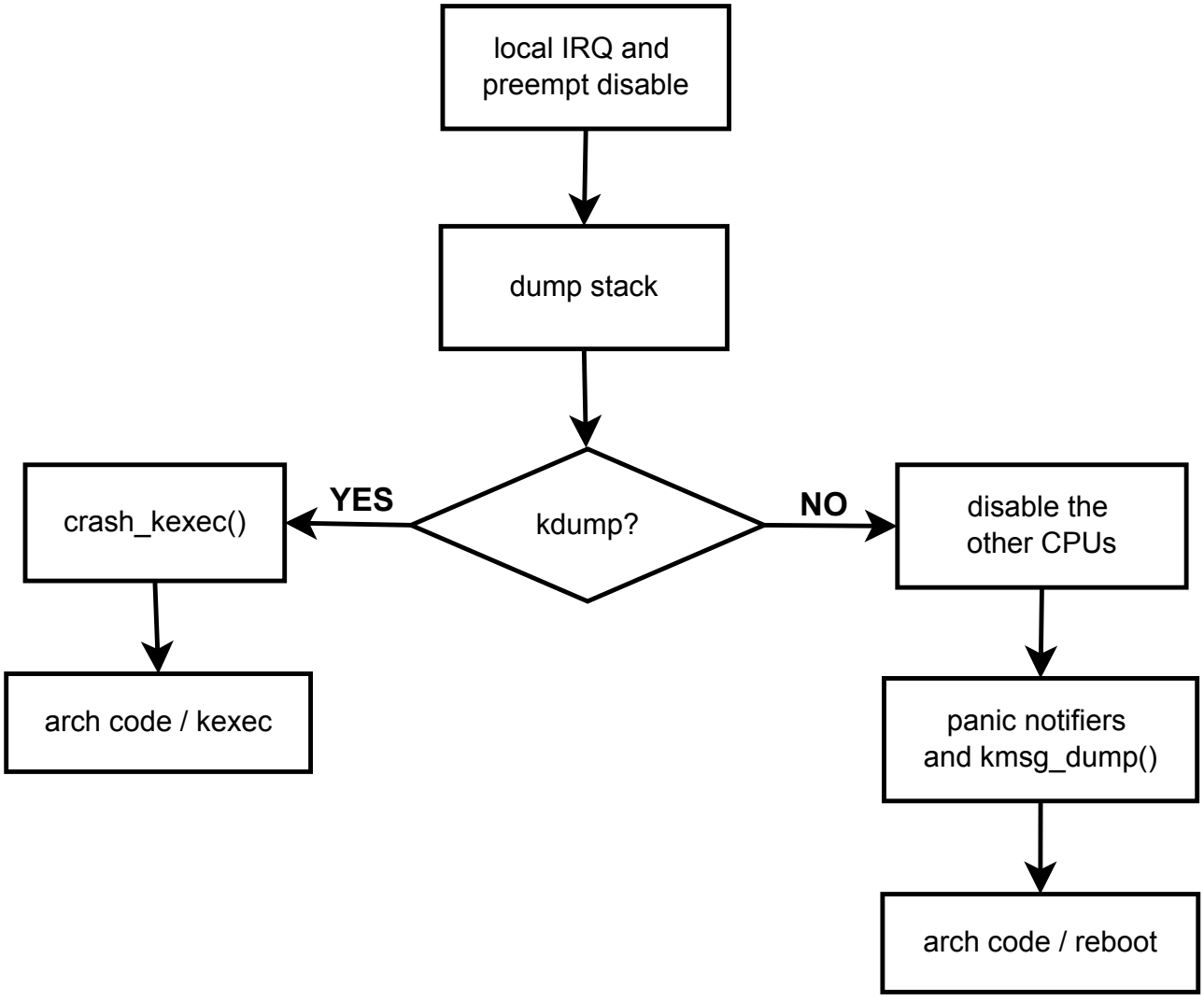
# `panic_print` VS pstore ordering

- `panic_print` parameter allows to show more data on dmesg during panic
  - Tasks info, system memory state, timers

- But such function runs after pstore! So can't collect the data.

- Idea: re-order the code **[discussion]**
  - Move the call earlier in the panic path

# Panic (over-simplified) code path



# The code re-ordering

```
/* Simplified function names */

void panic()
[...]
                        <-----------------------|
if (!panic_notifiers)                           |
    crash_kexec(); /* kdump */                  |
                                                |
panic_notifiers();                              |
                                                |
kmsg_dump(KMSG_DUMP_PANIC); /* pstore! */       |
                                                |
if (panic_notifiers)                            |
    crash_kexec();                              |
                                                |
panic_print(); ------------------------------|
[...]
```

# And then, the discussion starts...

- Problems with such approach: `panic_print` before kdump is risky
  - **[discussion]** with Baoquan and others

- Alternative: propose less invasive change, moving that **before pstore** only **[discussion]**

- New problem then: what if users want `panic_print` before kdump?
  - Makes sense if vmcore is too much
  - Only possible if we run the panic notifiers before kdump! So the notifiers journey begins...

# Outline

- The genesis of this work: SteamOS
- **Panic notifiers: discussion and refactor**
- Chaos on kdump: a real case of interrupt storm
- Challenges of GFX on panic: dream or reality?

# Notifier call chains

- List of callbacks to be executed (usually) in any order
  - There's a (frequently unused) "priority" tune for call ordering
- Multiple types - atomic callbacks, blocking callbacks, etc
  - Panic notifiers == list of atomic callbacks executed on panic

```
/* Example from kernel/rcu/tree_stall.h */
/* Don't print RCU CPU stall warnings during a kernel panic. */
static int rcu_panic(...)
{
    rcu_cpu_stall_suppress = 1;
    return NOTIFY_DONE;
}
static struct notifier_block rcu_panic_block = {
    .notifier_call = rcu_panic,
};
atomic_notifier_chain_register(&panic_notifier_list, &rcu_panic_block);
```

# Deep dive into panic notifiers

- Any driver (even OOT) can register a notifier, to do...anything!
  - Risky for kdump reliability / but sometimes, notifiers could be necessary
- "Solution": a new kernel parameter, `crash_kexec_post_notifiers`
  - Proper name for a bazooka shot: all-or-nothing option, runs ALL notifiers before kdump
- Middle-ground idea: panic notifiers filter! User selects **which** notifiers to run
  - kdump maintainers kinda welcome the feature: **[discussion]**
  - But really paper over a real issue: notifiers is a no man's land
  - **Very good analysis** from Petr Mladek exposed the need of a refactor

# Mladek's refactor proposal

- Split panic notifiers in more lists, according to their "goals"
  - Information ones: extra info dump, stop watchdogs
  - Hypervisor/FW poking notifiers
  - Others: actions taken when kdump isn't set (LED blink, halt)
- Ordering regarding kdump
  - Hypervisor list before kdump
  - Info list also before, **IF** any `kmsg_dump()` is set
  - Final list runs only if kdump isn't set
- V1 submitted ~1y ago
  - Special thanks to Petr Mladek for the idea and all reviews. Thanks also Baoquan and Michael Kelly (Hyper-V) for the great discussions!

# 1st step - fixing current panic notifiers

- First thing: build a list with all existing in-tree panic notifiers
  - As of today (6.6-rc2): 47 notifiers (18 on `arch/`)
- Fix / improve them, before splitting in lists. Some patterns:
  - **Decouple multi-purpose** notifiers
  - **Change ordering** through the notifier's priorities
    - Machine halt or firmware-reset - put'em to run last
    - Disabling watchdogs (RCU, hung tasks): run ASAP
  - **Avoid regular locks**
    - Panic path disables secondary CPUs, interrupts, preemption
    - `mutex_trylock()` and `spin_trylock()` FTW

# Real example: pvpanic

```
/* drivers/misc/pvpanic/pvpanic.c - simplified code */
static void pvpanic_send_event() {
-     spin_lock(&pvpanic_lock);
+     if (!spin_trylock(&pvpanic_lock))
+         return;

static int panic_panic_notify(...) {
        pvpanic_send_event(PVPANIC_PANICKED);
}

[...]

+ /* Call our notifier very early on panic */
static struct notifier_block pvpanic_panic_nb = {
    .notifier_call = pvpanic_panic_notify,
-     .priority = 1,
+     .priority = INT_MAX,
};
```

# List splitting (yay, a 4th list!)

- Original plan was splitting in 3 lists, but... ended-up with 4
- **Hypervisors** list: hypervisor/FW notification, LED blinking
  - Hyper-V, PPC/fadump, pvpanic, LEDs stuff, etc
- **Informational** list: dump extra info, disable watchdogs
  - KASLR offsets, RCU/hung task watchdog off, `ftrace_dump_on_oops`
- **Pre-reboot**: includes the remaining ones (halt, risky funcs)
  - S390 and PPC/pseries FW halt, IPMI interfaces notification
- **Post-reboot**: contains previously hardcoded (arch) final calls
  - SPARC "stop" button enabling (if reboot on panic not set)
  - List to be renamed on V2 (loop list)

# The notifier "levels" model

- One of the biggest questions regarding panic notifiers
  - Which ones should run **before** kdump?
  - Usual / possible answer: low risk / necessary ones

- Introduce the concept of **panic notifier levels**
  - Fine-grained tuning of which lists run before/after kdump
  - Defaults to:
    - Hypervisor always run before
    - Sometimes informational also (if `kmsg_dump()` is set)

- Implementation maps levels into bits and order the lists
  - Was gently called "black magic" on review

# Subsequent improvements

- Proposal to convert `panic_print` into a panic notifier
  - Good acceptance, fits perfectly the informational list

- Stop exporting `crash_kexec_post_notifiers`
  - Sadly some users of panic notifiers forcibly set this parameter in code
  - Hyper-V is one of such users, and they have reasons for that...

# Hyper-V case / arm64 ~~custom crash handler~~

- Hyper-V requires hypervisor action in case of kdump
  - Requires to unload its "vmbus connection" before crash kernel takes over
- x86 does it on crash through `machine_ops()` crash shutdown
  - arm64 though doesn't have similar architecture hook
- **Discussion** with arm64 maintainers revealed little interest in adding that
  - Unworthy complexity / not a good idea to mimic x86 case
- Forcing panic notifiers seems a last resort for Hyper-V
  - Unless some alternative for arm64 is implemented

# Pros / Cons and follow-up discussion

- Exhaustive **discussion** exposed plenty conflicting views

- First of all, not really clear what should run before kdump
  - The notifiers lists are incredibly flexible and "loose"
  - How to be sure anyone knowledgeable on panic will review?
  - Brainstorm: somehow force registering users to add the cb name to a central place?

- Less is more: too much flexibility is not a good fit for panic

- Also, are notifier lists reliable on panic path?
  - What if memory corruption corrupts the list?
  - Alternatives? Hardcoded calls? (headers/exports hell)

# Next steps / V2

- Rework lists as suggested (move some callbacks here and there)
- Split submission - first the lists, then the refactor (kdump vs notifiers order)
- Consider ways of improving panic notifiers review
  - Improve documentation
  - Central place for registering !?

# Outline

- The genesis of this work: SteamOS
- Panic notifiers: discussion and refactor
- **Chaos on kdump: a real case of interrupt storm**
- Challenges of GFX on panic: dream or reality?

# Shifting gears: an interrupt storm tale

- Another painful area to deal with is device state on kdump
- A regular kexec would handle device's quiesce process
  - `.shutdown()` callback
  - Crash kexec (kdump) can't risk that -> way more limited environment
- Real case: device caused an interrupt storm, kdump couldn't boot

# The problem

- Intel NIC running under PCI-PT (SR-IOV)
  - No in-tree driver - DPDK instead
- Custom tool collecting NIC stats, triggered weird NIC FW bug
  - Symptom: lockups on host, non-responsive system
  - (Non-trivial) cause: **NIC interrupt storm**
- Kdump attempt: unsuccessful -> crash kernel hung on boot
  - Guess what? Still the interrupt storm!

# Look 'ma, no PCI reset

- Despite kexec is a new boot, there are many differences from FW boot
  - A fundamental limitation is the lack of PCI controller reset
- x86 has no "protocol" / standard for root complexes resets
  - PPC64 has a FW-aided PCI reset (`ppc_pci_reset_phbs`)
- Multiple debug attempts later...an idea: **clear devices's MSIs** on boot
  - But how to achieve this? PCI layer is initialized much later
- **x86 early PCI infrastructure** FTW! (Special thanks to Gavin Shan)

# `pci=clearmsi` proposal

- Through the early PCI trick, we could clear the MSIs of all PCI devs
  - Interrupt storm was shut-off and kdump boot succeeded
  - **Patches** to linux-pci (~3y ago)

- Some concerns from Bjorn (PCI maintainer)
  - First: limited approach -> `pci_config_16()`
  - This conf mode access is limited to first domain/segment

- Other concern: solution only for x86
  - In principle, this affects more archs

# Discussion

- Also, was not really clear exactly what was the precise point of failure
  - Thanks Thomas Gleixner for **clarifying** that
  - Interrupt flood happens right when interrupts are enabled on `start_kernel()`

- MSIs are DMA writes to a memory area (interrupt remapping tables)
  - An IOMMU approach was suggested
    - Clearing these mappings and IOMMU error reporting early in boot

- Proper cleaning routines to run on panic kernel also suggested

# Potential next steps

- Attempt implementing the IOMMU idea
  - Too limited? What if no IOMMU?
- Investigate other archs to see how's the status
  - Reliably reproduce the problem!
- Extend early PCI conf access mode?
  - Bjorn would be unhappy

# Outline

- The genesis of this work: SteamOS
- Panic notifiers: discussion and refactor
- Chaos on kdump: a real case of interrupt storm
- **Challenges of GFX on panic: dream or reality?**

# Final problem: GFX on panic

- GPUs are complex beasts / interrupts are disabled on panic
  - Even regular kexec are challenging for them!

- Currently, no reliable way to dump data on display during panic
  - Though it would be great for users to see something on crash
  - Reliable GFX on kdump? Wishful thinking

# Framebuffer reuse

- While working on kdumpst, experimenting with GFX on kdump
  - Managed to make it work only with framebuffers
  - Why not restore the FB on kdump then?

- Interesting **discussion** shows it's definitely not trivial
  - Once a GPU driver takes over, HW is reprogrammed
  - GOP driver (UEFI) programs FB/display
  - We'd need to reprogram the device either on panic (ugh) or on kdump kernel

# Current approaches

- Noralf Trønnes **proposal** (~4y ago)
  - Iterates on available framebuffers, find a suitable one

- Jocelyn Falempe **proposal** (last week)
  - Works with simpledrm currently, API to get a scanout buffer
  - Seems on early stages, with great potential / community acceptance

- Panic time approaches are risky / limited, must be simple
  - Not sure if that's possible one day for amdgpu / i915

# Different approach: FW notification

- What if we print nothing on panic, but defer for FW / next kernel?

- UEFI panic notification **proposal** (~1y ago)
  - Simple UEFI variable set on kernel panic (through notifiers!)
  - Next kernel clears the var (and potentially prints something)
  - Simple and flexible - FW could plot a different logo

- UEFI maintainer (Ard) not really convinced
  - Suggestions for using UEFI pstore for tracking that
  - Orthogonal goals / limited space on UEFI / dmesg "privacy"

- Next steps: might try to implement that solution in a prototype

# Conclusion

- Quite a long path, from Linux gaming to panic notifiers refactor
- Everything on panic is polemic / conflicting
  - "Slightly" long road ahead for the refactor
  - V2 of the refactor soon(tm), not so invasive
- HW quiesce on crash kexec is still full of issues
  - Interesting area for some research / multi-arch work (IMHO)
- GFX on panic: still in early stages, other OSes / game consoles seems to have it
  - The UEFI approach, while kinda orthogonal, it's way simpler


Feel free to reach me on IRC (**gpiccoli** - OFTC/Libera)