KVM Platform Device Passthrough

Eric Auger, Linaro
KVM Forum
Oct 14, 2014
Goal: efficiently assign platform devices to KVM guests
Agenda

- VFIO Framework
- Focus on IRQ assignment
  - Understand legacy frameworks
  - Why hardware-assisted IRQ forwarding is crucial?
- Forwarded IRQ Integration with KVM/VFIO
- Experimental Results
VFIO Platform Driver

- allows user-side to
  - mmap device MMIO regions
  - route physical IRQ to eventfd
  - Dma map buffers on iommu
QEMU VFIO device

- Setup routes between guest and assigned device
  - MMU
  - IOMMU
  - IRQ injection path

- Generate guest device device tree node
MMIO & IRQ Paths
ARM IRQ Handling

Level Sensitive
IRQ line request

inactive  pending  Active & pending  Active  Active & pending  pending

ack  device Iq status reset  deactivate

ISR

time

Edge Sensitive
IRQ line request

inactive  pending  Active  Active & pending  pending

ack  deactivate

ISR

time
Assigned Level Sensitive IRQ Model

- Physical IRQ
- HOST
- ISR
- thread
- ACK
- deactivate
- virtual IRQ
- GUEST
- ISR
- device IRQ
- status reset
- ACK
Level Sensitive IRQ Implementation Challenges

1) Physical IRQ completion
2) Virtual IRQ modeling
3) Virtual IRQ completion propagation
Basic vfio/irqfd ARM porting

- VFIO Mask/unmask
- Trap on completion
Performance Challenges on ARM

- 1 VM switch when injecting
- 1 VM switch when completing
- VM Switch really costly on ARM

Goal: Propose a new method to save completion VM switch using ARM GIC virtualization features
GIC Forwarding Feature

- GIC can automatically complete physical IRQ on virtual IRQ completion
- Host only drops the running priority of the CPU I/F to allow other physical IRQs to be signaled
- Same IRQ cannot be signaled before its deactivation by GIC HW
Forwarded IRQ Patch

- “ARM: forwarding physical interrupts to a guest VM” from M. Zyngier
  - Enable mode where priority drop and deactivate are separated, Linux wide
    - Current used mode is simultaneous prio drop & deactivate
  - Provides separate operations to program IRQ forwarding at
    - IRQCHIP
    - VGIC
vfio/irqfd/forward

- No mask/unmask anymore
- Guest completion propagated by GIC HW
- No VM switch at completion
- Natural and optimized implementation
IRQ Path with KVM (irqfd/forward)

Host

Qemu

Xgmac QEMU VFIO DEVICE

vfio driver

iommu driver

KVM/irqfd

Physical IRQ

IO platform device

IOMMU

Guest

Xgmac driver

HW completion
Forwarded IRQ Integration

Allow userspace to configure forwarding of a VFIO device IRQ

VFIO DEVICE_SET_IRQS (irq_index, eventfd)

KVM_SET_DEVICE_ATTR (vfio fd, irq index, gsi)

KVM_IRQFD (eventfd, gsi)

VFIO platform driver

is_forwarded?

get hwirq from irq index

KVM-VFIO device

Set_forwarded()

KVM

VGIC

VGIC forwarding programming

VFIO_DEVICE_SET_IRQS

KVM_SET_DEVICE_ATTR

KVM_IRQFD
Performance Measures

- **Calxeda Midway**
  - Communication between 2 nodes
  - 1Gb/s switch

- **2 xgmacs**
  - eth0 assigned to host
  - eth1 assigned to guest if any

- **Versions:**
  - All kernels are 3.17rc3
  - QEMU is 2.1.0
Comparison

- **Native Performance**
  - Node 1
    - host 10.5.3.7
      - netperf client, IRQ count
    - Eth0
    - TCP_STREAM (Tx)
    - TCP_MAERTS (Rx)
    - TCP_RR (RTT)
  - Node 2
    - host 10.5.3.10
      - netserver
    - Eth0

- **Guest Performance**
  - host 10.5.3.7
    - Eth0
    - Eth1
  - guest 10.5.3.8
    - netperf client, IRQ count
Round Trip Time

TCP_RR

Native Perf: 18350 trans/s
Xgmac IRQ rate on guest (IRQ/s)

Native IRQ/s:
- Tx: 41901
- Rx: 115162
- RR: 36700

% versus native:
- irqfd: 19025
- forward: 24179
- irqfd: 26952
- forward: 36413
- irqfd: 26223
- forward: 27051
Throughput with 3 TCP/IP patterns

<table>
<thead>
<tr>
<th>Native BW (Mb/s)</th>
<th>Tx</th>
<th>Rx</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>669</td>
<td>933</td>
</tr>
<tr>
<td>file</td>
<td>449</td>
<td>578</td>
</tr>
<tr>
<td>bulk</td>
<td>933</td>
<td>933</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting</th>
<th>default</th>
<th>file</th>
<th>bulk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native BW (Mb/s)</td>
<td>669</td>
<td>449</td>
<td>333</td>
</tr>
<tr>
<td>Remote Tx &amp; Rx socket buffer size (-S)</td>
<td>8kB</td>
<td>8kB</td>
<td>64kB</td>
</tr>
<tr>
<td>Local send size (-m)</td>
<td>8kB</td>
<td>4kB</td>
<td>8kB</td>
</tr>
<tr>
<td>Remote received size (-M)</td>
<td>8kB</td>
<td>4kB</td>
<td>8kB</td>
</tr>
</tbody>
</table>

Legend:
- irqfd
- forward
Status & Next
# QEMU patches & dependencies

<table>
<thead>
<tr>
<th>#</th>
<th>QEMU Patches</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KVM platform device passthrough</td>
<td>E. Auger</td>
</tr>
<tr>
<td>1</td>
<td>Dynamic sysbus device allocation support</td>
<td>A. Graf</td>
</tr>
<tr>
<td>2</td>
<td>machvirt dynamic sysbus device instantiation</td>
<td>E. Auger</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Kernel Patches</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>VFIO support for platform devices</td>
<td>A. Motakis</td>
</tr>
<tr>
<td>1</td>
<td>ARM: KVM: add irqfd support</td>
<td>E. Auger</td>
</tr>
<tr>
<td>2</td>
<td>KVM-VFIO IRQ forward control</td>
<td>E. Auger</td>
</tr>
<tr>
<td>3</td>
<td>ARM: Forwarding physical interrupts to a guest VM</td>
<td>M. Zyngier</td>
</tr>
</tbody>
</table>
Conclusion

• Main functional bricks are available for efficient KVM platform device passthrough

• Forwarded IRQ usage shows improvements on
  – Sustained IRQ rate
  – Latency
  – Bandwidth, on some patterns

• Please test and use VFIO platform
  – Start integrating your devices
  – Share issues with complex device tree nodes
  – Work ongoing on AArch64 too
Questions?
Thanks!
Backup Slides
Irqfd Standard ARM Porting

**Host**
- xgmac
- GIC
- Gen IRQ
- VFIO plat driver

**KVM (EL1)**
- Vfio_irq_handler
- Signal eventfd

**Guest**
- vGIC
- Gen IRQ
- Xgmac driver

- vIRQ = 1
- vAck
- vDeactivate

**VM switch**
- p for physical, v for virtual
- Eoi = end-of-interrupt
- DIR = deactivate

**Flowchart**
- IRQ = 1
- Ack
- mask
- deactivate
- Direct Access to IRQ status reg
- IRQ = 0
- Maintenance IRQ
- unmask
- Virqfd Resamplefd handler
- Signal resamplefd
- vIRQ = 1
- vIRQ = 0

**Notes**
- KVM (EL2)
- vIRQ = 1
- VM switch
Forwarded IRQ

Host
- xgmac
- GIC
- Gen IRQ
- VFIQ plat driver
- KVM (EL1)
- IRQ = 1
- Ack
- EOI (prio drop)

KVM (EL2)
- vGIC
- Gen IRQ
- Xgmac driver
- vIRQ = 1
- vAck
- vEOI+vDIR

Guest
- VM switch

Direct Access to IRQ status reg
- pIRQ = 0
- GIC Auto DIR

time