IT 云智能网卡网络/存储卸载接口规范

（2022年）

算网融合产业及标准推进委员会

2022年8月

版权声明

参与编写单位

中国移动通信研究院、中国电信研究院、锐捷网络股份有限公司、中兴通讯股份有限公司、华为技术有限公司、弘协网络科技（北京）有限责任公司

主要撰稿人

王瑞雪、秦凤伟、吴林泽、吴航、阎松明、赵宝鑫、郭力军、刘红巧、曾宏宽

前　言

本标准提出对 IT 云 vSwitch/vRouter 卸载及存储卸载接口要求，用于指导 IT 云 SDN 系统和智能网卡设备的试点和集采。

本标准是 SDN 系统系列标准之一，该系列标准的结构、名称或预计的名称如下：

表1 系列标准

序号	标准编号	标准名称
1		
2		

目　录

一、范围

本标准规定了对IT云vSwitch/vRouter卸载及存储卸载接口要求，适用于IT云SDN系统和智能网卡设备的试点、集采。

二、规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单（不包括勘误的内容）或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

表1 规范性引用文件

版本号	更新时间	主要内容或重大修改

三、术语、定义和缩略语

下列术语、定义和缩略语适用于本标准：

表2　缩略语及解释

缩略语	全称	解释
SDN	Software Defined Network	软件定义网络
vSW	Virtual Switch	虚拟网络交换机
vRouter	Virtual Router	虚拟网络路由器
DPDK	Data Plane Development Kit	数据平面开发套件

OS	Operating System	操作系统
MTU	Maximum Transmission Unit	最大传输单元
vDPA	vHost data path acceleration	虚拟化硬件加速

表3　术语及定义

术语	解释
软件定义网络	Sofware Define network，将控制平面和转发平面分割，数据包将会根据更优化的数据平面被转发，路由(控制平面)功能则运行在某机柜服务器的虚拟机内。
资源池	资源池主要指可以使用的软件和硬件资源集合。 在虚拟化架构中，资源池主要指可以使用的虚拟化资源，比如虚拟机CPU、内存、存储和网络等资源。
软硬解耦	在业务部署过程中，去除软硬件的相互依赖，部署软件不需要关注特定的硬件平台，提高了业务部署的通用性和高效性。

四、适用场景

本规范适用于IT云资源池。

五、接口概要

IT云业务资源池采用NFV/SDN 架构，支持虚拟化场景和裸金属场景。其中虚拟化场景，智能网卡需要实现vSwitch/vRouter（包括控制面和转发面）的卸载及存储的卸载；裸金属场景，智能网卡需要实现弹性裸金属以及云盘挂载功能。本规范针对vSwitch/vRouter卸载和存储卸载场景，本规范主要涵盖vSwitch/vRouter控制面和转发面之间的标准接口，主要包括以下内容：

(1) 软件版本要求：针对vSwitch/vRouter卸载所涉及到的vSwitch/vRouter、DPDK等软件版本进行规范；针对存储卸载所涉及到的Nova computer、SPDK、卡OS等版本进行规范；智能网卡厂商按照基准版本提供驱动、接口服务，虚拟层厂商按照基准版本实现软件部分的对接；

(2) 流表接口要求：采用用户态DPDK标准流表卸载接口rte_flow，要求虚拟层vSwitch/vRouter按照规定方式实现流表从控制面到转发面的卸载，智能网卡厂商支持该部分流表卸载所用到的接口；

(3) 网卡配置要求：对于mtu设置、bond模式配置等功能需求，定义基于DPDK库中的标准统一函数，要求虚拟层vSwitch/vRouter按照规定函数实现配置的下发，智能网卡厂商支持规定的接口函数功能

(4) 网络端口要求：要求智能网卡支持基于用户态的vdpa框架，实现为虚拟机提供统一标准的virtio网络端口，并支持虚拟机热迁移以及裸金属的冷迁移功能

(5) 安装要求：智能网卡驱动以动态库（.SO）的形式与SDN厂商对接。

(6) 网卡安装时需提供/etc/smart_nic_init.cfg配置文件，云平台读取该文件后提供/user/bin/config.json文件，vsw从配置文件获取相应信息，进行启动

基于以上规范要求，规定虚拟层基于标准的接口实现流表卸

载、网卡配置和网络端口的配置，规定智能网卡厂商在基准版本软件中实现统一的标准接口功能。

六、软件版本要求

要求智能网卡厂商按照规定的软件版本实现接口功能，虚拟层厂商实现对应软件模块的对接工作。

- DPDK：20.11
- OpenStack：Pike
- QEMU：4.0.0
- VirtIO：1.0
- SPDK：21.01

七、流表接口要求

采用DPDK标准卸载接口rte_flow，具体参照以下原则：

· 支持卸载IPv4和IPv6报文，传输层协议包含TCP、UDP，其他报文均不卸载，上送软件vSwitch/vRouter处理（如ARP、ICMP、组播、广播）

· 上送软件vSwitch/vRouter的报文需为原报文格式（带VXLAN隧道头），不得修改

· 智能网卡执行VXLAN_ENCAP操作时，须将VXLAN隧道头的源端口号（流表下发时默认填0）进行修改：

- 针对分片IP报文：按照内层报文的三元组{协议号、源IP地址、目的IP地址}进行hash计算，保证不同分片的外层vxlan头源端口号一致

- 针对未分片IP报文：按照内层报文的五元组{协议号、源IP地址、目的IP地址、源端口、目的端口}进行hash计算

（一）接口内容

通过DPDK rte_flow_create接口实现流表的卸载，智能网卡需支持本章列出的所有接口和参数：

序号	名称	内容	说明
1	结构体	rte_flow_attr	流表规则属性
	参数	uint32_t group	每一个报文需匹配每个group
		uint32_t reserved：29	用高两位代表流表类型： 0：精确匹配转发流表 1：模糊匹配转发流表//IT云不涉及 2：DP-HASH流表//IT云不涉及 3：ACL规则流表
		uint32_t ingress	仅在流表类型为ACL规则时（reserved=3）有效，代表ACL规则为入向或出向 在本项目中不涉及1和2类型
2	接口函数	rte_flow_create	在某一端口上创建一条流表规则
	参数	uint16_t port_id	端口id
		const struct rte_flow_attr * attr	流表规则属性
		const struct rte_flow_item pattern[]	流表规则匹配项（包含自定义ITEM）
		const struct rte_flow_action actions[]	流表规则动作项（包含自定义ACTION）

序号	名称	内容	说明
		struct rte_flow_error * error	错误信息
	返回值	成功时返回rte_flow句柄，否则为错误信息	

1．原生ITEM

序号	名称	内容	说明
1	TYPE	RTE_FLOW_ITEM_TYPE_ETH	匹配以太层信息
	SPEC	rte_flow_item_eth()	Struct名
		>struct rte_ether_addr src	源mac地址
		>struct rte_ether_addr dst	目的mac地址
		>rte_be16_t type	协议类型： VLAN：0x8100 IPV4：0x0800 IPV6：0x86dd
2	TYPE	RTE_FLOW_ITEM_TYPE_VLAN	匹配vlan信息
	SPEC	rte_flow_item_vlan()	Struct名
		>rte_be16_t tci	Vlan tci值
		>rte_be16_t inner_type	内层协议类型： IPV4：0x0800 IPV6：0x86dd
3	TYPE	RTE_FLOW_ITEM_TYPE_IPV4	匹配ipv4信息
	SPEC	rte_flow_item_ipv4()	Struct名
		>struct rte_ipv4_hdr hdr	Ipv4数据结构
		>>rte_be32_t src_addr	源IP地址
		>>rte_be32_t dst_addr	目的IP地址
		>>uint8_t next_proto_id	传输层协议号： TCP：6 UDP：17
		>>uint8_t type_of_service	服务类型
4	TYPE	RTE_FLOW_ITEM_TYPE_IPV6	匹配ipv6信息
	SPEC	rte_flow_item_ipv6()	Struct名
		>struct rte_ipv6_hdr hdr	Ipv6数据结构
		>>uint8_t src_addr [16]	源IP地址

序号	名称	内容	说明
		>>uint8_t dst_addr [16]	目的IP地址
		>>uint8_t proto	传输层协议号： TCP：6 UDP：17
		>>rte_be32_t vtc_flow	服务类型
5	TYPE	RTE_FLOW_ITEM_TYPE_VXLAN	匹配vxlan信息
	SPEC	rte_flow_item_vxlan()	Struct名
		>uint8_t vni[3]	Vni值
		>uint8_t rsvd0[3]	使用Reseved0字段（22-24位）实现vxlan染色
6	TYPE	RTE_FLOW_ITEM_TYPE_TCP	匹配TCP信息
	SPEC	rte_flow_item_tcp()	Struct名
		>struct rte_tcp_hdr hdr	TCP数据结构
		>>rte_be16_t src_port	TCP源端口
		>>rte_be16_t dst_port	TCP目的端口
		>>uint8_t tcp_flags	TCP标志位。当前仅关心其中的0、1、2 bit，即FIN、SYN和RST标志位。通过设置rte_flow_item中的mask来实现仅匹配FIN、SYN和RST标志位均为0的正常传输流程报文，忽略连接状态相关的SYN、FIN和RST报文，从而保证这些报文可以上送控制面。
7	TYPE	RTE_FLOW_ITEM_TYPE_UDP	匹配UDP信息
	SPEC	rte_flow_item_udp()	Struct名
		>struct rte_udp_hdr hdr	UDP数据结构
		>>rte_be16_t src_port	UDP源端口

序号	名称	内容	说明
		>>rte_be16_t dst_port	UDP目的端口
8	TYPE	RTE_FLOW_ITEM_TYPE_TAG	报文TAG标记
	SPEC	rte_flow_item_tag()	Struct名
		>uint32_t data	TAG值
		>uint8_t index	TAG序号
9	TYPE	RTE_FLOW_ITEM_TYPE_ICMP	
	SPEC	>struct rte_icmp_hdr hdr	
		>>uint8_t icmp_type	ICMP报文类型
		>>uint8_t icmp_code	ICMP报文代码
10	TYPE	RTE_FLOW_ITEM_TYPE_END	ITEM结束

2．原生ACTION

序号	名称	内容	说明
1	TYPE	RTE_FLOW_ACTION_TYPE_COUNT	流表统计
	CONF	rte_flow_action_count()	Struct名
		>uint32_t id	Count_id
2	TYPE	RTE_FLOW_ACTION_TYPE_PORT_ID	转发到端口
	CONF	rte_flow_action_port_id()	Struct名
		>uint32_t id	端口id
3	TYPE	RTE_FLOW_ACTION_TYPE_OF_POP_VLAN	POP VLAN
4	TYPE	RTE_FLOW_ACTION_TYPE_OF_PUSH_VLAN	PUSH VLAN
	CONF	rte_flow_action_of_push_vlan()	Struct名
		>rte_be16_t ethertype	协议类型
5	TYPE	RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_VID	设置vlan id
	CONF	rte_flow_action_of_set_vlan_vid()	Struct名
		>rte_be16_t vlan_vid	Vlan id
6	TYPE	RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_PCP	设置vlan优先级
	CONF	rte_flow_action_of_set_vlan_pcp()	Struct名
		>uint8_t vlan_pcp	Vlan优先级

序号	名称	内容	说明
7	TYPE	RTE_FLOW_ACTION_TYPE_VXLAN_DECAP	解封装vxlan头
8	TYPE	RTE_FLOW_ACTION_TYPE_VXLAN_ENCAP	封装vxlan头
9	CONF	rte_flow_action_vxlan_encap()	Struct名
		>struct rte_flow_item * definition	Vxlan头key值，包含 • ETH/IPV4/UDP/VXLAN/END • ETH/IPV6/UDP/VXLAN/END • ETH/VLAN/IPV4/UDP/VXLAN/END • ETH/VLAN/IPV6/UDP/VXLAN/END
10	TYPE	RTE_FLOW_ACTION_TYPE_DEC_TTL	TTL减一
11	TYPE	RTE_FLOW_ACTION_TYPE_SET_MAC_SRC	修改源mac地址
	CONF	rte_flow_action_set_mac()	Struct名
		>uint8_t mac_addr[RTE_ETHER_ADDR_LEN]	Mac地址值
12	TYPE	RTE_FLOW_ACTION_TYPE_SET_DST_SRC	修改目的mac地址
	CONF	rte_flow_action_set_mac()	Struct名
		>uint8_t mac_addr[RTE_ETHER_ADDR_LEN]	Mac地址值
13	TYPE	RTE_FLOW_ACTION_TYPE_SAMPLE	采样转发，流镜像
	CONF	rte_flow_action_sample()	Struct名
		>uint32_t ratio	采样比例
		>const struct rte_flow_action * actions	执行动作
14	TYPE	RTE_FLOW_ACTION_TYPE_SET_IPV4_SRC	修改源IP (IPv4)
	CONF	rte_flow_action_set_ipv4	Struct名
	TYPE	>rte_be32_t ipv4_addr	IP地址值
15	TYPE	RTE_FLOW_ACTION_TYPE_SET_IPV4_DST	修改目的IP (IPv4)

序号	名称	内容	说明
	CONF	rte_flow_action_set_ipv4	Struct名
		>rte_be32_t ipv4_addr	IP地址值
16	TYPE	RTE_FLOW_ACTION_TYPE_SET_IPV6_SRC	修改源IP (IPv6)
	CONF	rte_flow_action_set_ipv6	Struct名
		>uint8_t ipv6_addr[16]	IP地址值
17	TYPE	RTE_FLOW_ACTION_TYPE_SET_IPV6_DST	修改目的IP (IPv6)
	CONF	rte_flow_action_set_ipv6	Struct名
		>uint8_t ipv6_addr[16]	IP地址值
18	TYPE	RTE_FLOW_ACTION_TYPE_DROP	丢弃报文
19	TYPE	RTE_FLOW_ACTION_TYPE_SET_TAG	为报文设置内标记。(上送时使用rte_mbuf结构体中dynfield1[8]低两字节（16bit）保存index和data，index用高8bit，data用低8bit)
	CONF	rte_flow_action_set_tag()	Struct名
		>uint32_t data	TAG值(只使用低8bit)
		>uint8_t index	TAG序号
20	TYPE	RTE_FLOW_ACTION_TYPE_INDIRECT	引用handle操作
	CONF	rte_flow_action_handle()	Struct名
		uint32_t id	Handle id
21	TYPE	rte_flow_action_modify_field()	Struct名
	CONF	>enum rte_flow_modify_op operation	操作，包含：RTE_FLOW_MODIFY_SET RTE_FLOW_MODIFY_ADD（可选） RTE_FLOW_MODIFY_SUB（可选）

序号	名称	内容	说明
		>struct rte_flow_action_modify_data dst	目的字段
		>struct rte_flow_action_modify_data srt	源字段
		>uint32_t width	待修改字段长度
22	TYPE	RTE_FLOW_ACTION_TYPE_END	ACTION结束

3．FIELD

针对MODIFY_FIELD action，智能网卡需支持以下FIELD：

字段	描述
RTE_FLOW_FIELD_IPV4_DSCP	DSCP值，用于优先级、探针、染色
RTE_FLOW_FIELD_IPV6_DSCP	
RTE_FLOW_FIELD_VXLAN_RSVD0	VXLAN头保留字段reserved0，用于染色

4．自定义ITEM

序号	名称	内容	说明
1	TYPE	RTE_FLOW_ITEM_TYPE_OUTER_IPV4	匹配vxlan外层ipv4信息
	SPEC	rte_flow_item_ipv4()	Struct名
		>struct rte_ipv4_hdr hdr	Ipv4数据结构
		>>rte_be32_t src_addr	源IP地址
		>>rte_be32_t dst_addr	目的IP地址
2	TYPE	RTE_FLOW_ITEM_TYPE_OUTER_IPV6	匹配vxlan外层ipv6信息
	SPEC	rte_flow_item_ipv6()	Struct名
		>struct rte_ipv6_hdr hdr	Ipv6数据结构
		>>uint8_t src_addr [16]	源IP地址
		>>uint8_t dst_addr [16]	目的IP地址

（二）同节点二层转发

同节点内VM间完成二层转发，接口调用如下：

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>spec = rte_flow_item_eth	src	[src-vm-mac]
	dsc	[dst-vm-mac]
	type	0x8100 or 0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type!=0x8100，所有值全为0)		
>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若 eth.type =0x0800 or vlan.inner_type =0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]
	rte_ipv4_hdr.next_proto_id	6 or 17 or 132
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若 eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17 or 132
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	0
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_VID(若 eth.type=0x8100 and has_vlan=1)		

接口	参数	值
>>conf = rte_flow_action_of_set_vlan_vid	vlan_vid	[vlan_vid]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_PCP(若 eth.type=0x8100 and has_vlan=1)		
>>conf = rte_flow_action_of_set_vlan_pcp	vlan_pcp	[vlan_pcp]
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[dst-vm-port-id]
>>type = RTE_FLOW_ACTION_TYPE_END		

（三）跨节点二层转发

跨节点完成VM间二层转发，分出节点和入节点两个方向，接口调用如下：

1．出节点

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>spec = rte_flow_item_eth	src	[src-vm-mac]
	dsc	[dst-vm-mac]
	type	0x8100 or 0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type!=0x8100，所有值全为0)		
>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type =0x0800 or vlan.inner_type =0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]

接口	参数	值
	rte_ipv4_hdr.next_proto_id	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	0
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_OF_POP_VLAN (若eth.type=0x8100 and has_vlan=1)		
>>type = RTE_FLOW_ACTION_TYPE_VXLAN_ENCAP		
>>conf = rte_flow_action_vxlan_encap	rte_flow_item definition	• ETH/IPV4/UDP/VXLAN/END • ETH/IPV6/UDP/VXLAN/END • ETH/VLAN/IPV4/UDP/VXLAN/END • ETH/VLAN/IPV6/UDP/VXLAN/END
>>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>>spec = rte_flow_item_eth	src	[src-vtep-mac]
	dsc	[dst-vtep-mac]
	type	0x8100 or 0x0800 or 0x86dd

接口	参数	值
	has_vlan	1 or 0
>>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type=0x8100 and has_vlan=1)		
>>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type =0x0800 or vlan.inner_type =0x0800)		
>>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.src_addr	[src-vtep-ip]
	rte_ipv4_hdr.dst_addr	[dst-vtep-ip]
	rte_ipv4_hdr.next_proto_id	17
>>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.src_addr	[src-vtep-ip]
	rte_ipv6_hdr.dst_addr	[dst-vtep-ip]
	rte_ipv6_hdr.proto	17
>>>type = RTE_FLOW_ITEM_TYPE_UDP		
>>>spec = rte_flow_item_udp	rte_udp_hdr.src_port	0
	rte_udp_hdr.dst_port	4789
>>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>>spec = rte_flow_item_vxlan	vni	[vni]
>>>type = RTE_FLOW_ITEM_TYPE_END		
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[bond-port-id]
>>type = RTE_FLOW_ACTION_TYPE_END		

2. 入节点

接口	参数	值
rte_flow_create	port_id	[bond-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>type = RTE_FLOW_ITEM_TYPE_ETH		

接口	参数	值
>>spec = rte_flow_item_eth	src	[src-vm-mac]
	dsc	[dst-vm-mac]
	type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN		
>>spec = rte_flow_item_vlan	tci	0
	inner_type	0
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type=0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]
	rte_ipv4_hdr.next_proto_id	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type=0x86dd)		
>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	[vni]
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_VXLAN_DECAP		
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_OF_PUSH_VLAN（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_push_vlan	ethertype	[eth.type]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_VID（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_set_vlan_vid	vlan_vid	[vlan_vid]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_PCP（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_set_vlan_pcp	vlan_pcp	[vlan_pcp]
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[dst-vm-port-id]
>>type = RTE_FLOW_ACTION_TYPE_END		

（四）同节点三层转发

同节点内VM间完成三层转发，接口调用如下：

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>spec = rte_flow_item_eth	src	[src-vm-mac]
	dsc	[gw-mac]
	type	0x8100 or 0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type!=0x8100，所有值全为0)		
>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type =0x0800 or vlan.inner_type =0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]
	rte_ipv4_hdr.next_proto_id	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	0
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_SRC		
>>conf = rte_flow_action_set_mac	mac_addr	[gw-mac]

接口	参数	值
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_DST		
>>conf = rte_flow_action_set_mac	mac_addr	[dst-vm-mac]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_VID(若eth.type=0x8100)		
>>conf = rte_flow_action_of_set_vlan_vid	vlan_vid	[vlan_vid]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_PCP(若eth.type=0x8100)		
>>conf = rte_flow_action_of_set_vlan_pcp	vlan_pcp	[vlan_pcp]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV4_SRC (若有IPv4 FIP，正向流)		
>>conf = rte_flow_action_set_ipv4	ipv4_addr	[src-fip]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV6_SRC (若有IPv6 FIP，正向流)		
>>conf = rte_flow_action_set_ipv6	ipv6_addr	[src-fip]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV4_DST (若有IPv4 FIP，反向流)		
>>conf = rte_flow_action_set_ipv4	ipv4_addr	[dst-vm-ip]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV6_DST (若有IPv6 FIP，反向流)		
>>conf = rte_flow_action_set_ipv6	ipv6_addr	[dst-vm-ip]
>>type = RTE_FLOW_ACTION_TYPE_DEC_TTL		
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[dst-vm-port-id]
>>type = RTE_FLOW_ACTION_TYPE_END		

（五）跨节点三层转发

跨节点完成VM间三层转发，分出节点和入节点两个方向，接口调用如下：

1．出节点

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-

接口	参数	值
>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>spec = rte_flow_item_eth	src	[src-vm-mac]
	dsc	[src-gw-mac]
	type	0x8100 or 0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type!=0x8100，所有值全为0)		
>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type =0x0800 or vlan.inner_type =0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]
	rte_ipv4_hdr.next_proto_id	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	0
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_OF_POP_VLAN (若eth.type=0x8100 and has_vlan=1)		
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_SRC		
>>conf = rte_flow_action_set_mac	mac_addr	[src-gw-mac]
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_DST		
>>conf = rte_flow_action_set_mac	mac_addr	[dst-gw-mac]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV4_SRC (若有IPv4 FIP)		
>>conf = rte_flow_action_set_ipv4	ipv4_addr	[src-fip]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV6_SRC (若有IPv6 FIP)		

接口	参数	值
>>conf = rte_flow_action_set_ipv6	ipv6_addr	[src-fip]
>>type = RTE_FLOW_ACTION_TYPE_VXLAN_ENCAP		
>>conf = rte_flow_action_vxlan_encap	rte_flow_item definition	• ETH/IPV4/UDP/VXLAN/END • ETH/IPV6/UDP/VXLAN/END • ETH/VLAN/IPV4/UDP/VXLAN/END • ETH/VLAN/IPV6/UDP/VXLAN/END
>>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>>spec = rte_flow_item_eth	src	[src-vtep-mac]
	dsc	[dst-vtep-mac]
	type	0x8100 or 0x0800 or 0x86dd
	has_vlan	1 or 0
>>>type = RTE_FLOW_ITEM_TYPE_VLAN (若eth.type=0x8100 and has_vlan=1)		
>>>spec = rte_flow_item_vlan	tci	[vlan-tci]
	inner_type	0x0800 or 0x86dd
>>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若 eth.type =0x0800 or vlan.inner_type =0x0800)		
>>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.src_addr	[src-vtep-ip]
	rte_ipv4_hdr.dst_addr	[dst-vtep-ip]
	rte_ipv4_hdr.next_proto_id	17
>>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若 eth.type =0x86dd or vlan.inner_type=0x86dd)		
>>>spec = rte_flow_item_ipv6	rte_ipv6_hdr.src_addr	[src-vtep-ip]

接口	参数	值
	rte_ipv6_hdr.dst_addr	[dst-vtep-ip]
	rte_ipv6_hdr.proto	17
>>>type = RTE_FLOW_ITEM_TYPE_UDP		
>>>spec = rte_flow_item_udp	rte_udp_hdr.src_port	0
	rte_udp_hdr.dst_port	4789
>>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>>spec = rte_flow_item_vxlan	vni	[vni]
>>>type = RTE_FLOW_ITEM_TYPE_END		
>>type = RTE_FLOW_ACTION_TYPE_DEC_TTL		
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>rte_flow_action_port_id	id	[bond-port-id]
>>type = RTE_FLOW_ACTION_TYPE_DEC_TTL		
>>type = RTE_FLOW_ACTION_TYPE_END		

2. 入节点

接口	参数	值
rte_flow_create	port_id	[bond-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>type = RTE_FLOW_ITEM_TYPE_ETH		
>>spec = rte_flow_item_eth	src	[src-gw-mac]
	dsc	[dst-gw-mac]
	type	0x0800 or 0x86dd
>>type = RTE_FLOW_ITEM_TYPE_VLAN		
>>spec = rte_flow_item_vlan	tci	0
	inner_type	0
>>type = RTE_FLOW_ITEM_TYPE_IPV4 (若eth.type=0x0800)		
>>spec = rte_flow_item_ipv4	rte_ipv4_hdr.dst_addr	[dst-vm-ip]
	rte_ipv4_hdr.next_proto_id	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_IPV6 (若eth.type=0x86dd)		

接口	参数	值
>>spec = rte_flow_item_ipv4	rte_ipv6_hdr.dst_addr	[dst-vm-ip]
	rte_ipv6_hdr.proto	6 or 17
>>type = RTE_FLOW_ITEM_TYPE_VXLAN		
>>spec = rte_flow_item_vxlan	vni	[vni]
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>RTE_FLOW_ACTION_TYPE_VXLAN_DECAP		
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_SRC		
>>conf = rte_flow_action_set_mac	mac_addr	[dst-gw-mac]
>>type = RTE_FLOW_ACTION_TYPE_SET_MAC_DST		
>>conf = rte_flow_action_set_mac	mac_addr	[dst-vm-mac]
>>type = RTE_FLOW_ACTION_TYPE_OF_PUSH_VLAN（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_push_vlan	ethertype	[eth.type]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_VID（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_set_vlan_vid	vlan_vid	[vlan_vid]
>>type = RTE_FLOW_ACTION_TYPE_OF_SET_VLAN_PCP（若目的vNIC为Trunk Port）		
>>conf = rte_flow_action_of_set_vlan_pcp	vlan_pcp	[vlan_pcp]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV4_DST (若有IPv4 FIP)		
>>conf = rte_flow_action_set_ipv4	ipv4_addr	[dst-vm-ip]
>>type = RTE_FLOW_ACTION_TYPE_SET_IPV6_DST (若有IPv6 FIP)		
>>conf = rte_flow_action_set_ipv6	ipv6_addr	[dst-vm-ip]
>>type = RTE_FLOW_ACTION_TYPE_DEC_TTL		
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[dst-vm-port-id]
>>type = RTE_FLOW_ACTION_TYPE_END		

（六）流镜像

使用RTE_FLOW_ACTION_TYPE_SAMPLE接口，将匹配到的flow完成100%采样转发到相应的内部port（使用固定的端口完成本地镜像）或出口port（远端镜像到物理端口，使用RTE_FLOW_ACTION_VXLAN_ENCAP封装vxlan隧道完成远端镜像）

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>…		
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_COUNT		
>>conf = rte_flow_action_count	id	[count_id]
>>…		
>>type = RTE_FLOW_ACTION_TYPE_PORT_ID		
>>conf = rte_flow_action_port_id	id	[dst-vm-port-id]
>>type = RTE_FLOW_ACTION_TYPE_SAMPLE		
>>conf = rte_flow_action_sample	ratio	1
	rte_flow_action actions	• 本地镜像：port_id • 远端镜像：vxlan_encap/port_id
>>>…		
>>>type = RTE_FLOW_ACTION_TYPE_END		
>>type = RTE_FLOW_ACTION_TYPE_END		

（七）安全组

1．有状态安全组

启用安全组功能，需要将连接的首包上送慢速路径，由慢速

路径判断该连接是否通过安全组规则。如果通过安全组规则校验，则根据7.2、7.3、7.4和7.5中定义的不同场景下发定义卸载流表。在卸载流表中需要增加TCP或UDP相关的匹配规则，详情参见7.1.1中RTE_FLOW_ITEM_TYPE_TCP和RTE_FLOW_ITEM_TYPE_UDP的相关描述。需要特别注意的是TCP的连接状态相关的SYN、FIN和RST报文不能卸载，需要上送慢速路径，目前是通过卸载流表中匹配tcp_flags中的第0、1、2 BIT均为0来实现的。

2．无状态安全组

rte_flow_attr, rte_flow_item与二/三层转发的接口相同

接口	参数	值
rte_flow_create	port_id	[src-vm-port-id]
>rte_flow_attr	ingress	1
>rte_flow_item	-	-
>>…		
>>type = RTE_FLOW_ITEM_TYPE_END		
>rte_flow_action	-	-
>>type = RTE_FLOW_ACTION_TYPE_DROP		
>>type = RTE_FLOW_ACTION_TYPE_END		

（八）ACL规则流表

针对运维统计需求，通过rte_flow接口（rte_flow_attr.reserved=3）将ACL流表规则卸载到智能网卡上，主要用于染色、统计、探针等功能。

智能网卡需支持在路由转发前和后的ACL规则，ACL规则在创建时必须指定方向（ingress或egress），ingress代表报文进入智能网卡时（即路由转发前）执行ACL规则，egress代表报文出智能网卡时（即路由转发后）执行ACL规则。

智能网卡对于转发面处理（硬件）和上送控制面（软件）下行的报文需要全部执行（入向ACL→路由转发→出向ACL）的流程。

智能网卡需支持对ACL规则进行分组，同一端口同一方向的ACL规则需要对每个分组进行匹配查询，通过flow_attr中的group参数标识不同的规则组编号，要求支持双向共4个规则组（单向2个规则组），每个规则组16条ACL规则，所有规则总数不超过32条。

ACL规则可在入向时通过RTE_FLOW_ACTION_TYPE_SET_TAG为报文设置TAG标记，在出向时通过匹配TAG标记即可实现对原报文的统计。

ACL规则支持7.1.3章节ACITON和上送（UPCALL）ACTION任意组合，可以使用RTE_FLOW_ACTION_TYPE_INDIRECT以引用1个ACTION_HANDLE（仅ACTION_COUNT），要求智能网卡支持创建32个ACTION_HANDLE，相关接口如下：

序号	名称	内容	说明
1	结构体	rte_flow_action_handle	Action handle
	参数	uint32_t id	Action handle id
2	接口函数	rte_flow_action_handle_create	创建action handle
	参数	uint16_t port_id	端口id

序号	名称	内容	说明
		const struct rte_flow_indir_action_conf * conf	Handle action配置
		const struct rte_flow_action * action	Action（仅支持count）
		struct rte_flow_error * error	错误信息
	返回值	rte_flow_action_handle为成功，否则为错误信息	
3	接口函数	rte_flow_action_handle_destroy	删除action handle
	参数	uint16_t port_id	端口id
		struct rte_flow_action_handle * handle	Action handle信息
		struct rte_flow_error * error	错误信息
	返回值	0为成功，否则为错误信息	
4	接口函数	rte_flow_action_handle_query	查询action handle
	参数	uint16_t port_id	端口id
		const struct rte_flow_action_handle * handle	Handle action
		void *data	结构体（见8.5章节） rte_flow_query_count()
		struct rte_flow_error *error	错误信息
	返回值	0为成功，否则为错误信息	

八、配置要求

要求智能网卡厂商实现标准接口，虚拟层厂商调用接口实现对应功能。

（一）MTU

使用DPDK原生rte_eth_dev_set_mtu()接口（智能网卡用户态驱

动实现），设置虚拟端口的入向MTU值。

OvS调用如下DPDK接口实现对应功能：

DPDK：/lib/librte_ethdev/rte_ethdev.h

序号	名称	内容	说明
1	接口函数	rte_eth_dev_set_mtu	设置端口mtu值
	参数	uint16_t port_id	端口id
		uint16_t mtu	MTU值
	返回值		0为成功，否则为错误信息

（二）QoS

通过rte_mtr接口实现基于端口的QoS，基于流的QoS作为可选项

DPDK：/lib/librte_ethdev/rte_mtr.h

序号	名称	内容	说明
1	结构体	rte_mtr_params	MTR对象参数
	参数	uint32_t meter_profile_id	Profile id
		int use_prev_mtr_color	Meter输入颜色
		enum rte_color * dscp_table	Meter输入颜色
		int meter_enable	Meter使能状态
		uint64_t stats_mask	统计计数器集
		uint32_t meter_policy_id	Policy id
		uint32_t direction	1为入向，代表数据从虚拟机进入网卡vf口 2为出向，代表数据从网卡vf口进入虚拟机
2	接口函数	rte_mtr_create	为指定端口创建MTR对象
	参数	uint16_t port_id	端口id

序号	名称	内容	说明
		uint32_t mtr_id	MTR id
		struct rte_mtr_params *params	MTR对象参数
		int shared	MTR对象被单流使用为0
		struct rte_mtr_error *error	错误信息
	返回值		0为成功，否则为错误信息
3	接口函数	rte_mtr_destroy	删除MTR对象
	参数	uint16_t port_id	端口id
		uint32_t mtr_id	MTR id
		struct rte_mtr_error *error	错误信息
	返回值		0为成功，否则为错误信息
4	接口函数	rte_mtr_meter_profile_add	Profile添加
	参数	uint16_t port_id	端口id
		uint32_t meter_profile_id	Profile id
		struct rte_mtr_meter_profile *profile	Profile参数
		struct rte_mtr_error *error	错误信息
	返回值		0为成功，否则为错误信息
5	接口函数	rte_mtr_meter_profile_delete	Profile删除
	参数	uint16_t port_id	端口id
		uint32_t meter_profile_id	Profile id
		struct rte_mtr_error *error	错误信息
	返回值		0为成功，否则为错误信息
6	接口函数	rte_mtr_meter_profile_update	Profile更新
	参数	uint16_t port_id	端口id
		uint32_t mtr_id	MTR id
		uint32_t meter_profile_id	Profile id
		struct rte_mtr_error *error	错误信息
	返回值		0为成功，否则为错误信息

（三）隧道配置

使用rte_flow_tunnel保存vxlan隧道信息，通过rte_flow_tunnel_decap_set接口完成隧道解封装配置：

DPDK：/lib/librte_ethdev/rte_flow.h

序号	名称	内容	说明
1	结构体	rte_flow_tunnel	隧道信息
	参数	enum rte_flow_item_type	隧道类型，值为RTE_FLOW_ITEM_TYPE_VXLAN
		rte_be32_t dst_addr	目的ip地址（ipv4）-本机IP
		uint8_t dst_addr[16]	目的ip地址（ipv6）-本机IP
		rte_be16_t tp_dst	目的端口，值为4789
2	接口函数	rte_flow_tunnel_decap_set	配置隧道
	参数	uint16_t port_id	Port_id
		struct rte_flow_tunnel *tunnel	隧道信息
		struct rte_flow_error *error	错误信息
	返回值	0为成功，否则为错误信息	

（四）端口Bond

通过内核态驱动或者用户态驱动设置物理网口Bond模式，bond策略在智能网卡中执行。

智能网卡需要支持以下bond模式：

- BONDING_MODE_ACTIVE_BACKUP

- BONDING_MODE_BALANCE_XOR(可选)
- BONDING_MODE_8023AD（默认）

智能网卡需要支持以下hash模式：

- XMIT_POLICY_LAYER2
- XMIT_POLICY_LAYER23
- XMIT_POLICY_LAYER34

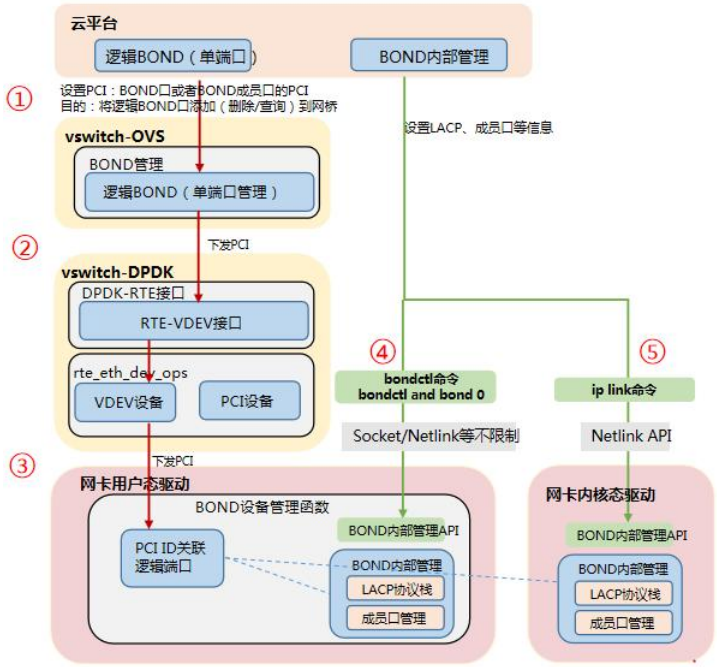

图 1

云平台：获取智能网卡的用户态还是内核态bond信息，用户态bond则调用bondctl命令，内核态bond则直接调用ip link命令，设置bond内部管理信息；

用户态智能网卡：提供bond内部管理工具，用于LACP状态、成员口配置等管理，同时需要具备bond持久化能力；

vSwitch/vRouter：将逻辑端口添加至vSwitch/vRouter网桥

（1）云平台下发BOND成员口PCI至vswitch-ovs；

（2）vswitch-ovs直接使用DPDK RTE接口 透传PCI信息至网卡驱动;

（3）网卡驱动获取PCI信息后，创建vdev设备或pci设备（两者二选一）并注册至DPDK的rte_eth_dev框架中，DPDK框架为其分配PORT ID；

（4）DPDK将生成的PORT ID返回给vswitch-ovs, vswitch-ovs将该端口添加至网桥。

涉及到的各种接口内容如下：

标号	位置	功能描述	接口

1	云平台
	<-->
	vswitch
		云平台向vswitch提供逻辑BOND信息，vswitch将该逻辑BOND口添加至网桥	位置：/usr/bin/config.json
			权限：其他用户或属组具备“读”权限
			内容：
			{
			"bond_info": {
			"name": "bond0",--云平台指定
			"pci": ""
			“busname”: “vdev”or”pci”
			“prefix”:”ys-bond”
			},
			"global_cfg": {
			"nic_driver_path": "" //网卡动态库所在的路径
			}
			}

2	vswitch ovs <-> vswitch dpdk	在DPDK内创建该逻辑BOND端口	int rte_eal_hotplug_add(const char *busname, // “pci” const char *devname, // “0000:7d:00.2”---pci const char *drvargs // ” bond_name=trunk1”) (const char *busname, // “vdev” const char *devname, // “prefix+name” const char *drvargs // ”NULL”) int rte_eth_dev_get_port_by_name(const char *name, // “trunk1” uint16_t *port_id //出参)

3	vswitch dpdk
	<-->
	网卡用户态驱动
		各厂商网卡实现该结构体中的收发包以及设备管理钩子函数	struct rte_eth_dev{
			eth_rx_burst_t rx_pkt_burst;
			eth_tx_burst_t tx_pkt_burst;
			…
			const struct eth_dev_ops *dev_ops;
			…
			}

4	云平台 <--> 网卡用户态管理工具	用于BOND端口内部LACP模式/成员口信息等增删改查	bondctl add BOND_DEV [mode BONDMODE]//创建bond bondctl delete BOND_DEV//删除bond bondctl set BOND_DEV mode BONDMODE//设置模式 bondctl set SLAVE_DEV master BOND_DEV//添加成员口 bondctl set SLAVE_DEV nomaster//删除成员口 bondctl set BOND_DEV primary SLAVE_DEV//设置主成员口

			bondctl set BOND_DEV xmit_hash_policy XMIT_HASH_POLICY//设置bond策略 bondctl [OPTIONS] show [dev BOND_DEV] //查询bond状态 bondctl [OPTIONS] show bond_slave [dev SLAVE_DEV] //查询bond成员状态 bondctl xstats [dev BOND_DEV] //查询 bond LACP协议状态 bondctl xstats bond_slave [dev SLAVE_DEV] //查询bond成员LACP协议状态
5	云平台 <--> 网卡内核态管理工具	用于BOND端口内部LACP模式/成员口信息等增删改查	ip link add BOND_DEV type bond [mode BONDMODE]//bond配置命令 ip link delete BOND_DEV//删除bond ip link set BOND_DEV type bond mode BONDMODE//设置bond模式 ip link set SLAVE_DEV master BOND_DEV//添加成员口 ip link set SLAVE_DEV nomaster//删除成员口 ip link set BOND_DEV type bond primary SLAVE_DEV//设置主成员口（主备模式） ip link set BOND_DEV type bond xmit_hash_policy XMIT_HASH_POLICY//设置bond策略（负载均衡模式） ip [OPTIONS] link show type bond [dev BOND_DEV] //查询bond状态 ip [OPTIONS] link show type bond_slave [dev SLAVE_DE]//查询bond成员状态

其中，接口4和接口的可选参数如下：

BONDMODE：active-backup | 802.3ad

XMIT_HASH_POLICY：layer2 | layer2+3 | layer3+4

BOND_DEV：逻辑bond名称，如：bond0

SLAVE_DEV：bond成员口名称，如：eth0

OPTIONS：-d[etails] | -s[tatistics] | -h[uman-readable]

（五）流统计

通过rte_flow_query()、rte_flow_action_handle_query()接口统计流表count信息，count信息由struct rte_flow_query_count()实现。

DPDK：/lib/librte_ethdev/rte_flow.h

序号	名称	内容	说明
1	结构体	rte_flow_query_count	Flow计数信息
	参数	uint32_t reset	开始执行后是否重置
		uint32_t hits_set	命中字段是否设置
		uint32_t bytes_set	字节字段是否设置
		uint64_t hits	命中流表的次数
		uint64_t bytes	命中流表的字节数
2	接口函数	rte_flow_query	查询流
	参数	uint16_t port_id	Port_id
		struct rte_flow *flow	Flow详情
		void *data	结构体 rte_flow_query_count()
		struct rte_flow_error *error	错误信息
	返回值	0为成功，否则为错误信息	
3	结构体	rte_flow_action_handle	Action handle
	参数	uint32_t id;	Action handle id
4	接口函数	rte_flow_action_handle_create	创建action handle
	参数	uint16_t port_id	端口id
		const struct rte_flow_indir_action_conf * conf	间接action配置
		const struct rte_flow_action * action	间接action内容

序号	名称	内容	说明
		struct rte_flow_error * error	错误信息
	返回值	rte_flow_action_handle为成功，否则为错误信息	
5	接口函数	rte_flow_action_handle_destroy	删除action handle
	参数	uint16_t port_id	端口id
		struct rte_flow_action_handle * handle	Action handle信息
		struct rte_flow_error * error	错误信息
	返回值	0为成功，否则为错误信息	

（六）流表老化

通过rte_flow_destroy()接口定时删除流表，由ovs决定老化策略。

DPDK：/lib/librte_ethdev/rte_flow.h

序号	名称	内容	说明
1	接口函数	rte_flow_destroy	删除流
	参数	uint16_t port_id	Port_id
		struct rte_flow *flow	Flow详情
		struct rte_flow_error *error	错误信息，默认为null
	返回值	0为成功，否则为错误信息	

（七）流表DUMP

自定义如下rte_flow接口函数，由智能网卡用户态驱动实现，OvS调用相关接口查询网卡上当前流表信息：

DPDK：/lib/librte_ethdev/rte_flow.h

序号	名称	内容	说明
1	接口函	rte_flow_dump_start	开始查询

序号	名称	内容	说明
	数		
	参数	void **context	上下文
		uint32_t type	流表类型，7.1章节 rte_flow_attr.reserved
	返回值	0为成功，否则为错误信息	
2	接口函数	rte_flow_dump_next	执行查询n条流表
	参数	void *context	上下文
		uint32_t count	希望查询的流条目数
		uint32_t *dumped_count	输出参数，实际查询到的流表条目数
		const struct rte_flow_item **pattern[]	输出参数，实际查询到的流表规则匹配项
		const struct rte_flow_action **actions[]	输出参数，实际查询到的流表规则动作项
	返回值	0为成功，否则为错误信息	
3	接口函数	rte_flow_dump_done	结束查询，释放内存
	参数	void *context	上下文
	返回值	0为成功，否则为错误信息	

九、网络端口要求

要求智能网卡支持基于用户态的vdpa框架，实现为虚拟机提供统一标准的virtio网络端口，并支持虚拟机热迁移功能。

OvS创建vdpa端口时，需携带以下参数（vf的pcie号、队列数）：

参数名	值	描述
vdpa-accelerator-devargs	<vf pci id>	端口PCI设备号
vdpa-max-queues	<num queues>	端口队列数

智能网卡厂商提供配置文件及读取脚本，供云平台获取端口

队列数量。

十、存储接口卸载

智能网卡厂商提供PMD driver，云厂商提供SPDK，需要支持以下接口

（一）创建块设备存储卸载控制器

1）接口功能

该接口用于创建块设备存储卸载控制器。

如果操作成功，命令执行完成，如果操作失败，则显示具体的失败原因。

2）接口名称

hw_dpu_rpc.py create_blk_controller [-h/--help] dbdf dev_name -q queues;

3）参数说明

参数	含义	备注
--help/-h	用于显示create_blk_controller命令的帮助信息	
dbdf	表示需要绑定的Host上的VF/PF的dbdf号	将dbdf的最后一个字节映射为VF/PF ID
bdev_name	表示需要绑定的bdev名称	
-q/--queues	用户设置VF/PF队列数	
queues	VF/PF队列数	

Ironic agent通过配置文件获取dbdf列表，配置文件如下：

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<node>

<Queue type=’share’, amount=1024/>//amount是PF的队列总数，目前virtio-net和virtio-blk为共享模式，共享队列模式

<device type=’virtio-net’>

<src id=’1’>//为PF ID信息

<target address=’0000:00:00:00’/>

</device>

<device type=’virtio-net’>

<src id=’2’>

<target address=’0000:00:00:01’/>

</device>

<device type=’virtio-blk’>

<src id=’3’>

<target address=’0000:00:00:02’/>

</device>

</node>

（二）删除块设备存储控制器

1）接口功能

该接口用于删除由create-blk-controller创建的块设备存储卸载控制器。

如果操作成功，命令执行完成，如果操作失败，则显示具体的失败原因。

2）接口名称

hw_dpu_rpc.py delete_controller [-h/--help] dbdf;

3）参数说明

参数	含义	备注
--help/-h	用于显示delete_controller命令的帮助信息	
dbdf	表示需要删除的块设备对应的PF的dbdf号	

（三）获取块设备存储控制器信息

1）接口功能

该接口用于获取由create-blk-controller创建的块设备存储卸载控制器。

如果操作成功，命令执行完成，如果操作失败，则显示具体的失败原因。

2）接口名称

hw_dpu_rpc.py get_controllers [-h/--help] --dbdf/-d dbdf

3）参数说明

参数	含义	备注
--help/-h	用于显示get_controllers命令的帮助信息	
--dbdf/-d	表示其后输入的是dbdf信息，且必须输入dbdf信息	
dbdf	表示需要查询哪个控制器信息	

（四）获取块设备存储控制器的IO状态信息

1）接口功能

该接口用于获取由create-blk-controller创建的块设备存储卸载

控制器的IO状态信息。

如果操作成功，命令执行完成，如果操作失败，则显示具体的失败原因。

2）接口名称

hw_dpu_rpc.py controller_get_iostat [-h/--help] --dbdf/-d dbdf

3）参数说明

参数	含义	备注
--help/-h	用于显示controller_get_iostat命令的帮助信息	
--dbdf/-d	表示其后输入的是dbdf信息，且必须输入dbdf信息	
dbdf	表示需要查询哪个控制器的IO状态信息	

要求支持的state信息如下：

```
"bytes_read": 36864,          /*  读取字节数  */
"num_read_ops": 2,           /*  读取IO数  */
"bytes_written": 0,          /*  写入字节数  */
"num_write_ops": 0,           /*  写入IO数  *
"read_latency_ticks": 178904, /*  读取累积时长(tsc tick) */
"write_latency_ticks": 0,       /*  写入累积时长(tsc tick) */
"completed_read_ios": 0,      /*成功完成的读请求数*/
"err_read_ios": 0,            /*  错误结束的读请求数*/
"completed_write_ios": 0,     /*  成功完成的写请求数*/
"err_write_ios": 0,           /*错误结束的写请求数*/
"flush_ios": 0,               /*  处理的刷新请求数*/
"completed_flush_ios": 0,     /*成功完成的刷新请求数*/
"err_flush_ios": 0,           /*错误结束的刷新请求数*/
"fatal_ios": 0,               /*丢弃未完成的命令数*/
```

（五）扩容块设备存储控制器对应的后端盘

1）接口功能

该接口用于扩容由create-blk-controller创建的块设备存储卸载

控制器对应的后端盘。

如果操作成功，命令执行完成，如果操作失败，则显示具体的失败原因。

2）接口名称

hw_dpu_rpc.py bdev_rbd_resize [-h/--help] dbdf new_size_in_mb

3）参数说明

参数	含义	备注
--help/-h	用于显示delete_controller命令的帮助信息	
dbdf	表示需要扩容设备对应的PF的dbdf号	
new_size_in_mb	表示需要扩容的大小	单位MB

十一、编制历史

版本号	更新时间	主要内容或重大修改	编制人	技术审核人	部门审核人
1.0.0	2022.8.18	初稿	秦凤伟		

算网融合产业及标准推进委员会（TC621）

地址：北京市海淀区花园北路52号

邮编：100191

电话：010-6230XXXX

传真：010-62304980

网址：www.ccnis.org.cn