svn UPDATE

by Steven Kreuzer

I hope your new year is off to a wonderful start and that you had some time over the holidays to upgrade your machines to FreeBSD 12. While I think it is safe to say that the time and effort put into FreeBSD in 2018 was nothing short of remarkable, I have a strong suspicion that 2019 is going to be an even more exciting year for development. We are not even a full month into this year, and I am already seeing some very interesting changes being committed to HEAD. I am eagerly looking forward to what the future has in store and I hope you are feeling the same.

vmm(4): Mask Spectre feature bits on AMD hosts — https://svnweb.freebsd.org/changeset/base/343166

For parity with Intel hosts, which already mask out the CPUID feature bits that indicate the presence of the SPEC_CTRL MSR, do the same on AMD. Eventually, we may want to have a better support story for guests, but for now, limit the damage of incorrectly indicating an MSR we do not yet support.

nvdimm: Add a driver for the NVDIMM root device — https://svnweb.freebsd.org/changeset/base/343143

The NVDIMM root device is parent to the individual ACPI NVDIMM devices. Add a driver for the NVDIMM root device that can own enumeration of NVDIMM devices as well as NVDIMM SPA ranges that the system has.

vmm(4): Take steps towards multicore bhyve AMD support — https://svnweb.freebsd.org/changeset/base/343075

Vmm's CPUID emulation presented Intel topology information to the guest, but disabled AMD topology information and in some cases passed through garbage. For example, CPUID leaves 0x8000_001[de] were passed through to the guest, but guest CPUs can migrate between host threads, so the information presented was not consistent. This could easily be observed with 'cpucontrol -i 0xfoo /dev/cpuctl0'.

Slightly improve this situation by enabling the AMD topology feature flag and presenting at least the CPUID fields used by FreeBSD itself to probe topology on more modern AMD64 hardware (Family 15h+). Older stuff is probably less interesting. I have not been able to empirically confirm it is sufficient, but it should not regress anything either.

Fix bhyve's NVMe Completion Queue entry values — https://svnweb.freebsd.org/changeset/base/342762

The function which processes Admin commands was not returning the Command Specific value in Completion Queue Entry, Dword 0 (CDW0). This affects commands such as Set Features, Number of Queues which returns the number of queues supported by the device in CDW0. In this case, the host will only create 1 queue pair (Number of Queues is zero based). This also masked a bug in the queue counting logic.

Create new EINTEGRITY error with message "Integrity check failed" — https://svnweb.freebsd.org/changeset/base/343111

An integrity check such as a check-hash or a cross-correlation failed. The integrity error falls between EINVAL that identifies errors in parameters to a system call and EIO that identifies errors with the underlying storage media. EINTEGRITY is typically raised by intermediate kernel layers such as a filesystem or an in-kernel GEOM subsystem when they detect inconsistencies. Uses include allowing the mount(8) command to return a different exit value to automate the running of fsck(8) during a system boot.

These changes make no use of the new error, they just add it. Later commits will be made for the use of the new error number and it will be added to additional manual pages as appropriate.

Add support for marking interrupt handlers as suspended — https://svnweb.freebsd.org/changeset/base/342170

The goal of this change is to fix a problem with PCI shared interrupts during suspend and resume.

I have observed a couple of variations of the following scenario. Devices A and B are on the same PCI bus and share the same interrupt. Device A's driver is suspended first, and the device is powered down. Device B generates an interrupt. Interrupt handlers of both drivers are called. Device A's interrupt handler accesses registers of the powered-down device and gets back bogus values (I assume all 0xff). That data is interpreted as interrupt status bits, etc. So, the interrupt handler gets confused and may produce some noise or enter an infinite loop, etc.

This change affects only PCI devices. The pci(4) bus driver marks a child's interrupt handler as suspended after the child's suspend method is called and before the device is powered down. This is done only for traditional PCI interrupts, because only they can be shared.

Optimize RISC-V copyin(9)/copyout(9) routines — https://svnweb.freebsd.org/changeset/base/343275

The existing copyin(9) and copyout(9) routines on RISC-V perform only a simple byte-by-byte copy. Improve their performance by performing word-sized copies where possible.

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, daughter, and dog.