开放数据中心委员会

[编号 ODCC-2023-0500D]

# 存储部件特性 SR-IOV

# 测试规范

开放数据中心委员会

2023-09 发布

# 版权声明

# 编写组

**项目经理：**

张华 深圳忆联信息系统有限公司

**工作组长：**

郭亮 中国信息通信研究院

**贡献专家：**

李军 深圳忆联信息系统有限公司

王钦 深圳忆联信息系统有限公司

豆坤 三星半导体有限公司

洪源基 三星半导体有限公司

趟炳圭 三星半导体有限公司

王晋强 深圳大普微电子科技有限公司

曹仲 深圳大普微电子科技有限公司

崔志勇 深圳大普微电子科技有限公司

尹作刚 浪潮电子信息产业股份有限公司

秦文政 浪潮电子信息产业股份有限公司

殷军博 浪潮电子信息产业股份有限公司

黄福帅 联想（北京）有限公司

吴福磊 联想（北京）有限公司

李朝兰 北京忆恒创源科技股份有限公司

崔志焕 SK 海力士半导体（中国）有限公司

金泰贵 SK 海力士半导体（中国）有限公司

李敏沃 SK 海力士半导体（中国）有限公司

杜松 英韧科技(上海)有限公司

石健 英韧科技(上海)有限公司

# 目 录

版权声明 ..... I

编写组 ..... II

一、 引言 ..... 1

（一） SR-IOV 特性介绍 ..... 1

（二） 编写目的 ..... 2

（三） 缩略语 ..... 2

二、 SR-IOV 测试概要 ..... 3

（一） 测试范围 ..... 3

（二） 测试用例概括 ..... 3

1. 基本功能测试 ..... 3

2. 协议一致性测试 ..... 3

3. 兼容性测试 ..... 4

4. 应用场景测试 ..... 4

5. 性能测试 ..... 4

三、 SR-IOV 测试环境 ..... 4

（一） 宿主机虚拟机环境搭建（以 CentOS 7.9 为例） ..... 5

1. 确认 CPU 是否支持虚拟化 ..... 5

2. 宿主机开启虚拟化，并开启 SR-IOV 支持 ..... 5

3. 修改系统内核启动参数，重启系统，使其开启 IOMMU，支持 PCI Pass-through ..... 6

4. Disable autoprobe action (it is supported if kernel version is 4.12 or above, which can ignore) ..... 7

5. 在宿主机上安装虚拟化软件 ..... 7

6. 安装虚拟机 ..... 7
（二） SSD SR-IOV 配置 ..... 7
1. 使用 lspci -nn 命令找到待分配的 PCI 设备，使能 SR-IOV ..... 7
2. 用 nvme-cli 命令查询 SSD 的辅助控制器 ..... 7
3. 创建 VF 对应的 NS（namespace） ..... 8
4. 辅助控制器的资源分配和上线 ..... 9
5. 配置 SR-IOV QoS ..... 11
（三） 虚拟机分配 VF ..... 11
1. 通过图形界面方式分配 VF ..... 12
2. 通过命令行直通给虚拟机 ..... 12
（四） SR-IOV 大规模环境部署 ..... 13
四、 SR-IOV 基本功能测试 ..... 14
（一） 测试需求 ..... 14
（二） 测试方法和测试标准 ..... 15
1. SR-IOV 功能开关测试 ..... 15
2. VF 数量规格验证 ..... 15
3. 队列资源分配测试 ..... 16
4. 中断资源分配测试 ..... 17
5. namespace 资源分配验证 ..... 17
6. 特殊场景功能验证 ..... 18
五、 SR-IOV 协议一致性测试 ..... 19
（一） 测试需求 ..... 19
1. PCIe 虚拟化相关协议内容 ..... 19
2. NVMe 虚拟化相关协议内容 ..... 19

（二） 协议虚拟化部分覆盖测试 ..... 20
1. Identify Controller ..... 20
2. Primary Controller Capabilities ..... 22
3. Secondary Controller list ..... 26
4. Virtualization Management command ..... 28
5. Virtualization Enhancements ..... 31
（三） VF NVME 命令集覆盖测试 ..... 34
1. 测试方法概述 ..... 34
2. 测试项举例 ..... 34
六、 SR-IOV 兼容性测试 ..... 35
（一） 测试需求 ..... 35
（二） 测试项举例 ..... 36
七、 SR-IOV 应用场景测试 ..... 37
（一） 测试需求 ..... 37
（二） 测试方法和测试标准 ..... 37
1. 虚拟机 shutdown 测试 ..... 37
2. 虚拟机 reboot 测试 ..... 37
3. 虚拟机 force reset ..... 38
4. 虚拟机 force off ..... 38
5. 虚拟机添加、删除 PCIe 设备(卸载、加载驱动) ..... 39
6. VF Offline, 对并列 VF 影响（PF 不挂载 NS） ..... 39
7. VF Offline，对 PF、并列 VF 的影响（PF 挂载 NS） ..... 39
8. VF FLR，对并列 VF 的影响 ..... 40
9. PF 控制器复位 ..... 40

10. Host 服务器 reboot ..... 41
11. Host 服务器 shutdown ..... 41
12. Host 服务器掉电 ..... 41
13. FIO 顺序读写带宽稳定性 ..... 42
14. FIO 随机读写 iops 稳定性 ..... 42
八、 SR-IOV 性能测试 ..... 42
（一） 测试需求 ..... 42
（二） 测试方法和测试标准 ..... 43
1. 验证 SR-IOV 模式与非 SR-IOV 模式下的性能 ..... 43
2. 验证多 VF 性能是否均衡 ..... 47
3. 验证 QoS 功能有效性 ..... 52
4. QoS 精度验证 ..... 55
5. 长时间 IO 读写压力测试 ..... 56

# 存储部件特性 SR-IOV 测试规范

## 一、 引言

### （一）SR-IOV 特性介绍

SR-IOV 是由 PCI-SIG 组织定义的 PCIe 规范的扩展规范，是一种基于硬件的虚拟化解决方案，可提高性能和可伸缩性。SR-IOV 标准允许在虚拟机之间高效共享 PCIe 设备，可以获得能够与本机性能媲美的 I/O 性能。

SR-IOV 协议引入了两种类型功能的概念：物理功能 PF 和虚拟功能 VF。PF 用于管理 SR-IOV 功能。PF 拥有完全配置资源，可以用于配置或控制 PCIe 设备。VF 是与 PF 关联的一种功能，是一种轻量级 PCIe 功能，可以与物理功能以及与同一物理功能关联的其他 VF 共享一个或多个物理资源。VF 仅允许拥有用于其自身行为的配置资源。

在不支持 SR-IOV 的虚拟化场景下，各个虚拟机均通过代理（VMM）实现对 SSD 资源的访问，VMM 实现地址转换、IO 调度等功能。在高 IOPS 场景下 VMM 开销较高甚至成为瓶颈。在 SR-IOV 环境下，所有虚拟机均对应 SSD 的一个 VF，依托 SSD 提供的多 namespace 功能实现对共享 SSD 资源的直接访问，一方面虚拟机通过 VF 直接读写 SSD 物理空间，提升 IO 性能，提升 IOPS 与降低时延，另一方面虚拟机的 IO 请求 Bypass VMM，进一步降低了处理器开销。

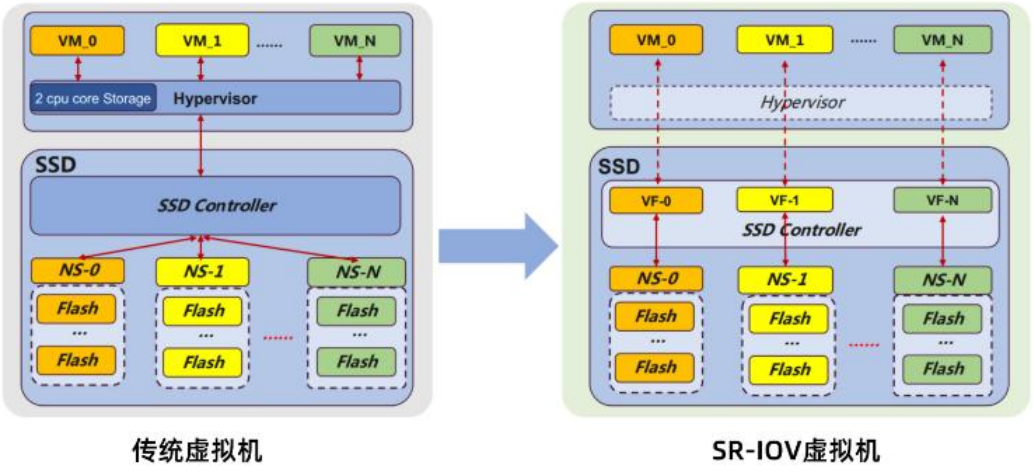


图 1 传统虚拟机与 SR-IOV 虚拟机对比

SSD 盘 SR-IOV 特性主要是应对云上大盘拆小卖的场景。鉴于 SSD 盘的规格越来越大，拆小卖的需求长时间存在。相比之下，SPDK 软件方案实现了大盘拆小的能力，但是以消耗服务器上的 CPU 为代价。因此，SR-IOV 技术在这个维度上是很有价值的。

## （二）编写目的

针对 SR-IOV 特性测试，从目前调研看，各个互联网厂家都有独自的一套测试方案，测试重点各有不同，缺乏统一的操作规范和判别标准。

本测试规范基于互联网厂家以及 SSD 厂家测试用例，制定了企业级固态硬盘 SR-IOV 特性的基准测试方法，适用于企业级固态硬盘 SR-IOV 特性的基准测试和选型测试。

## （三）缩略语

下列缩略语适用于本标准。

| 简称 | 英文 | 中文 |
|---|---|---|
| SR-IOV | Single Root IO Virtualization | 单根 I/O 虚拟化 |

| | | |
|---|---|---|
| FLR | Function Level Reset | 功能层复位 |
| QoS | Quality of Service | 服务质量 |
| SVM | AMD Secure Virtual Machine | AMD 的虚拟化技术 |
| VMX | Virtual-Machine Extensions Intel | 虚拟化技术 |
| PCIe | Peripheral Component Interconnect Express | 快速外设组件互连 |
| IOMMU | Input/Output Memory Management Unit | 输入输出的内存管理单元 |

# 二、 SR-IOV 测试概要

## （一）测试范围

支持 SR-IOV 特性的 SSD 在产品规格和特性支持力度方面存在多种差异。本测试规范旨在聚焦于典型、具有代表性的场景，同时也会对其他规格和特性做相关描述。

- 聚焦在单端口盘
- 聚焦在单 PF 场景
- 聚焦在非共享 namespace 场景

## （二）测试用例概括

测试规范会从五方面展开：

### 1. 基本功能测试

包括 SRIOV 功能开关验证；VF 数量规格验证；队列资源分配验证；中断资源分配验证；namespace 资源分配验证；特殊场景功能验证。

### 2. 协议一致性测试

包括 PCIe 虚拟化相关协议验证；NVMe 虚拟化相关协议验证；VF NVME 命令集验证。

### 3. 兼容性测试

涉及不同的服务器，操作系统，驱动和内核版本验证，异常下，盘片行为验证。

### 4. 应用场景测试

覆盖虚拟机应用过程中的典型场景。包括虚拟机 shutdown、reboot、force reset、force off，移除添加 PCI 设备，VF offline online，虚拟机加、卸载驱动，VF FLR，服务器重启、服务器上、下电等。

### 5. 性能测试

包含 SR-IOV 模式与非 SR-IOV 模式下的性能验证；多 VF 性能是否均衡验证；QOS 功能有效性验证；QoS 精度验证；长时间 IO 读写压力验证。

## 三、 SR-IOV 测试环境

表 1 SR-IOV 软硬件测试环境要求

| 序号 | 名称 | 数量 | 配置要求 |
|---|---|---|---|
| 1 | 宿主机 | 1 | 英特尔或 AMD 处理器<br>支持 IOMMU<br>支持 VMX/SVM 和 SR-IOV 规范<br>操作系统推荐 linux |
| 2 | SSD | 1 | 支持 SRIOV |

| 序号 | 软件名称 | 版本要求 |
|---|---|---|
| 1 | nvme-cli | 推荐采用最新版本（version>=1.91） |
| 2 | fio | 推荐采用最新版本 |
| 3 | qemu-kvm | |

## （一）宿主机虚拟机环境搭建（以 CentOS 7.9 为例）

### 1. 确认 CPU 是否支持虚拟化

KVM 需要 CPU 硬件支持虚拟化，不同厂商虚拟化技术命令不同（如 Intel CPU 的虚拟化技术叫 vmx，AMD CPU 的虚拟化技术叫 svm）。执行如下命令，如果有结果显示就说明 CPU 支持虚拟化。

# sudo egrep -o '(vmx|svm)' /proc/cpuinfo

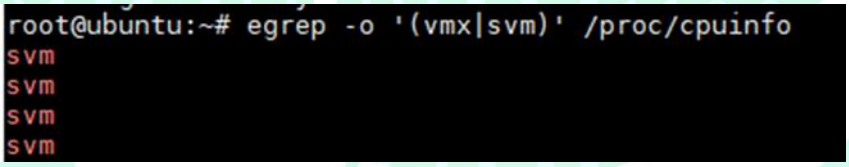


图 2 确认 CPU 是否支持虚拟化

### 2. 宿主机开启虚拟化，并开启 SR-IOV 支持

在服务器 BIOS 中，通常需要设置下面选项2：

- 使能 SRIOV 选项
- 使能 Virtualization Technology 选项
- 使能 PCI 64-Bit Resource Allocation

以 DELL R750 服务器设置为例，设置方法如下:

[1] nvme-cli 工具在 1.9 版本‘add identify secondary controller list’

[2] 服务器设置中如果没有对应选项，可忽略该项。

启动服务器时按住 F2 进入 BIOS 模式，以下各项设置成 ENABLE，保存退出

System BIOS->Processor Settings->virtualization technology

System BIOS->Processor Settings->iommu_support

System BIOS->Processor Settings->integrated devices->Sriov global enable

3. 修改系统内核启动参数，重启系统，使其开启 IOMMU，支持 PCI Pass-through

- 修改/etc/default/grub, 调整 GRUB_CMDLINE_LINUX 内容，例如 Intel，AMD 的配置。

a. Intel 的配置：GRUB_CMDLINE_LINUX=“intel_iommu=on iommu=pt”，

b. AMD 的配置：GRUB_CMDLINE_LINUX=“amd_iommu=on iommu=pt”

- 使用$ grub-mkconfig -o /boot/grub/grub.cfg 重新创建引导，重启系统，使得配置生效。

- 执行 $ dmesg | grep -E "DMAR|IOMMU" 查看 IOMMU 是否正确使能。

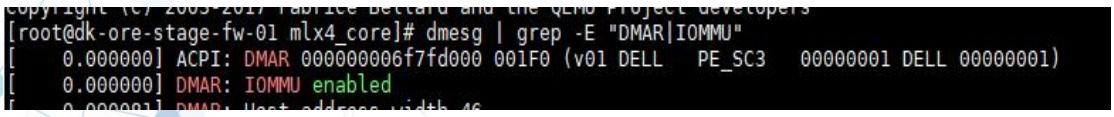


图 3 查看 IOMMU 是否正确使能

4. Disable autoprobe action (it is supported if kernel version is 4.12 or above, which can ignore)

# echo 0 > /sys/bus/pci/device/${bdf_pf}/sriov_drivers_autoprobe

在启用 SR-IOV 功能之前运行上面命令，通过主机上的兼容驱动程序禁用自动探测 VF。更新此条目不会影响已探测的 VF。

5. 在宿主机上安装虚拟化软件

推荐使用 qemu-kvm。

6. 安装虚拟机

采用命令行方式或者图形界面方式，根据需求在 KVM 环境下安装一个或多个虚拟机，虚拟机 OS 可以根据业务需求选择。宿主机和虚拟机可以采用网桥方式配置网络或者在宿主机通过 virsh console 连接虚拟机。

## （二）SSD SR-IOV 配置

1. 使用 lspci -nn 命令找到待分配的 PCI 设备，使能 SR-IOV

- 查询 SSD:

# lspci -nn | grep Non

- 使能硬盘的 SRIOV, n 代表 VF 个数：

# echo n > /sys/bus/pci/devices/${bdf_pf}/sriov_numvfs

2. 用 nvme-cli 命令查询 SSD 的辅助控制器

# nvme list-secondary /dev/nvmeX3

下图是查询的辅助控制器对应的信息：初始状态都为 offline，VQ 和 VI 都为 0。

```
    SCID      : Secondary Controller Identifier : 0x0005
    PCID      : Primary Controller Identifier   : 0x0041
    SCS       : Secondary Controller State      : 0x0000 (Offline)
    VFN       : Virtual Function Number         : 0x0005
    NVQ       : Num VQ Flex Resources Assigned  : 0x0000
    NVI       : Num VI Flex Resources Assigned  : 0x0000
  SCEntry[5  ]:
```

图 4 list-secondary 辅助控制器查询信息

### 3. 创建 VF 对应的 NS（namespace）

根据 NVME 协议，PF 和 VF 可以共享命名空间 NS，如下图中的 NS F，也可以独占 NS，如 NS A, NS B 等。协议中对于共享 NS 没有做强制要求，存储部件厂家支持力度不同。

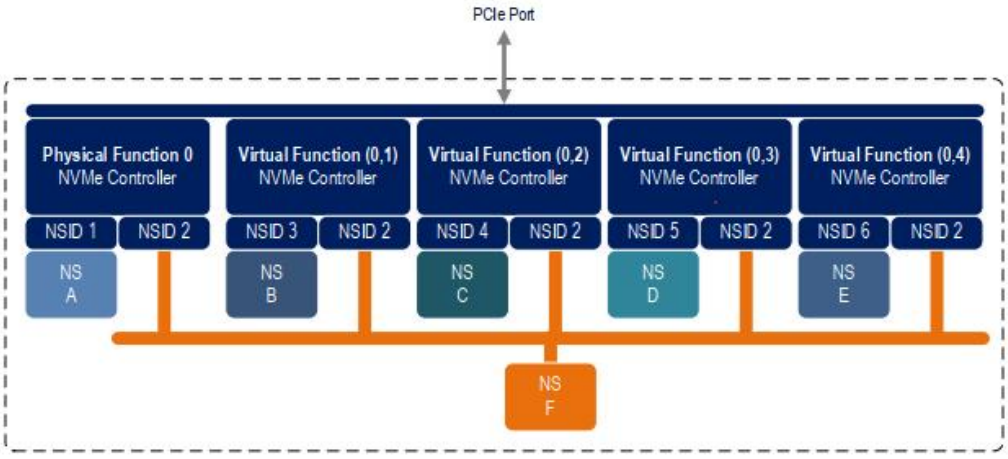


图 5 nvme 协议中 SR-IOV NS 分配示例

[3] nvmeX 中 X 根据实际环境中盘符修改

NS 的大小由命令设置。下面以配置 1 个私有的 NS 为例，如果配置多个，就重复以下步骤即可：

a. 创建 NS

# nvme create-ns /dev/nvmeX4 -s ${ns_size} -c ${ns_size} -f 0 -d 0 -m 0

b. attach NS 到辅助控制器

# nvme attach-ns /dev/nvmeX1 -n 1 -c ${辅助控制器 ID5}

**4. 辅助控制器的资源分配和上线**

对于辅助控制器的资源分配和上线，需要参照 Spec 中的参数配置，一般资源数不少于 2。下面以辅助控制器 ID 0x2, 分配 VI 和 VQ 数目 9 为例，如果需要配置多个，重复以下步骤即可。需要注意的是有的存储厂家对辅助控制器 VI 资源的数量内部设置成了固定值，不允许外部配置。

- Allocate 9 VQ resource to secondary controller 2 (CID 2)

# nvme virt-mgmt /dev/nvmeX1 -c 0x2 -r 0 -a 8 -n 0x9

nvme virt-mgmt 选项说明如下：

[ --cntlid=<NUM>, -c <NUM> ] --- Controller Identifier(CNTLID)

[ --rt=<NUM>, -r <NUM> ] --- Resource Type(RT): [0,1]

[4] nvmeX 中 X 根据实际环境中盘符修改

[5] 辅助控制器开始 ID 各个厂家定义不同，比如深圳忆联 ESSD 从 0x2 开始，三星 PM1733 从 0x1 开始

0h: VQ Resources

1h: VI Resources

[ --act=<NUM>, -a <NUM> ] --- Action(ACT): [1,7,8,9]

1h: Primary Flexible

7h: Secondary Offline

8h: Secondary Assign

9h: Secondary Online

[ --nr=<NUM>, -n <NUM> ] --- Number of Controller Resources(NR)

除了 nvme-cli 封装的命令 nvme virt-mgmt 外，也可以采用标准的 nvme 命令格式配置 VQ 资源，命令如下：

# nvme admin-passthru /dev/nvmeX6 --opcode=0x1C --cdw10=0x10008 --cdw11=${VQ_count}

- Allocate 9 VI resource to secondary controller 2(CID 2)

# nvme virt-mgmt /dev/nvmeX1 -c 0x2 -r 1 -a 8 -n 0x9

除了 nvme-cli 封装的命令 nvme virt-mgmt 外，同样也可以采用标准的 nvme 命令格式配置 VI 资源，命令如下：

# nvme admin-passthru /dev/nvmeX1 --opcode=0x1C --cdw10=0x10108 --cdw11=${VI_count}

- Issue FLR on VF （FLR 操作为可选项[7]）

[6] nvmeX 中 X 根据实际环境中盘符修改

[7] 部分 ESSD 厂家配置时，不支持 FLR 操作

# echo 1 > /sys/bus/pci/devices/${bdf_vf }/reset

- Set secondary controller online

# nvme virt-mgmt /dev/nvmeX8 -c 2 -a 9

除了 nvme-cli 封装的命令 nvme virt-mgmt 外，同样也可以采用标准的 nvme 命令格式配置 VI 资源，命令如下：

# nvme admin-passthru /dev/nvmeX1 --opcode=0x1C --cdw10=0x10009 --cdw11=0

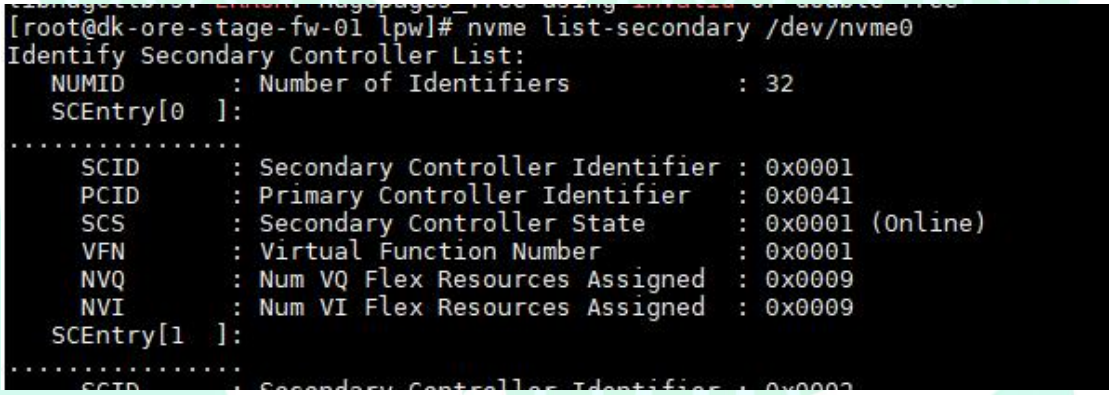


图 6 查看辅助控制器状态信息

### 5. 配置 SR-IOV QoS

SR-IOV QoS 特性协议中没有做强制要求，支持 SR-IOV QoS 的 SSD，通常会在 vendor 字段定义。具体设定方法需要根据厂家说明书配置。通常 QoS 的配置是和 VF ID 相关联。

## （三）虚拟机分配 VF

虚拟机分配 VF，可以通过两种方式，一种是图形界面方式，另一种是通过命令行直通给虚拟机。

[8] nvmeX 中 X 根据实际环境中盘符修改

### 1. 通过图形界面方式分配 VF

a. 登陆 HOST 的 virt-manager 图形界面 Virtual Machine Manager，双击已经安装的 GUEST。

b. 点击信息配置栏，选择 Add Hardware->PCI Host Device，将 VF 对应的 PCI 设备选上。

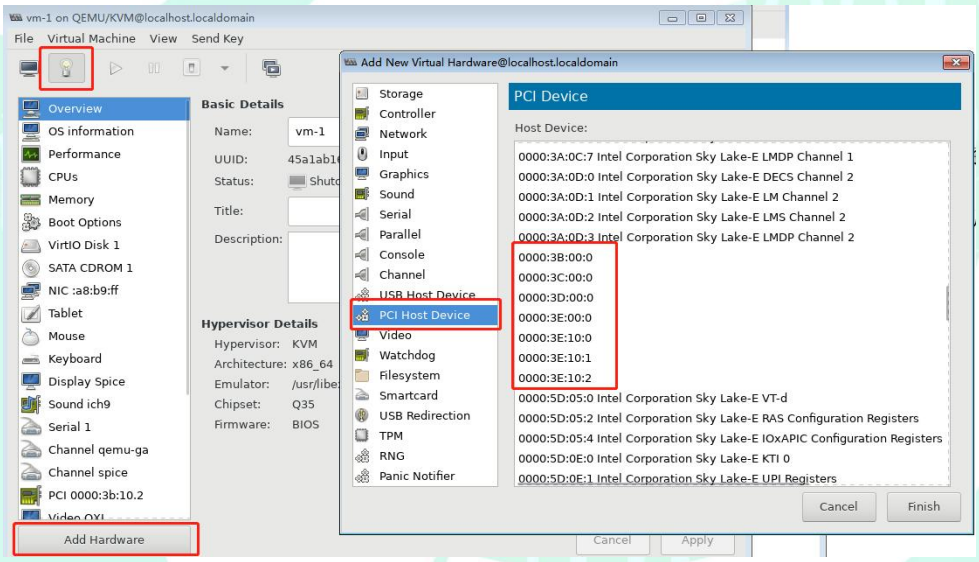


图 7 图形界面分配 VF

### 2. 通过命令行直通给虚拟机

将 VF 设备绑定到 VFIO 驱动，配置几个 VF，就将其绑定到 VFIO 驱动，下面以 VF1 为例：

a. 查看 PCI 设备的 vendor id 和 device id

```
# lspci -nn | grep Non
```

b. 加载驱动

```
# modprobe vfio
# modprobe vfio-pci
```

c. 将初始的 VF 驱动解绑

# echo ${bdf_vf } >/sys/bus/pci/devices/${bdf_vf }/driver/unbind

d. 把上一步解绑的 VF 绑定到 vfio 的驱动上

# echo ${bdf_vf } > /sys/bus/pci/drivers/vfio-pci/bind

e. 开启虚拟机，将 VF 设备通过 VFIO 的方式直通给虚拟机

执行如下命令开启虚拟机，并将 VF 设备直通给虚拟机

# qemu-system-x86_64 -enable-kvm -m ${vf_memory_size} -smp ${vf_cpu_count} ${image_path} -device vfio-pci, host=${bdf_vf}

Eg.#qemu-system-x86_64 -enable-kvm -m 2G -smp 16 /home/lpw/img1/linux.img -device vfio-pci, host=0000:86:00.1

如果在虚拟机上可以查询到 NVMe 设备，如下图所示，就说明搭建成功


图 8 虚拟机中查询 nvme 设备

## （四）SR-IOV 大规模环境部署

由于 SR-IOV 配置信息在盘片掉电、重启后不会被保存，因此在需要进行大规模环境部署时，重新为每张盘片设置 SR-IOV 配置信息将耗费大量时间和精力。

为了解决这个问题，我们可以采用配置文件的方式来记录盘片的配置信息。通过脚本来解析这些配置文件，实现对 SSD 盘片的 SR-IOV 设置。以下列出了三种场景下的配置流程：


图 9 SR-IOV 不同场景下的配置流程

## 四、 SR-IOV 基本功能测试

### （一）测试需求

SRIOV 基本功能测试，主要包含以下几方面：

- SRIOV 功能开关测试
- VF 数量规格验证
- 队列资源分配测试
- 中断资源分配验证

- namespace 资源分配验证
- 特殊场景功能验证

## （二）测试方法和测试标准

### 1. SR-IOV 功能开关测试

(1) 测试方法概述

测试注意点：

- 验证 SR-IOV 默认开/关状态在常见应用场景下是否可持久保存。

常见应用场景，包含重启服务器，盘片固件升级，盘片异常掉电，盘片控制器复位，加载、卸载驱动等，

- 如果 SSD 支持 SR-IOV 开/关状态改变，

a. 验证开/关状态切换，

b. 新状态生效后，在常见应用场景下验证是否可持久保存。

(2) 测试项举例

| | |
|---|---|
| 测试用例 | 验证SR-IOV默认开关状态可持久化保存 |
| 测试目的 | 验证 SR-IOV 默认开关状态可持久化保存 |
| 测试方法 | 测试步骤：<br>1. 查询盘片SRIOV默认开启状态<br>2. 依次执行下面场景：服务器重启，盘片热插拔，盘片控制器复位，加载、卸载nvme驱动<br>3. 步骤2中，每个场景执行后，查询盘片SR-IOV开启状态 |
| 通过准则 | 步骤 3 查询到的状态与步骤 1 默认状态一致 |

### 2. VF 数量规格验证

(1) 测试方法概述

测试注意点：

- VF 验证数量需要覆盖 1，典型业务数量和最大支持数量。
- VF 绑定虚拟机测试基本 IO 功能是否正常。

(2) 测试项举例

| 测试用例 | 验证盘片支持VF能力 |
| --- | --- |
| 测试目的 | 验证盘片支持VF能力 |
| 测试方法 | 假定SSD最多支持16个VF<br>测试步骤：<br>1. 通过配置VF使能个数为1，然后绑定虚拟机后，进行3分钟基本IO功能测试，查看是否OK；<br>2. 配置VF使能个数为最大16个，绑定虚拟机，进行3分钟基本IO功能测试，查看是否OK；<br>3. 配置VF使能个数从2-16个，并绑定虚拟机，进行3分钟基本IO功能测试，查看是否OK |
| 通过准则 | 步骤1中，VF使能1个成功，VF下IO读写均正常<br>步骤2中，VF使能16个成功，VF下IO读写均正常<br>步骤3中，VF使能2-16个成功，VF下IO读写均正常 |

### 3. 队列资源分配测试

(1) 测试方法概述

测试注意点：

- 覆盖到每个 VF 分配最小 VQ 数量场景，即 2 个队列(Admin Submission Queue x 1 + IO Submission Queue x 1)；
- 覆盖到多 VF 分配最大支持 VQ 数量场景，每个 VF 分配的 VQ 数量可以不均匀；
- PF，VF 在上述场景下，IO 读写正常

(2) 测试项举例

| 测试用例 | 验证所有VF只分配最小队列资源后能正常运行 |
| --- | --- |
| 测试目的 | 队列资源分配测试 |
| | 1. 创建8个NS； |

| | |
|---|---|
| 测试方法 | 2. 创建8个VF，每个VF分配2个队列，onlineVF，把8个NS分别挂载到8个VF；<br>3. 把VF分别挂载到不同虚拟机；<br>4. fio分别对8个VF进行64K随机读写，读比例50%，1线程64队列，持续30分钟；<br>5. 观察IO读写情况。 |
| 通过准则 | 步骤5中， VF均读写正常，无IO错误和数据不一致 |

| | |
|---|---|
| 测试用例 | 验证VF分配最大队列资源后能正常运行 |
| 测试目的 | 队列资源分配测试 |
| 测试方法 | 1. 创建8个NS；<br>2. 创建8个VF，对VF1~7每个分配2个VQ队列，VF8分配剩下所有的队列；<br>3. onlineVF，把8个NS挂载到VF8；<br>4. 把VF8挂载到虚拟机；<br>5. fio对VF8进行64K随机读写，读比例50%，1线程64队列，持续5分钟；<br>6. 观察IO读写情况。 |
| 通过准则 | 步骤6中， VF8读写正常，无IO错误和数据不一致 |

### 4. 中断资源分配测试

先确认 SSD 是否支持中断资源配置，如果支持，注意下面几点：

- 配置不同 VI 数量验证，需要从两方面考虑，一种是对应中断模式最大支持的中断数量，二是处理器对 VI 数量的最大支持力度；
- 不同 VI 资源配置下，PF，VF IO 读写正常。

### 5. namespace 资源分配验证

(1) 测试方法概述

测试注意点：

- 验证 nvme 控制器是否正确执行命名空间管理命令，并能够正常关联到二级控制器中；

- namespace 在对应 PF/VF 下，IO 读写一致性验证；
- 如果 SSD 支持 namespce 共享，验证 PF 和 VF 之间共享，VF 之间共享两种场景。

(2) 测试项举例

| 测试用例 | 验证所有VF只分配最小队列资源后能正常运行 |
|---|---|
| 测试目的 | 队列资源分配测试 |
| 测试方法 | 假定业务典型应用场景为4VF。<br>1. 按照等容量创建5个NS；<br>2. 创建4个VF，平均分配VQ、VI资源；<br>3. Online VF，把5个NS分配挂载到PF和4个VF；<br>4. 把4个VF分配挂载到4个虚拟机；<br>5. 格式化每个NS为文件系统，分别挂载到HOST和对应虚拟机测试目录下；<br>6. 在5个挂载目录下，随机产生100G文件[9]，计算随机产生文件的md5值；<br>7. 在挂载目录下，对步骤6生成的文件进行拷贝，重新计算md5，并和步骤6计算的md5值比较；<br>8. 重复步骤7，测试5次。 |
| 通过准则 | 步骤7两次md5值一致 |

### 6. 特殊场景功能验证

SSD 除了单端口盘外，还存在多种形态，比如双端口盘，加密盘。

测试注意点：

- 双端口盘 SR-IOV 特性验证，可以当作两个单端口盘验证，区别在于 namespace 大小，VQ，VI 资源数量会小于单端口盘；

[9] 文件大小，根据 NS 大小，和步骤 8 重复测试次数调整。

- 加密盘的验证，如果加密特性是针对 namespace，在验证 SR-IOV 特性中，需要把加密特性一起加入验证，如果加密特性不是针对 namespace，需要根据 SSD 具体加密实现方式，制定测试策略。

# 五、 SR-IOV 协议一致性测试

## （一）测试需求

通过 nvme-cli 工具下发指令并检查返回值来验证 SR-IOV 的协议一致性，确保待测样品能够支持协议规范内的虚拟化相关特性。

### 1. PCIe 虚拟化相关协议内容

PCIe 规范中有一章节“Single Root I/O Virtualization and Sharing”对 SR-IOV 做了协议描述。SSD 芯片验证阶段，会进行 PCIe 协议一致性验证。本测试规范针对 PCIe 虚拟化内容，只涉及到 PF SRIOV 寄存器验证。

### 2. NVMe 虚拟化相关协议内容

本文档所用 NVMe 协议基于 NVMe Spec 1.4c 版本。SR-IOV 相关协议内容包含：

- Identify Controller（5.15.2.2，NVMe Spec 1.4c）
- Primary Controller Capabilities（5.15.2.10，NVMe Spec 1.4c）
- Secondary Controller list（5.15.2.11，NVMe Spec 1.4c）
- Virtualization Management command（5.22，NVMe Spec 1.4c）
- Virtualization Enhancements（8.5，NVMe Spec 1.4c）

在 NVMe Spec 1.4c 中，虚拟化相关内容为可选实现。

## （二）协议虚拟化部分覆盖测试

### 1. Identify Controller

(1) 协议解读

用于识别 NVMe SSD 是否支持 SR-IOV。nvme-cli 工具通过 id-ctrl 命令可以返回协议中定义的 Identify Controller 数据结构，该数据结构描述了设备的能力和支持的特性。数据结构中的 Byte 76，bit 2 用于支持是否关联了 SR-IOV 的 Virtual Function。

对应的 nvme-cli 命令为“nvme id-ctrl”。

**Figure 251: Identify – Identify Controller Data Structure**

| Bytes | O/M [1] | Description |
|---|---|---|
| | | **Controller Capabilities and Features** |
| 01:00 | M | **PCI Vendor ID (VID):** Contains the company vendor identifier that is assigned by the PCI SIG. This is the same value as reported in the ID register in section 2.1.1. |
| 03:02 | M | **PCI Subsystem Vendor ID (SSVID):** Contains the company vendor identifier that is assigned by the PCI SIG for the subsystem. This is the same value as reported in the SS register in section 2.1.17. |
| 23:04 | M | **Serial Number (SN):** Contains the serial number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.10 for unique identifier requirements. Refer to section 1.5 for ASCII string requirements. |
| 63:24 | M | **Model Number (MN):** Contains the model number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.10 for unique identifier requirements. Refer to section 1.5 for ASCII string requirements. |
| 71:64 | M | **Firmware Revision (FR):** Contains the currently active firmware revision, as an ASCII string, for the NVM subsystem. This is the same revision information that may be retrieved with the Get Log Page command, refer to section 5.14.1.3. |
| 72 | M | **Recommended Arbitration Burst (RAB):** This is the recommended Arbitration Burst size. The value is in commands and is reported as a power of two (2^*n*). This is the same units as the Arbitration Burst size. Refer to section 4.13. |
| 75:73 | M | **IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor. The OUI shall be a valid IEEE/RAC assigned identifier that may be registered at http://standards.ieee.org/develop/regauth/oui/public.html. |
| 76 | O | **Controller Multi-Path I/O and Namespace Sharing Capabilities (CMIC):** This field specifies multi-path I/O and namespace sharing capabilities of the controller and NVM subsystem.<br>Bits 7:4 are reserved.<br>Bit 3 if set to '1', then the NVM subsystem supports Asymmetric Namespace Access Reporting (refer to section 8.20). If cleared to '0', then the NVM subsystem does not support Asymmetric Namespace Access Reporting.<br>Bit 2 if set to '1', then the controller is associated with an SR-IOV Virtual Function. If cleared to '0', then the controller is associated with a PCI Function or a Fabrics connection.<br>Bit 1 if set to '1', then the NVM subsystem may contain two or more controllers. If cleared to '0', then the NVM subsystem contains only a single controller. As described in section 1.4.1, an NVM subsystem that contains multiple controllers may be used by multiple hosts, or may provide multiple paths for a single host.<br>Bit 0 if set to '1', then the NVM subsystem may contain more than one NVM subsystem port. If cleared to '0', then the NVM subsystem contains only a single NVM subsystem port. |

图 10 nvme identify controller-Byte76 定义

(2) 命令示例

示例命令 1 - /dev/nvme0 是 Linux 系统下看到的 Primary Controller 的字符设备名。

```
# nvme id-ctrl /dev/nvme0 -H | grep -A 10 oacs
oacs      : 0x9e
  [9:9] : 0     Get LBA Status Capability Not Supported
  [8:8] : 0     Doorbell Buffer Config Not Supported
  [7:7] : 0x1   Virtualization Management Supported
```

```
[6:6] : 0    NVMe-MI Send and Receive Not Supported
[5:5] : 0    Directives Not Supported
[4:4] : 0x1  Device Self-test Supported
[3:3] : 0x1  NS Management and Attachment Supported
[2:2] : 0x1  FW Commit and Download Supported
[1:1] : 0x1  Format NVM Supported
[0:0] : 0    Security Send and Receive Not Supported
```

对于支持 SR-IOV 的盘输出应当为 Supported。

示例命令 2 - /dev/nvme2 是 Linux 系统下配置 VF 后看到的 Secondary Controller 的字符设备名。

```
# nvme id-ctrl /dev/nvme2 | grep cmic

cmic      : 0x6
```

对于 VF 设备 cmic 的 bit 2 应当置为 1。

### 2. Primary Controller Capabilities

(1) 协议解读

用于检查 Primary Controller 的信息和相关资源。nvme-cli 工具通过命令 primary-ctrl-caps 返回协议中定义的数据结构，该结构描述了 Primary Controller 相关的信息以及可用的 VQ/VI 资源。

根据 NVMe Spec 1.4c 定义，VQ/VI 资源分为私有（Private）和可变（Flexible）两种。私有资源为 controller 私有，不可分配；可变资源可以分配给不同 PF/VF。

对应的 nvme-cli 命令为“nvme primary-ctrl-caps”。

**Figure 256: Identify – Primary Controller Capabilities Structure**

| Bytes | Description |
|---|---|
| 01:00 | **Controller Identifier (CNTLID):** This field indicates the Controller Identifier of the primary controller. |
| 03:02 | **Port Identifier (PORTID):** This field indicates the Port Identifier of the NVM subsystem port associated with the primary controller. The Port Identifier for a PCI Express Port is the same as the Port Number field in Link Capabilities Register in the PCI Express Capability structure (refer to section 2.5.6). |
| 04 | **Controller Resource Types (CRT):** This field indicates the controller resources types supported. If a primary controller supports a controller resource type, then all associated secondary controllers shall support that controller resource type.<br>Bits 7:2 are reserved.<br>Bit 1 if set to '1', then VI Resources are supported. Bit 1 if cleared to '0', then VI Resources are not supported. Refer to section 8.5.2.<br>Bit 0 if set to '1', then VQ Resources are supported. Bit 0 if cleared to '0', then VQ Resources are not supported. Refer to section 8.5.1. |
| 31:05 | Reserved |
| 35:32 | **VQ Resources Flexible Total (VQFRT):** This field indicates the total number of VQ Flexible Resources for the primary and its secondary controllers. |
| 39:36 | **VQ Resources Flexible Assigned (VQRFA):** This field indicates the total number of VQ Flexible Resources Assigned to the associated secondary controllers. |
| 41:40 | **VQ Resources Flexible Allocated to Primary (VQRFAP):** This field indicates the total number of VQ Flexible Resources currently allocated to the primary controller. This value may change after a Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from '1' to '0') if a new value was set using the Virtualization Management command. The default value of this field is implementation specific. |
| 43:42 | **VQ Resources Private Total (VQPRT):** This field indicates the total number of VQ Private Resources for the primary controller. |
| 45:44 | **VQ Resources Flexible Secondary Maximum (VQFRSM):** This field indicates the maximum number of VQ Flexible Resources that may be assigned to a secondary controller. |
| 47:46 | **VQ Flexible Resource Preferred Granularity (VQGRAN):** This field indicates the preferred granularity of assigning and removing VQ Flexible Resources. Assigning and removing VQ Resources in this granularity minimizes any wasted internal implementation resources. |
| 63:48 | Reserved |
| 67:64 | **VI Resources Flexible Total (VIFRT):** This field indicates the total number of VI Flexible Resources for the primary and its secondary controllers. |
| 71:68 | **VI Resources Flexible Assigned (VIRFA):** This field indicates the total number of VI Flexible Resources Assigned to the associated secondary controllers. |
| 73:72 | **VI Resources Flexible Allocated to Primary (VIRFAP):** This field indicates the total number of VI Flexible Resources currently allocated to the primary controller. This value may change after a Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from '1' to '0') if a new value was set using the Virtualization Management command. The default value of this field is implementation specific. |
| 75:74 | **VI Resources Private Total (VIPRT):** This field indicates the total number of VI Private Resources for the primary controller. |
| 77:76 | **VI Resources Flexible Secondary Maximum (VIFRSM):** This field indicates the maximum number of VI Flexible Resources that may be assigned to a secondary controller. |
| 79:78 | **VI Flexible Resource Preferred Granularity (VIGRAN):** This field indicates the preferred granularity of assigning and removing VI Flexible Resources. Assigning and removing VI Resources in this granularity minimizes any wasted internal implementation resources. |
| 4095:80 | Reserved |

图 11 nvme identify-主控能力结构体

(2) 命令示例

示例命令如下：

```
# nvme primary-ctrl-caps /dev/nvme0 -H
NVME Identify Primary Controller Capabilities:
cntlid    : 0
portid    : 0
crt       : 0x3
  [1:1] 0x1     VI Resources are supported
  [0:0] 0x1     VQ Resources are supported
vqfrt     : 118
vqrfa     : 0
vqrfap    : 0
vqprt     : 14
vqfrsm    : 14
vqgran    : 1
vifrt     : 118
virfa     : 0
virfap    : 0
viprt     : 14
vifrsm    : 14
vigran    : 1
```

其中字段说明如下：

a. Cntlid 是 PF controller ID。

b. Crt 的值，两个 bit 分别表示是否支持 VQ/VI 资源。如果 VQ/VI 不支持配置，对应 bit 位为 0。是否支持配置 VQ/VI 资源取决于设备实现。

c. Vqfrt 表示总共可以用来分配给 PF/VF 的 VQ 可变资源数。该数字不代表当前可用资源数。

d. Vqfra 表示已经分配给 VF 的 VQ 可变资源数。

e. Vqrfap 表示当前已经分配给 PF 的 VQ 可变资源数。

f. Vqprt 表示当前已经分配给 PF 的 VQ 私有资源数。

g. Vqfrsm 表示总共可以分配给单个 VF 的 VQ 可变资源数。该数字不代表当前可用资源数。

h. Vqgran 表示分配/移除 VQ 资源数的最小粒度。按最小粒度分配可以最小化内部资源浪费。

i. Vifrt 表示总共可以用来分配给 PF/VF 的 VI 可变资源数。该数字不代表当前可用资源数。

j. Vufra 表示已经分配给 VF 的 VI 可变资源数。

k. Vurfap 表示当前已经分配给 PF 的 VI 可变资源数。

l. Viprt 表示当前已经分配给 PF 的 VI 私有资源数。

m. Vifrsm 表示总共可以分配给单个 VF 的 VI 可变资源数。该数字不代表当前可用资源数。

n. Vigran 表示分配/移除 VI 资源数的最小粒度。按最小粒度分配可以最小化内部资源浪费。

在对 PF/VF 做 VQ/VI 配置前，执行前述命令并查看可用资源类别及数量，与设备规格说明对比判断资源数是否正确；配置后，执行前述命令并与配置 PF/VF 时分配的 VQ/VI 资源做对比，以判断资源记录是否正确。

### 3. Secondary Controller list

(1) 协议解读

用于查看当前支持的所有 VF（Secondary Controller）的状态和资源配置情况。协议定义的数据结构最多返回 127 个 VF 信息。返回的条目包含处于 online 或 offline 状态的 VF。

对应 nvme-cli 的命令为“nvme list-secondary”。

**Figure 257: Secondary Controller List**

| Bytes | Description |
|---|---|
| 00 | **Number of Identifiers:** This field indicates the number of Secondary Controller Entries in the list. There are up to 127 entries in the list. A value of 0h indicates there are no entries in the list. |
| 31:01 | Reserved |
| 63:32 | **SC Entry 0:** This field contains the first Secondary Controller Entry in the list, if present. |
| 95:64 | **SC Entry 1:** This field contains the second Secondary Controller Entry in the list, if present. |
| ... | ... |
| (N*32+63): (N*32+32) | **SC Entry N:** This field contains the N+1 Secondary Controller Entry in the list, if present. |

**Figure 258: Secondary Controller Entry**

| Bytes | Description |
|---|---|
| 01:00 | **Secondary Controller Identifier (SCID):** This field indicates the Controller Identifier of the secondary controller described by this entry. |
| 03:02 | **Primary Controller Identifier (PCID):** This field indicates the Controller Identifier of the associated primary controller. |
| 04 | **Secondary Controller State (SCS):** This field indicates the state of the secondary controller.<br>Bits 7:1 are reserved.<br>Bit 0 if set to '1', then the controller is in the Online state. Bit 0 if cleared to '0', then the controller is in the Offline state. |
| 07:05 | Reserved |
| 09:08 | **Virtual Function Number (VFN):** If the secondary controller is an SR-IOV VF, this field indicates its VF Number, where VF Number > 0, and VF Number is no larger than the total number of VFs indicated by the TotalVFs register (refer to Single Root I/O Virtualization and Sharing Specification) in the PF's SR-IOV Extended Capability structure. If the secondary controller is not an SR-IOV VF, then this field is cleared to 0h. |
| 11:10 | **Number of VQ Flexible Resources Assigned (NVQ):** This field indicates the number of VQ Flexible Resources currently assigned to the indicated secondary controller. |
| 13:12 | **Number of VI Flexible Resources Assigned (NVI):** This field indicates the number of VI Flexible Resources currently assigned to the indicated secondary controller. |
| 31:14 | Reserved |

图 12 nvme secondary controller 结构体定义

(2) 命令示例

示例命令：

```
# nvme list-secondary /dev/nvme0 -c 3
Identify Secondary Controller List:
  NUMID       : Number of Identifiers          : 8
  SCEntry[3  ]:
.....
    SCID      : Secondary Controller Identifier : 0x0004
    PCID      : Primary Controller Identifier   : 0x0000
    SCS       : Secondary Controller State      : 0x0001 (Online)
    VFN       : Virtual Function Number         : 0x0004
    NVQ       : Num VQ Flex Resources Assigned  : 0x000e
    NVI       : Num VI Flex Resources Assigned  : 0x000e
  SCEntry[4  ]:
.....
    SCID      : Secondary Controller Identifier : 0x0005
    PCID      : Primary Controller Identifier   : 0x0000
    SCS       : Secondary Controller State      : 0x0000 (Offline)
    VFN       : Virtual Function Number         : 0x0005
    NVQ       : Num VQ Flex Resources Assigned  : 0x0000
    NVI       : Num VI Flex Resources Assigned  : 0x0000
  SCEntry[5  ]:
.....
    SCID      : Secondary Controller Identifier : 0x0006
    PCID      : Primary Controller Identifier   : 0x0000
    SCS       : Secondary Controller State      : 0x0000 (Offline)
    VFN       : Virtual Function Number         : 0x0006
    NVQ       : Num VQ Flex Resources Assigned  : 0x0000
    NVI       : Num VI Flex Resources Assigned  : 0x0000
  SCEntry[6  ]:
.....
```

```
  SCID     : Secondary Controller Identifier : 0x0007
  PCID     : Primary Controller Identifier   : 0x0000
  SCS      : Secondary Controller State      : 0x0000 (Offline)
  VFN      : Virtual Function Number         : 0x0007
  NVQ      : Num VQ Flex Resources Assigned  : 0x0000
  NVI      : Num VI Flex Resources Assigned  : 0x0000
SCEntry[7  ]:
.....
  SCID     : Secondary Controller Identifier : 0x0008
  PCID     : Primary Controller Identifier   : 0x0000
  SCS      : Secondary Controller State      : 0x0000 (Offline)
  VFN      : Virtual Function Number         : 0x0008
  NVQ      : Num VQ Flex Resources Assigned  : 0x0000
  NVI      : Num VI Flex Resources Assigned  : 0x0000
```

注意此处的命令行参数“-c 3”，表示返回结果中的 controller ID 从 3 开始，controller ID 小于 3 的不显示。Control ID 是结果输出中的“SCID”对应的数字，不是 SCEntry 中的数字。

#### 4. Virtualization Management command

(1) 协议解读

虚拟化管理命令仅在 Primary Controller（PF）上支持。该命令用于：

a. 修改 primary controller 的可变资源。

b. 为 secondary controller 分配可变资源。。

c. 修改 secondary controller 的 online/offline 状态。

对应的 nvme-cli 命令为“nvme virt-mgmt”。

注意 Secondary Controller 处于 Offline 状态才能对它分配 VQ/VI 可变资源。

当命令返回错误时，请参考 5.2.2.2 Primary Controller Capabilities 确认支持的特性，例如 VQ/VI 是否支持，是否有可用可变资源用于分配给 PF 或 VF。

**Figure 329: Virtualization Management – Command Specific Status Values**

| Value | Description |
|---|---|
| 1Fh | **Invalid Controller Identifier:** An invalid Controller Identifier was specified. |
| 20h | **Invalid Secondary Controller State:** The action requested for the secondary controller is invalid based on the current state of the secondary controller and its primary controller. |
| 21h | **Invalid Number of Controller Resources:** The specified number of Flexible Resources is invalid (e.g., the Number of Controller Resources (NR) is greater than VQ Resources Flexible Total (VQFRT) (refer to Figure 256), the Number of Controller Resources (NR) is greater than VQ Resources Flexible Secondary Maximum (VQFRSM) (refer to Figure 256)). |
| 22h | **Invalid Resource Identifier:** At least one of the specified resource identifiers was invalid (e.g., the Number of Controller Resources (NR) is greater than the number of remaining available flexible resources). |

图 13 nvme 虚拟化管理命令返回状态

虚拟化管理命令的返回值为 4 种：

a. Invalid Controller Identifier – 无效的 Controller ID。命令中使用了不合法的 Controller ID，如 Secondary Controller ID 不存在。请使用 list-secondary 命令获取正确的 Secondary Controller ID。

b. Invalid Secondary Controller State – Secondary Controller 状态异常。对 Secondary Controller 做管理操作时它处于不正确的状态。例如对处于 Online 状态的 Secondary Controller 配置 VQ/VI 资源。

c. Invalid Number of Controller Resources – 命令中给出的资源数量无效。例如尝试分配的 VQ 可变资源数大于可用 VQ 可变资源数。

d. Invalid Resource Identifier – 命令中给出的资源标识至少有一个是无效的。例如尝试分配的可变资源数大于剩余可分配的可变资源数。

(2) 命令示例

示例命令 1 – 为 PF 分配 VQ 可变资源：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -r 0 -n 8 -a 1

success, Number of Controller Resources Modified (NRM):0
```

示例命令 2 – 为 PF 分配 VI 可变资源：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -r 1 -n 8 -a 1

success, Number of Controller Resources Modified (NRM):0
```

示例命令 3 – 为 VF 分配 VQ 可变资源：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -r 0 -n 8 -a 8

success, Number of Controller Resources Modified (NRM):0
```

示例命令 4 – 为分配 VI 可变资源：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -r 1 -n 8 -a 8

success, Number of Controller Resources Modified (NRM):0
```

示例命令 5 – 设为 Online：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -a 9

success, Number of Controller Resources Modified (NRM):0
```

➢ 当 VF 已经处于 Online 状态时，再次设置 Online 状态会返回成功。

➢ VF 正常工作需要 2 个 VQ 资源，一个用于 admin，一个用于 I/O。如果分配 VQ 资源数少于 2 对 VF 做 Online 操作会失败。

示例命令 6 – 设为 Offline：

```
# nvme virt-mgmt /dev/nvme0 -c 2 -a 7

success, Number of Controller Resources Modified (NRM):0
```

➢ 当 VF 已经处于 Offline 状态时，再次设置 Offline 状态会返回成功。

### 5. Virtualization Enhancements

在虚拟化环境中，NVM 子系统可以通过多个控制器为物理机和虚拟机提供直接 I/O 访问通路。NVM 子系统可由多个 Primary Controller 和 Secondary Controller 组成，每个 Secondary Controller 均依附于 1 个 Primary Controller，并通过 Primary Controller 来为 Secondary Controller 动态分配资源。

通过向 Primary Controller 发送虚拟化管理命令可以为每个控制器分配或移除控制器资源（Controller Resources）。可分配控制器资源分为两种：

a. Virtual Queue Resource（VQ 资源）。每个资源用于管理 1 对 SQ/CQ（Submission Queue/completion Queue）。

b. Virtual Interrupt Resource （VI 资源）。每个资源用于管理 1 个中断向量。

控制器资源从可分配性上看分为两种，可变资源（Flexible Resources）和私有资源（Private Resources）。私有资源是固定分配给 Primary Controller 或 Secondary Controller 的资源，不能通过虚拟化管理命令配置。而可变资源 Primary Controller 和 Secondary Controller 则可以用虚拟化管理命令配置。

Primary Controller 和 Secondary Controller 是否支持私有资源或可变资源，可以通过 Primary Controller Capabilities 中的数据结构查看。

**Figure 463: Controller Resource Allocation**

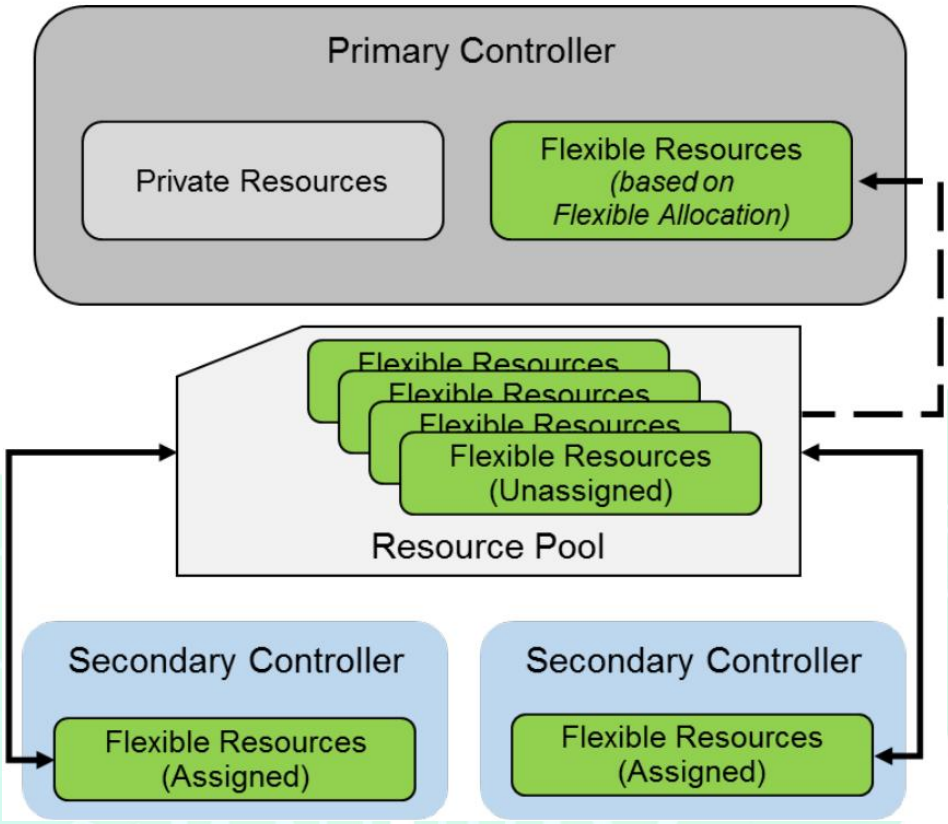


图 14 nvme 控制器资源分配

厂商可以选择为 Primary Controller 和 Secondary Controller 实现 NVMe Spec 1.4c 中的所有的，除明确指出只支持 Primary Controller 外的特性，但是这里建议对于特权操作（NVMe Spec 1.4c 中 7.14 小节所提）仅在 Primary Controller 上实现，以避免通过 Secondary Controller 执行特权操作带来的意外，例如通过 Secondary Controller 1 用 format 命令格式化关联在 Secondary Controller 2 上的命名空间。

PCI-SIG 组织的 Single Root I/O Virtualization and Sharing 规范扩展了 PCI Express 的定义，允许多个系统镜像（System Images，SIs）（如虚拟机）共享 PCI 设备硬件资源。SR-IOV 的优势是 Hypervisor

不需要参与 I/O 操作，SI 直接访问硬件资源，这在一些虚拟化场景中极大地提升了存储的性能。

Physical Function（PF）是一种支持 SR-IOV 能力的 PCI Express Function，它同时支持一个或多个依附于该 PF 的 Virtual Function（VF）。PF/VF 可以选择实现通过 Multi-path I/O 共享底层 NVM 子系统，以及共享命名空间的特性。

SR-IOV VF 如果设置了 NVM Express Class Code，应当实现兼容 NVM Express Controller，这是为了保证在非虚拟化环境中不修改系统镜像也能够正常使用。例如在 Linux 虚拟机中使用 VF 时，需要确保 VF 兼容 NVMe 标准，Linux 中的 nvme 驱动不需要修改就能直接通过 VF 访问 I/O。

### （三）VF NVME 命令集覆盖测试

#### 1. 测试方法概述

测试注意点：

- 命令测试范围需要根据 SSD 厂家提供的 VF 支持命令清单制定；
- VF 命令集需要在虚拟机下验证，VF 数量可以设置为 1。

#### 2. 测试项举例

| 测试用例 | 虚拟机上下发identify 测试 |
| --- | --- |
| 测试目的 | 验证SR-IOV功能遵从NVMe协议 |
| | 测试步骤：<br>1. 创建1个NS；<br>2. 创建1VF，并分配VQ队列，onlineVF，把NS挂载到VF； |

| | |
|---|---|
| 测试方法 | 3. 把VF挂载到虚拟机；<br>4. 在虚拟机上对盘片下发Identify命令（遍历以下全部所列Identify命令：Identify 0x1、Identify 0x2、Identify 0x3、Identify 0x11、Identify 0x12）[10] |
| 通过准则 | 步骤4中，命令下发成功 |

| | |
|---|---|
| 测试用例 | 虚拟机上下发get log page 测试 |
| 测试目的 | 验证SR-IOV功能遵从NVMe协议 |
| 测试方法 | 测试步骤：<br>1. 创建1个NS；<br>2. 创建1VF，并分配VQ队列，onlineVF，把NS挂载到VF；<br>3. 把VF挂载到虚拟机；<br>4. 在虚拟机上对盘片下发Get log page命令（包含以下全部所列Get log page命令：Get log page 0x1、Get log page 0x2、Get log page 0x3、Get log page 0x4、Get log page0x81）[11] |
| 通过准则 | 步骤4中，命令下发成功 |

# 六、 SR-IOV 兼容性测试

## （一）测试需求

本章主要从两方面进行验证：

- SR-IOV 兼容性验证，涉及不同的服务器，操作系统，驱动和内核版本。
- 异常下，盘片行为验证。

测试注意点：

[10] identify ID 设置范围根据产品规格书 VF 支持命令集修改

[11] Get log page ID 设置范围根据产品规格书 VF 支持命令集修改

- 服务器选择，可以覆盖目前主流厂家服务器，例如超聚变，浪潮，联想，戴尔等，以业务指定服务器为测试重点；
- 系统选择，可以在每个主流体系 OS 中选择一到两个验证，例如，RedHat 体系中的 RedHat，Centos；SUSE 体系中的 SUSE/Leap；Debian 体系中的 debian/ubuntu，OpenCloudOS 体系中的 OpenCloudOS，以业务指定系统为测试重点；
- 驱动验证中，除了 OS 自带的 nvme 驱动外，如果存储部件厂家有自己编写的驱动，也需要验证到，以业务指定驱动版本为测试重点；
- 内核版本，选取稳定版本内核，以业务指定的内核版本为测试重点；
- 兼容性验证，主要是关注在不同环境配置下，功能验证正常，IO 读写正常；
- 异常下 SSD 行为验证，可以结合第七章应用场景测试展开。

### （二）测试项举例

| 测试用例 | 操作系统兼容性测试 |
| --- | --- |
| 测试目的 | 验证不同操作系统下，SSD SR-IOV环境能正常搭建 |
| 测试方法 | 1. 服务器安装ubuntu系统；<br>2. 创建8个NS；<br>3. 创建8个VF，并平均分配VQ、VI资源，onlineVF，把8个NS分别挂载到8个VF；<br>4. 把VF分别挂载到不同虚拟机；<br>5. fio分别对8个VF进行64K随机读写，读比例50%，1线程64队列，持续30分钟；<br>6. 观察IO读写情况；<br>7. 更换服务器操作系统为centos，重复步骤3~步骤6 |
| | 1. SR-IOV在不同操作系统上搭建正常， |

| | |
|---|---|
| 通过准则 | 2. Step5：VF均读写正常，无IO错误和数据不一致。 |

# 七、 SR-IOV 应用场景测试

## （一）测试需求

本章主要描述了客户在应用 SR-IOV 特性，搭建虚拟机场景时，所遇到的 VM 层面，服务器 Host 层面的正常、非正常操作，进行的配置、恢复，以及异常操作下配置、复位的验证。

测试涵盖客户在使用过程中遇到的各种场景。验证 SR-IOV 设备在实际使用中的稳定性。

## （二）测试方法和测试标准

### 1. 虚拟机 shutdown 测试

| | |
|---|---|
| 测试用例 | 虚拟机shutdown测试 |
| 测试目的 | 虚拟机应用场景中，shutdown虚拟机，对其他VF虚拟机以及测试虚拟机的影响 |
| 测试方法 | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. 两台虚拟机中，分别启动fio随机读写测试。<br>5. fio测试过程中，将VF1所在虚拟机进行shutdown 。<br>6. 观察VF2虚拟机IO是否正常。<br>7. 重新运行VF1所在的虚拟机，并重新启动fio IO读写 。<br>8. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step5：VF1所在的虚拟机正常shutdown。<br>2. Step7：VF1正常启动，fio测试无异常<br>3. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 2. 虚拟机 reboot 测试

| | |
|---|---|
| 测试用例 | 虚拟机reboot测试 |
| 测试目的 | 虚拟机应用场景中，reboot虚拟机，对其他VF虚 |

| | |
|---|---|
| | 拟机以及测试虚拟机的影响 |
| 测试方法 | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. 两台虚拟机中，分别启动fio随机读写测试。<br>5. fio测试过程中，将VF1所在虚拟机进行reboot。<br>6. 观察VF2虚拟机IO是否正常。<br>7. 重新运行VF1所在的虚拟机，并重新启动fio IO读写 。<br>8. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step5：VF1所在的虚拟机正常reboot。<br>2. Step7：VF1正常启动，fio测试无异常<br>3. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 3. 虚拟机 force reset

| 测试用例 | 虚拟机force reset测试 |
|---|---|
| 测试目的 | 虚拟机应用场景中，force reset虚拟机，对其他VF虚拟机以及测试虚拟机的影响 |
| 测试方法 | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. 两台虚拟机中，分别启动fio随机读写测试。<br>5. fio测试过程中，将VF1所在虚拟机进行force reset。<br>6. 观察VF2虚拟机IO是否正常。<br>7. 重新运行VF1所在的虚拟机，并重新启动fio IO读写 。<br>8. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step7：VF1正常启动，fio测试无异常<br>2. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 4. 虚拟机 force off

| 测试用例 | 虚拟机force off测试 |
|---|---|
| 测试目的 | 虚拟机应用场景中，force off虚拟机，对其他VF虚拟机以及测试虚拟机的影响 |
| | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. 两台虚拟机中，分别启动fio随机读写测试。 |

| | |
|---|---|
| 测试方法 | 5. fio测试过程中，将VF1所在虚拟机进行force off。<br>6. 观察VF2虚拟机IO是否正常。<br>7. 重新运行VF1所在的虚拟机，并重新启动fio IO读写 。<br>8. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step7：VF1正常启动，fio测试无异常<br>2. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 5. 虚拟机添加、删除 PCIe 设备(卸载、加载驱动)

| | |
|---|---|
| 测试用例 | 虚拟机添加、删除PCIe设备(卸载、加载驱动) |
| 测试目的 | 虚拟机应用场景中，添加、删除PCIe设备，对其他虚拟机的影响 |
| 测试方法 | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. 两台虚拟机中，VF2启动fio随机读写测试。<br>5. fio测试过程中，将VF1所在虚拟机设备驱动卸载，然后加载。。<br>6. 观察VF2虚拟机IO是否正常。<br>7. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step5：VF1设备驱动可以正常卸载后加载<br>2. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 6. VF Offline，对并列 VF 影响（PF 不挂载 NS）

| | |
|---|---|
| 测试用例 | VF Offline, 对并列VF影响（PF不挂载NS） |
| 测试目的 | 虚拟机应用场景中，VF Offline，对其他虚拟机的影响 |
| 测试方法 | 1. 创建2个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。<br>3. 把VF分别挂载到两台虚拟机。<br>4. fio测试过程中，将VF1所在虚拟机，对VF1进行offline以及Online操作。<br>5. 观察VF2虚拟机IO是否正常。<br>6. 观察其它虚拟机IO是否正常。 |
| 通过准则 | 1. Step4：VF1可以正常Offline以及Online<br>2. 测试过程中，VF2 虚拟机的fio测试无异常，无性能跌落。 |

### 7. VF Offline，对 PF、并列 VF 的影响（PF 挂载 NS）

| | |
|---|---|
| 测试用例 | VF Offline操作，对PF、并列VF的影响 |
| 测试目的 | 虚拟机应用场景中，VF Offline操作，对PF、并列VF的影响 |
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，将VF1反复进行Offline以及Online操作。<br>6. 观察PF，VF2的IO是否正常。 |
| 通过准则 | 1. Step5：VF1可以正常Offline以及Online<br>2. 测试过程中，PF，VF2的fio测试无异常，无性能跌落。 |

### 8. VF FLR，对并列 VF 的影响

| | |
|---|---|
| 测试用例 | VF FLR操作，对VF的影响 |
| 测试目的 | 虚拟机应用场景中，VF FLR操作，对其他VF的影响 |
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，将VF1进行FLR操作。<br>6. 观察其他VF的IO是否正常。 |
| 通过准则 | 1. Step5：VF进行FLR操作可以在Sepc宣称时间内完成。<br>2. 测试过程中，其他VF的fio测试无异常，无性能跌落。 |

### 9. PF 控制器复位

| | |
|---|---|
| 测试用例 | PF控制器复位测试 |
| 测试目的 | 虚拟机应用场景中，PF 控制器复位 |
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，在Host对控制器进行复位操作。<br>6. 观察Host以及虚拟机是否正常识别盘。 |

| | |
|---|---|
| 通过准则 | 1. Step5：控制器可以正常复位。<br>2. Step6：复位完成后，Host能识别盘，虚拟机无法识别。 |

## 10. Host 服务器 reboot

| | |
|---|---|
| 测试用例 | Host服务器复位测试 |
| 测试目的 | 虚拟机应用场景中，Host服务器复位 |
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，对Host服务器reboot操作。<br>6. 观察Host以及虚拟机是否正常识别盘。 |
| 通过准则 | 1. Step5：Host reboot正常。<br>2. Step6：reboot完成后，Host能识别盘，虚拟机无法识别。 |

## 11. Host 服务器 shutdown

| | |
|---|---|
| 测试用例 | Host服务器shutdown测试 |
| 测试目的 | 虚拟机应用场景中，Host服务器shutdown |
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，对Host服务器shutdown操作。<br>6. 重启启动服务器，观察Host以及虚拟机是否正常识别盘。 |
| 通过准则 | 1. Step5：Host shutdown正常。<br>2. Step6：重新启动完成后，Host能识别盘，虚拟机无法识别。 |

## 12. Host 服务器掉电

| | |
|---|---|
| 测试用例 | Host服务器掉电测试 |
| 测试目的 | 虚拟机应用场景中，Host服务器强制掉电 |

| | |
|---|---|
| 测试方法 | 1. 创建3个NS。<br>2. 创建2个VF。并把2个NS分别挂载到2个VF。另一个NS挂载到PF<br>3. 把VF分别挂载到虚拟机。<br>4. 3个NS启动fio随机读写测试。<br>5. fio测试过程中，对Host服务器断电。<br>6. 重启上电并启动服务器，观察Host以及虚拟机是否正常识别盘。 |
| 通过准则 | 1. Step6：重新启动完成后，Host能识别盘，虚拟机无法识别。 |

### 13. FIO 顺序读写带宽稳定性

| | |
|---|---|
| 测试用例 | FIO顺序读写稳定性 |
| 测试目的 | 虚拟机应用场景中，虚拟盘的读写稳定性 |
| 测试方法 | 1. 创建1个NS。<br>2. 创建1个VF。<br>3. 把VF挂载到虚拟机。<br>4. 启动fio进行bs128k长时间顺序读写测试。 |
| 通过准则 | Step4：顺序读写带宽稳定，latency稳定。 |

### 14. FIO 随机读写 iops 稳定性

| | |
|---|---|
| 测试用例 | FIO随机读写稳定性 |
| 测试目的 | 虚拟机应用场景中，虚拟盘的读写稳定性 |
| 测试方法 | 1. 创建1个NS。<br>2. 创建1个VF。<br>3. 把VF挂载到虚拟机。<br>4. 启动fio进行bs4K长时间随机读写测试。 |
| 通过准则 | Step4：随机读写iops稳定，latency稳定。 |

## 八、 SR-IOV 性能测试

### （一）测试需求

性能测试用例主要包含以下几个方面：

a. 验证 SR-IOV 模式与非 SR-IOV 模式下的性能；

b. 验证多 VF 性能是否均衡；

c. 验证 QOS 功能有效性；

d. 验证 QoS 精度；

e. 验证长时间 IO 读写压力。

## （二）测试方法和测试标准

### 1. 验证 SR-IOV 模式与非 SR-IOV 模式下的性能

(1) 测试方法概述

可以从两个维度做组合验证：

a. SR-IOV 模式（单 VF）和非 SR-IOV 模式（PF）；

b. 服务器单盘场景和服务器多盘并发场景。

验证点包括：

a. 每张盘性能是否可以达到规格书性能[12]；

b. PF 和单个 VF 的性能差。

测试注意事项：

a. IO 模型采用规格书中规定的模型，例如 128K 顺序读、写，4K 随机读、写，随机读、写延时等，模型需要 SSD 在稳态下进行；

b. 多盘并发测试，可以覆盖跨 socket 场景，测试盘数根据实际业务需求确定；

c. VF 资源（VQ，VI 数量，namespace）采用最大值。

(2) 测试项举例

| | |
|---|---|
| 测试用例 | 非SR-IOV 模式和SR-IOV 模式下，单盘PF和VF的顺序性能测试 |
| 测试目的 | 在非SR-IOV和 SR-IOV应用场景中，验证单盘PF和VF的顺序读写性能是否达到产品规格书的标称 |

[12] 多盘并发场景，如果达不到规格书性能，需要进一步确认是服务器限制还是盘限制

| | |
|---|---|
| | 值并对比单盘VF和PF性能差距 |
| 测试方法 | 1. 擦除SSD，对SSD进行precondition，使SSD达到稳态<br>2. 设置block size 为：128K<br>3. 按照规格书推荐设置QD和numjobs<br>4. 设置读写比例为：100%read、70%read30%write、50%read50%write、30%read70%write、100%write13<br>5. 设置测试时间为10 minutes<br>6. 重复2-5（针对不同的IO 模型循环进行顺序读写测试，直至所有IO 模型测试完毕）<br>7. 统计不同的IO 模型PF throughput<br>8. 设置VF, NS为: VF=1，NS=1，分配全部VI、VQ资源和全部NS空间<br>9. 重复2-5（针对不同的IO 模型循环进行顺序读写测试，直至所有IO模型测试完毕）<br>10. 统计不同IO模型VF throughput<br>BW Consistency = ([BW in the 99.9th percentile slowest 1-second interval]÷[Average BW])*100% |
| 通过准则 | 1. 不同IO模型下的PF throughput不低于Spec标称值<br>2. 不同IO模型下的VF throughput与PF throughput 性能差距在10%以内 |

| | |
|---|---|
| 测试用例 | 非SR-IOV 模式和SR-IOV 模式下，单盘PF和VF的随机性能测试 |
| 测试目的 | 在非SR-IOV和 SR-IOV应用场景中，验证单盘PF和VF的随机性能是否达到产品规格书的标称值，并对比单盘VF和PF性能差距 |
| 测试方法 | 1. 擦除SSD，对SSD进行precondition操作，使SSD达到稳态<br>2. 设置block size 为：4K<br>3. 按照规格书推荐设置QD和numjobs<br>4. 设置读写比例为：100%read、70%read30%write、50%read50%write、30%read70%write、100%write14 |

[13] IO 模型组合可以按照产品规格书调整

[14] IO 模型组合可以按照产品规格书调整

| | |
|---|---|
| | 5. 设置测试时间为10 minutes<br>6. 重复2-5（针对不同的 IO 模型循环进行随机读写测试，直至所有IO 模型测试完毕）<br>7. 统计不同的IO 模型的PF IOPS<br>8. 设置VF, NS为: VF=1，NS=1，分配全部VI、VQ资源和全部NS空间<br>9. 重复2-5（针对不同的IO 模型循环进行随机读写测试，直至所有IO模型测试完毕）<br>10. 统计不同IO 模型 VF IOPS<br>IOPS Consistency = ([IOPS in the 99.9th percentile slowest 1-second interval]÷[Average IOPS])*100% |
| 通过准则 | 1. 不同的IO模型的PF IOPS不低于Spec标称值<br>2. 不同IO模型下的VF throughput与PF throughput 性能差距在10%以内 |

| | |
|---|---|
| 测试用例 | 非SR-IOV 模式和SR-IOV 模式下，多盘并发时PF和VF的顺序性能测试 |
| 测试目的 | 在非SR-IOV和SRIOV应用场景中，验证多盘并发下，各个盘PF和VF的顺序读写性能是否都达到产品规格书的标称值，并对比多盘并发时VF和PF性能差距 |
| 测试方法 | 1. 擦除SSD，对各个SSD进行precondition操作，并使SSD达到稳态<br>2. 设置block size 为：128K<br>3. 按照规格书推荐设置QD和numjobs<br>4. 设置读写比例为：100%read、70%read30%write、50%read50%write、30%read70%write、100%write15<br>5. 设置测试时间为10 minutes<br>6. 重复2-5（针对不同的IO 模型进行多盘并发顺序读写测试，直至所有IO 模型测试完毕）<br>7. 统计各个SSD在不同IO 模型下的throughput<br>8. 对各个SSD设置VF, NS为: VF=1，NS=1，分配全部VI、VQ资源和全部NS空间，每个盘VF 可以通过 numa 绑定到不同CPU核上<br>9. 重复2-5 (针对不同的IO 模型进行多盘并发顺序读写测试，直至所有IO模型测试完毕）<br>10. 统计各个SSD在不同IO 模型下的VF throughput |

[15] IO 模型组合可以按照产品规格书调整

| | |
|---|---|
| | BW Consistency = ([BW in the 99.9th percentile slowest 1-second interval]÷[Average BW])*100% |
| 通过准则 | 1. 多盘（盘数不多于业务需求）并发测试下，各个SSD的PF throughput 与 Spec 标称值差距在业务允许范围内。<br>2. 不同IO模型下的各个盘VF throughput与对应SSD PF throughput 性能差距在业务允许范围内。 |

| | |
|---|---|
| 测试用例 | 非SR-IOV 模式和SR-IOV 模式下，多盘并发时PF和VF的随机性能测试 |
| 测试目的 | 在非SR-IOV和SRIOV应用场景中，验证多盘并发下，各个盘PF和VF的随机读写性能是否都达到产品规格书的标称值，并对比多盘并发时VF和PF性能差距 |
| 测试方法 | 1. 擦除SSD，对各个SSD进行precondition操作，并使SSD达到稳态<br>2. 设置block size 为：4K<br>3. 按照规格书推荐设置QD和numjobs<br>4. 设置读写比例为：100%read、70%read30%write、50%read50%write、30%read70%write、100%write16<br>5. 设置测试时间为10 minutes<br>6. 重复2-5（针对不同的IO模型进行多盘并发测试，直至所有IO模型测试完毕）<br>7. 统计各个盘在不同IO模型下的IOPS<br>8. 对各个SSD设置 VF, NS为: VF=1，NS=1，分配全部VI、VQ资源和全部NS空间，每个盘VF 可以通过 numa 绑定到不同CPU核上<br>9. 重复2-5(针对不同的IO模型进行多盘并发随机读写测试，直至所有IO模型测试完毕）<br>10. 统计各个盘在不同IO模型下的VF IOPS<br>IOPS Consistency = ([IOPS in the 99.9th percentile slowest 1-second interval]÷[Average IOPS])*100% |
| 通过准则 | 1. 多盘（盘数不多于业务需求）并发测试下，各个盘的PF IOPS与Spec 标称值差距在业务允许范围内。<br>2. 不同IO模型下的各个盘VF IOPS与对应SSD PF throughput性能差距在业务允许范围内。 |

[16] IO 模型组合可以按照产品规格书调整

### 2. 验证多 VF 性能是否均衡

(1) 测试方法概述

可以通过下面几个维度进行组合验证：

a. 均衡业务测试和非均衡业务测试

是指在各个 VF 间，通过对 IO 模型指定，进行均衡业务分配和非均衡业务分配。

b. 非均衡业务场景下，大压力 IO 和小压力 IO

通过对 VF 间，调整 IO 负载压力，验证不同 IO 压力下，各个 VF 的性能表现。

c. 标准 namespace 容量分配和非标准 namespace 容量分配

标准 namespace 容量分配可以将整个盘的容量按照 VF 数量等分，非标准 namespace 容量分配是指各个 VF 之间分配的 namespace 容量不相等，例如采用递增容量分配方式。

d. VF 数量分配

除了单 VF 和最大的 VF 数量外，采用典型的 2VF，4VF，8VF 分配场景。

e. 如果 SSD 支持 QoS 配置，VF 间 QoS 按照总带宽性能均分和非均分

VF 间 QoS 限制，验证可以采用按照容量比例划分，当 VF 间等分容量时，可以采用等分总带宽的方式限制，如果 VF 间容量不均分，QoS 采用按照容量比例划分。

测试中各个 VF 下 IO 模型选择，包含顺序读、写，随机读、写，混合读写等。读写比例可以包含：100%read、70%read30%write、50%read50%write、30%read70%write、100%write 等。

测试注意点：

a. 均衡业务测试下，如果需要比较各个 VF 性能差异，VF 间需要分配相同队列数量和相同容量测试；

b. 每个虚机可以采用透传一个 VF 方式，避免虚机对 VF 间性能的影响；

c. 非对称模型，可以采用单 numa 虚机方式，减小环境对 VF 间性能影响；

d. 除了 IO 负载测试外，验证可以覆盖部分 VF 内运行 IO 负载，对另一些 VF 执行操作验证，例如管理命令下发，VF 的卸载，挂载操作等。模拟可能出现的线上数据流与控制流同时存在的场景。



测试预期：

a. 均衡业务测试下，各 VF 之间在相同 IO 模型下性能差异，最大值和最小值相差百分比需要在业务允许范围内，同时如果存储部

件支持 QoS 功能，可以比较在 QoS 功能打开关闭两种场景下，性能和时延的前后差异；

b. 均衡业务测试下，各 VF 之间采用产品规格书负载模型，验证多 VF 总带宽，总 IOPS 和各 VF 时延与规格书的性能差异，差异百分比需要在业务允许范围内；

c. 非均衡业务测试下，对于多 VF 中相同 IO 负载的 VF 间性能数据差异，需要在业务允许范围内；

d. 数据流与控制流同时存在的场景，控制操作对 IO 没有显著影响。

(2) 测试项举例

| | |
|---|---|
| 测试用例 | 多VF场景，均衡负载下各VF顺序读写性能一致性 |
| 测试目的 | 验证在多VF场景，均衡负载下，各VF顺序读写性能是否无明显差距。 |
| 测试方法 | 测试步骤：<br>1. 擦除SSD<br>2. 设置VF，NS为：VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO顺序读写模型设置block size为128K，QD为64，numjobs为117<br>5. 并行测试4个VF，4个VF均设置为sequential read only。<br>6. 测试时间为10minute，记录4个VF的读带宽。<br>7. 并行测试4个VF，4个VF均设置为sequential write only。<br>8. 测试时间为10minute，记录4个VF的写带宽。 |
| 通过准则 | 1. 四个VF读写性能应一致，差异在10%以内。 |

[17] QD，numjobs 可以按照规格书修改

| | |
|---|---|
| 测试用例 | 多VF场景，均衡负载下各VF随机读写性能一致性 |
| 测试目的 | 验证在多VF场景，均衡负载下，各VF随机读写性能是否无明显差距。 |
| 测试方法 | 测试步骤：<br>1. 擦除SSD<br>2. 设置VF，NS为：VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为4K，QD为64，numjobs为1618<br>5. 并行测试4个VF，4个VF均设置为randread only。<br>6. 测试时间为10minute，记录4个VF的读IOPS。<br>7. 并行测试4个VF，4个VF均设置为randwrite only。<br>8. 测试时间为10minute，记录4个VF的写IOPS。 |
| 通过准则 | 1. 四个VF读写性能应一致，差异在10%以内。 |

| | |
|---|---|
| 测试用例 | 多VF场景，非均衡负载下各VF顺序读写性能一致性 |
| 测试目的 | 验证在多VF场景下，各VF为非均衡读写负载下，顺序读写性能是否一致。 |
| 测试方法 | 测试步骤：<br>1. 擦除SSD<br>2. 设置VF，NS为：VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为128K，QD为64，numjobs为1<br>5. 并行测试4个VF，其中，VF1，VF3设置为sequential read only，VF2，VF4设置为sequential write only。<br>6. 测试时间为10minute，记录4个VF的读写带 |

[18] QD，numjobs 可以按照规格书修改

| | |
|---|---|
| | 宽。 |
| 通过准则 | 1. 每组测试结果，VF1，VF3，以及VF2，VF4的读写性能差异应该在业务允许范围内。 |

| | |
|---|---|
| 测试用例 | 多VF场景，非均衡负载下各VF随机读写性能一致性 |
| 测试目的 | 验证在多VF场景下，各VF为非均衡读写负载下，随机读写性能是否一致。 |
| 测试方法 | 测试步骤：<br>1. 擦除SSD<br>2. 设置VF，NS为：VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为4K，QD为64，numjobs为16.<br>5. 并行测试4个VF，其中，VF1，VF3设置为randread only，VF2，VF4设置为randwrite only。<br>6. 测试时间为10minute，记录4个VF的读写IOPS。 |
| 通过准则 | 1. 每组测试结果，VF1，VF3，以及VF2，VF4的读写性能差异应该在业务允许范围内。 |

| | |
|---|---|
| 测试用例 | 多VF场景，管理命令对读写性能一致性影响 |
| 测试目的 | 验证在多VF场景下，管理命令对VF读写性能一致性的影响。 |
| 测试方法 | 测试步骤：<br>1. 擦除SSD<br>2. 设置VF，NS为：VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为128K，QD为64，numjobs为1.<br>5. 并行测试2个VF，其中，VF1，VF3设置为sequential read only，VF2，VF4重复进行任意管理命令下发。<br>6. 测试时间为10minute，记录VF1及VF3读带宽。 |

| | |
|---|---|
| | 7. 并行测试2个VF，其中，VF1，VF3设置为sequential write only，VF2，VF4重复进行任意管理命令下发。<br>8. 测试时间为10minute，记录VF1及VF3写带宽。<br>9. FIO读写模型设置block size为4K，QD为64，numjobs为16.<br>10. 并行测试2个VF，其中，VF1，VF3设置为random read only，VF2，VF4重复进行任意管理命令下发。<br>11. 测试时间为10minute，记录VF1及VF3读IOPS。<br>12. 并行测试2个VF，其中，VF1，VF3设置为random write only，VF2，VF4重复进行任意管理命令下发。<br>13. 测试时间为10minute，记录VF1及VF3写IOPS。 |
| 通过准则 | 1. 每组测试结果，VF1，VF3的读写性能应一致，差异在10%以内。 |

### 3. 验证 QoS 功能有效性

(1) 测试方法概述

先确认 SSD 是否支持 QoS 功能，如果支持，可以通过下面几个场景验证：

a. 验证不同 VF 配置的流量配置不同时，QOS 功能有效性；

b. 验证其他 VF QoS 改变时，QoS 功能的有效性；

c. 验证其他 VF 状态变化，QoS 功能的有效性；

d. 当 QoS 功能支持掉电保存时，验证其有效性；

e. 典型 QoS 长时间 IO 读写压力测试。

测试注意点：

a. 验证 QoS 功能有效性时，除了覆盖单 VF，最大 VF，还需要验证到业务实际使用的 VF 数量场景，

b. 不同厂家 QoS 支持力度不同，例如总带宽限制，单独读带宽限制，单独写带宽限制，总 IOPS 限制等等，验证中需要针对产品支持力度，分别验证。

测试预期：

QoS 开启情况下，实测最高值和 QoS 设置值之间差异，需要满足产品规格书中精度定义。

(2) 测试项举例

| | |
|---|---|
| 测试用例 | SR-IOV 模式下，单盘单VF QoS功能有效性验证 |
| 测试目的 | 验证单盘单VF下，QoS配置不同时，QoS功能有效性。 |
| 测试方法 | 1. 擦除SSD<br>2. 设置VF，NS为: VF=1，NS=1，将NS attach到VF上<br>3. 对VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为128K，QD为64，numjobs为119<br>5. 打开VF的QoS功能，QoS分别限制为总读带宽的[10%, 30%, 50%, 70%, 100%]<br>6. 设置workload为sequential read only， 对VF进行顺序读测试。<br>7. 测试时间为10minute，记录VF的读带宽。<br>8. 打开VF的QoS功能，QoS分别限制为总写带宽的[10%, 30%, 50%, 70%, 100%]<br>9. 设置workload为sequential write only， 对VF进行顺序写测试。<br>10. 测试时间为10minute，记录VF的写带宽。<br>11. FIO读写模型设置block size为4K，QD为64，numjobs为161 |

[19] QD,numjobs 可以按照产品规格书修改

| | |
|---|---|
| | 12. 打开VF的QoS功能，QoS分别限制为总随机读性能的[10%, 30%, 50%, 70%, 100%]<br>13. 设置workload为random read only，对VF进行随机读测试。<br>14. 测试时间为10minute，记录VF的读IOPS。<br>15. 打开VF的QoS功能，QoS分别限制为总随机写性能的[10%, 30%, 50%, 70%, 100%]<br>16. 设置workload为random write only，对VF进行随机写测试。<br>17. 测试时间为10minute，记录VF的写IOPS。 |
| 通过准则 | 1. 测试结果的BW峰值或IOPS峰值与设定的QoS值，误差在10%以内。 |

| | |
|---|---|
| 测试用例 | SR-IOV 模式下，单盘多VF QoS功能有效性验证 |
| 测试目的 | 验证单盘多VF下，QoS配置不同时，QoS功能有效性 |
| 测试方法 | 1. 擦除SSD<br>2. 设置VF，NS为: VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. FIO读写模型设置block size为128K，QD为64，，numjobs为120<br>5. 打开每个VF的QoS功能，QoS分别限制为总读带宽的10%，20%，30%，40%<br>6. 并行测试4个VF，4个VF均设置为sequential read only。<br>7. 测试时间为10minute，记录4个VF的读带宽。<br>8. 打开每个VF的QoS功能，QoS分别限制为总写带宽的10%，20%，30%，40%<br>9. 并行测试4个VF，4个VF均设置为sequential write only。<br>10. 测试时间为10minute，记录4个VF的写带宽。<br>11. FIO读写模型设置block size为4K，QD为64，，numjobs为1621<br>12. 打开每个VF的QoS功能，QoS分别限制为总 |

[20] QD,numjobs 可以按照产品规格书修改

[21] QD,numjobs 可以按照产品规格书修改

| | |
|---|---|
| | 读IOPS的10%，20%，30%，40%<br>13. 并行测试4个VF，4个VF均设置为random read only。<br>14. 测试时间为10minute，记录4个VF的读IOPS。<br>15. 打开每个VF的QoS功能，QoS分别限制为总写IOPS的10%，20%，30%，40%<br>16. 并行测试4个VF，4个VF均设置为random write only。<br>17. 测试时间为10minute，记录4个VF的写IOPS。 |
| 通过准则 | 1. 每个VF的读写性能峰值与设定的QoS值，误差在10%以内 |

| | |
|---|---|
| 测试用例 | SR-IOV 模式下，单盘多VF之间QoS设置的影响 |
| 测试目的 | 验证多VF之间，改变其中一个VF的QoS，其他VF的QoS是否受影响 |
| 测试方法 | 1. 擦除SSD<br>2. 设置VF，NS为: VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. 打开每个VF的QoS功能，带宽分别限制为总顺序读/写带宽的10%，20%，30%，40%<br>5. FIO分别进行顺序读写测试，测试中途将VF4的QoS改成20%<br>6. 分别读取各个VF测试结果中的BW<br>7. 设置每个VF的QoS IOPS分别为总随机读/写IOPS的10%，20%，30%，40%<br>8. FIO分别进行随机读写测试，测试中途将VF4的QoS改成20%<br>9. 分别读取各个VF测试结果中的IOPS |
| 通过准则 | 1. VF1,VF2,VF3的性能峰值符合预设值，在VF4的QoS设置发生变化后，VF4的性能峰值符合自身QoS的变动 |

### 4. QoS 精度验证

(1) 测试方法概述

主要从两方面进行验证：

a. 根据 SSD 提供的 QoS 设置步长和精度，验证有效性；

b. 如果 SSD 支持多 LBAF，切换不同 LBAF 验证 QoS 精度。

(2) 测试项举例

| 测试用例 | SR-IOV 模式下，单盘单VF QoS精度验证 |
|---|---|
| 测试目的 | 验证产品QoS步长和精度 |
| 测试方法 | 1. 假设SSD IOPS QoS 为 1K IOPS，精度为1%<br>2. 创建1个VF，把VF挂载到虚拟机<br>3. VF设置读IOPS的峰值流量为2000；<br>4. VF进行4K顺序读性能测试，查看VF性能是否满足读IOPS Qos要求<br>5. VF设置读IOPS的峰值流量为3000；<br>6. VF进行4K顺序读性能测试，查看各VF性能是否满足读IOPS Qos要求 |
| 通过准则 | 1. 步骤2中，设置成功；<br>2. 步骤3中，VF性能满足读IOPS Qos要求，误差不超过1%<br>3. 步骤4中，设置成功；<br>4. 步骤5中，VF性能满足读IOPS Qos要求，误差不超过1% |

### 5. 长时间 IO 读写压力测试

(1) 测试方法概述

测试注意点：

a. IO 模型可以分为两大类，一类是覆盖典型的顺序读、写，随机读、写，混合读写，验证中可以覆盖不同块大小，另一类采用业务模型测试，例如抓取的现实业务 trace 回放，常用数据库测试等；

b. IO 过程中可以反复对 PF 下发非 IO 命令，观察 VF 读写情况；

c. 如果 SSD 支持 QoS，可以按照业务需求，配置典型的 QoS 值。

(2) 测试项举例

| | |
|---|---|
| 测试用例 | 单盘多VF在典型IO模型下长时间并发读写测试 |
| 测试目的 | 验证单盘多VF 在典型IO模型下长时间并发压力测试过程，对PF下发非IO命令，观察各VF性能是否稳定 |
| 测试方法 | 1. 擦除SSD<br>2. 设置VF，NS为: VF=4，NS=4，将每个NS attach到对应的VF上<br>3. 对每个VF进行precondition操作，并达到稳态<br>4. 并行地对每个VF进行长时间的典型顺序及随机性能测试，总测试时间不低于8h<br>5. 在测试过程中，反复对PF 下发非IO命令，观察每个VF读写情况<br>6. 统计各种IO 模型长时间测试下，每个VF读写性能以及延时 |
| 通过准则 | 1. 各种IO模型长时间压力测试下，VF读写性能保持稳定并且与标称值的差距在10%以内 |

ODCC公众号

ODCC订阅号