PCI Express I/O Virtualization Howto

Authors: Yu Zhao <yu.zhao@intel.com>
Donald Dutile <ddutile@redhat.com>

Overview

What is SR-IOV

Single Root I/O Virtualization (SR-IOV) is a PCI Express Extended capability which makes one physical device appear as multiple virtual devices. The physical device is referred to as Physical Function (PF) while the virtual devices are referred to as Virtual Functions (VF). Allocation of the VF can be dynamically controlled by the PF via registers encapsulated in the capability. By default, this feature is not enabled and the PF behaves as traditional PCIe device. Once it's turned on, each VF's PCI configuration space can be accessed by its own Bus, Device and Function Number (Routing ID). And each VF also has PCI Memory Space, which is used to map its register set. VF device driver operates on the register set so it can be functional and appear as a real existing PCI device.

User Guide

How can I enable SR-IOV capability

Multiple methods are available for SR-IOV enablement. In the first method, the device driver (PF driver) will control the enabling and disabling of the capability via API provided by SR-IOV core. If the hardware has SR-IOV capability, loading its PF driver would enable it and all VFs associated with the PF. Some PF drivers require a module parameter to be set to determine the number of VFs to enable. In the second method, a write to the sysfs file sriov_numvfs will enable and disable the VFs associated with a PCIe PF. This method enables per-PF, VF enable/disable values versus the first method, which applies to all PFs of the same device. Additionally, the PCI SRIOV core support ensures that enable/disable operations are valid to reduce duplication in multiple drivers for the same checks, e.g., check numvfs == 0 if enabling VFs, ensure numvfs <= totalvfs. The second method is the recommended method for new/future VF devices.

How can I use the Virtual Functions

The VF is treated as hot-plugged PCI devices in the kernel, so they should be able to work in the same way as real PCI devices. The VF requires device driver that is same as a normal PCI device's.

Developer Guide

SR-IOV API

To enable SR-IOV capability:

a. For the first method, in the driver:

```
int pci_enable_sriov(struct pci_dev *dev, int nr_virtfn);
```

'nr_virtfn' is number of VFs to be enabled.

b. For the second method, from sysfs:

```
echo 'nr_virtfn' > \
/sys/bus/pci/devices/<DOMAIN:BUS:DEVICE.FUNCTION>/sriov_numvfs
```

To disable SR-IOV capability:

a. For the first method, in the driver:

```
void pci_disable_sriov(struct pci_dev *dev);
```

b. For the second method, from sysfs:

```
echo  0 > \
/sys/bus/pci/devices/<DOMAIN:BUS:DEVICE.FUNCTION>/sriov_numvfs
```

To enable auto probing VFs by a compatible driver on the host, run command below before enabling SR-IOV capabilities. This is the default behavior.

```
echo 1 > \
/sys/bus/pci/devices/<DOMAIN:BUS:DEVICE.FUNCTION>/sriov_drivers_autoprobe
```

To disable auto probing VFs by a compatible driver on the host, run command below before enabling SR-IOV capabilities. Updating this entry will not affect VFs which are already probed.

```
echo  0 > \
/sys/bus/pci/devices/<DOMAIN:BUS:DEVICE.FUNCTION>/sriov_drivers_autoprobe
```

Usage example

Following piece of code illustrates the usage of the SR-IOV API.

```
static int dev_probe(struct pci_dev *dev, const struct pci_device_id *id)
{
        pci_enable_sriov(dev, NR_VIRTFN);

        ...

        return 0;
}

static void dev_remove(struct pci_dev *dev)
{
        pci_disable_sriov(dev);

        ...
}

static int dev_suspend(struct pci_dev *dev, pm_message_t state)
{
        ...

        return 0;
}

static int dev_resume(struct pci_dev *dev)
{
        ...

        return 0;
}

static void dev_shutdown(struct pci_dev *dev)
{
        ...
}

static int dev_sriov_configure(struct pci_dev *dev, int numvfs)
{
        if (numvfs > 0) {
                ...
                pci_enable_sriov(dev, numvfs);
                ...
                return numvfs;
        }
        if (numvfs == 0) {
                ....
                pci_disable_sriov(dev);
                ...
                return 0;
        }
}

static struct pci_driver dev_driver = {
        .name =         "SR-IOV Physical Function driver",
        .id_table =     dev_id_table,
        .probe =        dev_probe,
        .remove =       dev_remove,
        .suspend =      dev_suspend,
        .resume =       dev_resume,
        .shutdown =     dev_shutdown,
        .sriov_configure = dev_sriov_configure,
};
```