AER functionality of pass-throughed PCI-e device in Qemu

June 1 2017
Cao jin <caoj.fnst@cn.fujitsu.com>
Fujitsu Limited.
Outline

- Background knowledge
- What’s the problem
- Solution
- Current status and future work
PCI device error mechanism

- PCI device is poor

2 kinds of errors
- PERR: Parity Error. Used for signaling data parity errors on all transactions except Special Cycle transaction
- SERR: System error. Used for signaling address parity error, and data parity error in Special Cycle transaction
PCI Express device error mechanism

- **AER: Advanced Error Reporting**
  - PCIe proprietary, optional extended capability
  - Support Error classification
    - Correctable Errors
    - Uncorrectable Errors
      - Non fatal errors
      - Fatal errors
  - Support severity programming for uncorrectable errors
Advanced Error Reporting

■ Correctable Errors

The errors that hardware can recover without any loss of information. Hardware corrects these errors without software intervention.
Advanced Error Reporting

- Uncorrectable Errors
  - Fatal: render the particular link and related hardware unreliable
  - Non fatal: cause a particular transaction to be unreliable but the link is otherwise fully functional
Advanced Error Reporting

Error logging & signaling

1. AER capability register is implemented by PCIe element who supports AER
2. Root complex is a abstract concept in spec. In practice, it equals root port in X86.
Error recovery on Linux

Recovery is platform specific

Documentation/PCI/pci-error-recovery.txt

- Provide error recovery infrastructure for linux. Error recovery API:
  
  ```c
  struct pci_error_handlers
  {
    int (*error_detected)(struct pci_dev *dev, enum pci_channel_state);
    int (*mmio_enabled)(struct pci_dev *dev);
    int (*link_reset)(struct pci_dev *dev); // Deleted recently, AER driver will do link reset
    int (*slot_reset)(struct pci_dev *dev);
    void (*resume)(struct pci_dev *dev);
  }
  ```

Documentation/PCI/pcieaer-howto.txt

- Describe the basics of AER driver
- Provides the infrastructure to support PCIe AER capability
- Gathers the comprehensive error information if errors occurred
- Performs error recovery actions
Error recovery process on Linux

1. Error is detected, logged, then sent to root port
2. Interrupt to signal OS via MSI/MSIX
3. AER IRQ handler perform recovery mainly via recovery API implemented by device driver
QEMU

- PCI-e device is pass-throughed to VM for performance
  - Via VFIO driver: Documentation/vfio.txt
  - VFIO provides a framework to implement user space driver
  - Qemu acts as the user space driver for the pass-throughed device
What’s the problem

- QEMU VM with pass-throughed PCIe device will *vm_stop* on any error event

![Diagram showing QEMU VM with pass-throughed PCIe device and its interaction with VFIO, AER, and the root port in the Host. The diagram also shows the software and hardware layers.](image-url)
Solutions

- Solution 1 starts with
  - QEMU emulates the hardware logic to signal error message to guest
  - Then just let guest do the recovery
Solution 1

■ Problem A
  ■ Depends on: PCIe Multi-function Hot plug/unplug[*]
    • Causation: all functions of a multi-function device could be assigned
    • Commit: 0d1c7d88ad & 3f1e1478db of QEMU

■ Problem B
  ■ Configuration restraint of multi-function device
    • What: topology of multi-function device should look the same between host and guest.
    • Why: link reset request from one function would reset all functions
    • Result: involve many check during initialization

Solution 1

Problem C: Link reset 2 times on fatal error.

- Found during test
- AER driver in host reset link first
- vfio-pci device detect fatal error in host, forward these info to guest as it is
- vfio-pci translate guest link reset to host link reset[*]

- Effort was made to serialize them: involves vfio_pci driver modification

[*]If there is ever a case where a driver within the guest could trigger a link reset for the purposes of error recovery when the host has not, I think this must be the case – Alex Williamson <alex.williamson@redhat.com>
Solution 1

Problem D: guest will oops by igb driver during recovery
- We use Intel 82576 NIC for test
- Our patchset exposed the issue of igb driver, also exposed that 2 link reset are not fully separated
- Fix guest oops issue with 629823b872 of kernel

Problem D solved, but still can’t recover
- Hardware resetting from host link reset is parallel with guest recovery
- Guest recovery involves many register access(cfg&mmio)
- Guest register reading got invalid value
Solution 1

After much investigation, finally got a really workable version

Key point: skip host link reset for fatal error

- In regular environment, a standard fatal error recovery steps look like:
  error_detected → … → reset link → … → resume

- In our case, it becomes
  reset link (host) → error_detected (guest) → … → reset link (guest) → … → resume (guest)

Don’t translate a guest link reset to a host link reset

- Why: guest link reset should reset the virtual link inside Qemu, a virtual link represented by software would never be broken.

- So, only virtual devices under the link need to be reset. In our case, vfio-pci device’s reset will be translate into a FLR (Function level reset)

Solution 1

Comments on the workable version

- If skip host link reset, how guarantee vfio’s user will do the link reset?
  - VFIO’s user is not necessarily a VM, could be dpkg, etc.
  - User is not reliable.

- Function Level Reset (FLR) doesn’t equal to link reset.

Conclusion

- Hard issue, Need re-consideration for complete solution
Solution 2

- Non fatal recovery only
  - From Michael S. Tsirkin <mst@redhat.com>
  - Could workaround the link reset issue for fatal error
  - Also take multi-function device which has different drivers into consideration

- Community Sounds?
  - Fatal error has witness, but non fatal error?
  - Extensibility(fatal error support in the future)?
Current status and future work

For solution 2

Can we find a real scenario that could trigger real non-fatal error?

- Hardware error mainly results from heat, humidity, dust, vibration and bad electrical connections.
- It is hard to trigger real hardware errors.

- AER driver debugging uses aer_inject tools to fake error (include driver & user space application)

For solution 1, continue the investigation

- SR-IOV: enterprise use case, VF doesn’t have link reset issue
- Optimization to fully skip the back-to-back link reset
Design principles – Alex Williamson

■ Do the right thing for the guest
  ■ Don't presume that different reset types are equivalent
  ■ leaving gaps where we expect the guest/host to do a link reset and don't follow through on other reset requests.
  ■ Notify the guest immediately for the error.

■ Do the right thing for the host
  ■ Should not give the user the opportunity to leave a device in a state where we haven't at least performed a bus reset on link error (which means host link reset is necessary).
shaping tomorrow with you