Google

KVM FORUM

All bark no bite:

vCPU stall detection for KVM guests

sebastianene@google.com

Agenda

- Motivation for adding a new watchdog-like device
- Looking at the existing watchdog infrastructure in Linux Kernel
- Emulating the device in crosvm and state diagram
- Linux kernel frontend driver
- Next steps & lessons learned

The Problem

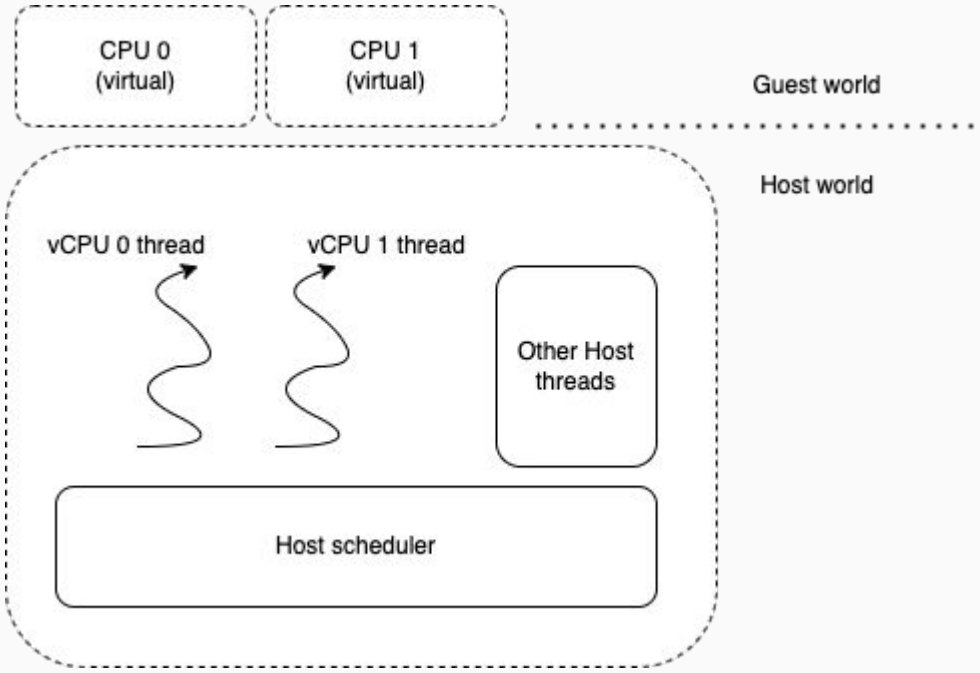
Why add a new stall detector for KVM guests ?

- No mechanism to detect stalled guests from the outside world (host):
  - Stalled vCPU threads appear busy (runnable!) to the host
  - Existing solutions (eg. Chrome OS SMC watchdog) do not account for **stolen time** and can result in spurious resets
- **Stolen time** represents the time taken from the guest while the host is busy doing something else
- Need to handle vCPU hotplug in the guest

Watchdog framework in the Linux Kernel

- The Linux Kernel uses the /dev/watchdog interface to receive userspace notifications
  - **not KVM related but bare-metal behaviour !**
- In normal operation, the notification informs the system that everything is in order
  - This indicates that the userspace daemon is still responsive
- If the notification is not received by the watchdog, the system dumps its state before rebooting

Why this doesn't work for guests ?



Because:

- We need to account for stolen time
  - vCPUs are backed by POSIX host threads which can be scheduled independently
  - What happens if the watchdog expires while the vCPU is not running?
- Since stolen time is accounted separately for each vCPU, we require a strong CPU affinity when we send the 'heart beat' notification which cannot be guaranteed by the userspace
  - E.g. CPU hotplug

What can possibly go wrong ?

1. Guest runs and sends an MMIO notification to the emulated watchdog device. While writing device registers, it exits the guest on the data abort path.

1. The VMM on the host receives the notification and re-arms the timer for the next expiration period.

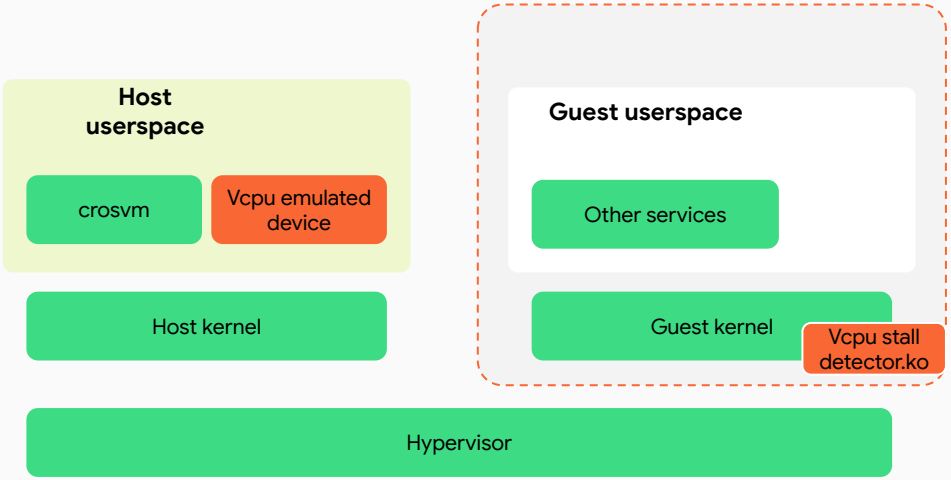
1. The timer starts decrementing its internal counter, but the guest is not scheduled to run.

1. The timer expires because the guest wasn't scheduled in time
   => this triggers a spurious reset !

The guest is unaware that it was not scheduled in time !

A Solution

Proposed solution



- Add a backend driver in CrosVM that emulates a watchdog-like device
- Define a set of registers for the emulated device
- Add a frontend driver in the Linux Kernel guest that knows how to interact with the added device

CrosVM backend driver

- Every MMIO device is abstracted by the BusDevice interface
- A device registers on the memory bus by providing the size and the memory region to KVM
- MMIO events are dispatched to the registered device which performs the necessary logic
- A separate crosvm worker thread spins up for the internal clock
- If the internal clock decrements the internal counter to 0, the vcpu stall detector fires !
- Extract guest time from /proc/stat to adjust per-vCPU timer expiration

Per-vcpu register frame for the stall detector device

Register Name	Default Value (after reset)	Access permission	Offset	Usage
WDT_REG_STATUS	0	R/W	0x0	Enable(0x1) / Disable(0x0) the watchdog
WDT_REG_LOAD_CNT	0	R/W	0x4	A write in this register will load the number of starting ticks in the WDT_REG_CURRENT_CNT register.
WDT_REG_CURRENT_CNT	0	R	0x8	This register is decremented on each clock tick. When it reaches '0' value and pre-timeout is not enabled, it asserts the output line.
WDT_REG_CLOCK_FREQ_HZ	10 (Hz)	R/W	0xC	Internal clock frequency range [100Hz ..1Hz]

Linux kernel frontend driver

- Standard misc driver
  - Upstream objection to proposed inclusion in the watchdog framework
- The stall detector is probed using device-tree
- Deliver the `heart beat` notifications from per-cpu hrtimers
- Registers for cpu hotplug events
  - Disarm/re-arm the hrtimer accordingly

```
vmwdt@9030000 {
  compatible = "qemu,vcpu-stall-detector";
  reg = <0x9030000 0x10000>;
  clock-frequency = <10>;
  timeout-sec = <8>;
};
```

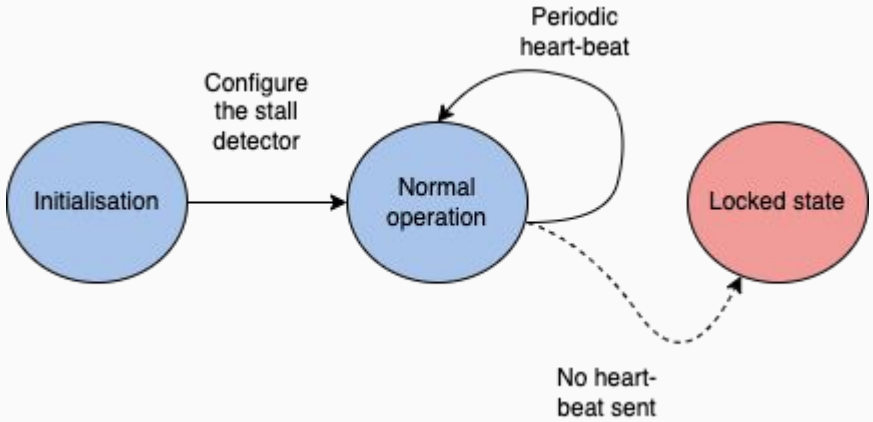
The states of the vcpu stall detector

Initialisation
- Configure the internal clock register WDT_REG_CLOCK_FREQ_HZ
- Compute the number of ticks that the counter will start decrementing and program WDT_REG_LOAD_CNT
- Enable the stall detector by writing WDT_REG_STATUS

Normal operation:
- Update the number of ticks in WDT_REG_LOAD_CNT

Locked state:
- Send a reset vcpu state message to the guest

If the guest stalls it will fail to deliver a heart beat and we will enter the Locked state.



Next steps & lessons learned

- Report diagnostic messages from the VMM (crosm) before resetting the guest
- Patch landed upstream after 12 revisions ! https://git.kernel.org/pub/scm/linux/kernel/git/gregkh/char-misc.git/commit/?h=char-misc-next&id=6c93c6f3bad468ce4b8c843227d60fbeb02fd741
- CrosVM changes now merged: https://chromium-review.googlesource.com/c/crosvm/crosvm/+/3768290
- Functionality targeting Android U
  - Although nothing Android specific for this mechanism

Thank you !