中国联通 NFV 虚拟化中间件

白皮书

中国联通

2016 年 9 月

目录

1 概述

1.1 问题背景

虚拟化技术是NFV（网络功能虚拟化）的一项重要使能技术，其使得在标准化的通用硬件设备（x86服务器、存储和交换设备）向上提供一致的接口和环境，上层的多种网络设备功能可以透明的使用和访问底层资源。如下图1-1所示，在NFV环境中，虚拟化中间件运行在通用硬件设备和应用之间，虚拟化中间件提供了物理硬件（即通用设备）的虚拟化功能；将物理硬件的计算、存储和网络通信资源进行池化，并以虚拟机和虚拟网络的形式提供给应用（即虚拟化的网元功能VNF）。

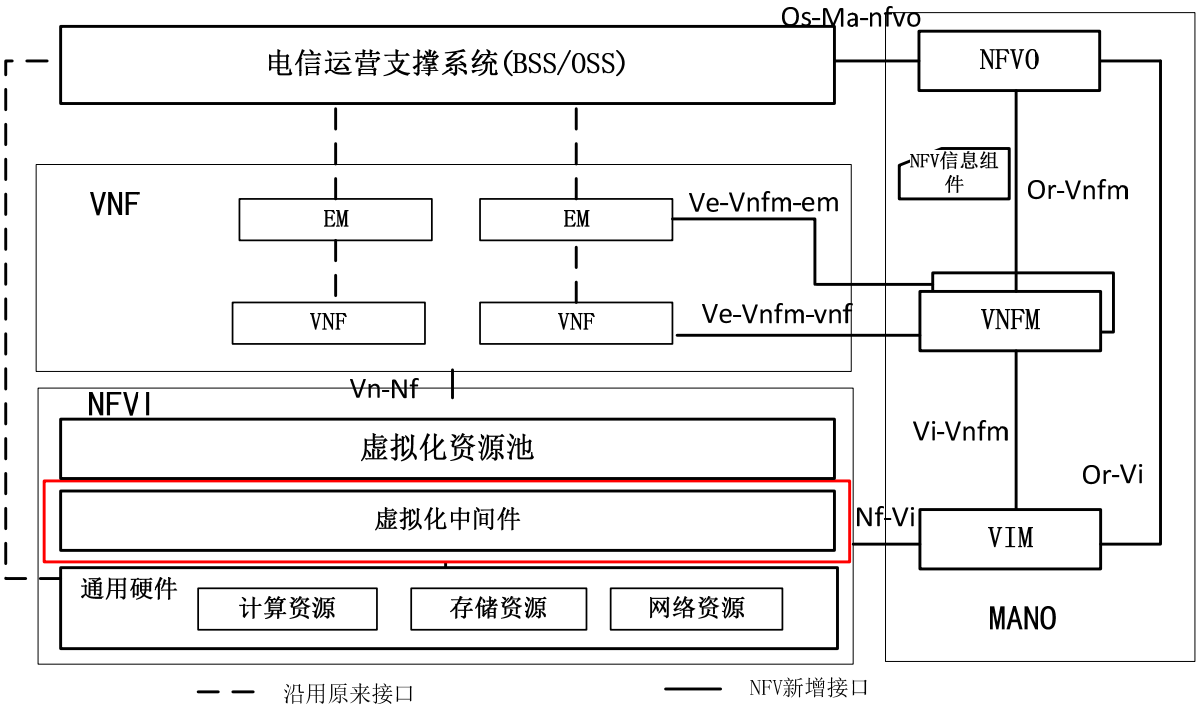

图 1-1 虚拟化中间件在 NFV 逻辑架构中的位置

虚拟化中间件技术来源于IT，其技术发展以IT应用和服务需求为主要驱动力。但随着NFV技术对虚拟化技术的采用，虚拟化中间件技术能否较好适用于NFV电信领域，以及NFV电信领域对虚拟化中间件技术的特殊需求、如何对虚拟化中间件技术和实现进行相关测试等，成为NFV实践和部署中的重要问题。

1.2 范围

本白皮书重点分析NFV对虚拟化中间件的相关技术要求，以及相关指标和测试方法，并结合联通当前已有的工作和未来规划展望未来的发展方向。

本文将提供如下内容：

（1）虚拟化中间件技术的概念和分类

（2）NFV对虚拟化中间件的需求梳理

（3）虚拟化中间件的测试方法、基准和配置

（4）根据联通的实验室测试实践经验，在附录中提供相关操作和配置示例

2 虚拟化中间件简介

2.1 概念描述

虚拟化指的是对某些事物进行虚拟化创建的行为，例如对硬件计算平台、操作系统、存储设备、网络资源等进行虚拟化，其通过抽象出一个虚拟的软件或硬件接口，从而向上层软件提供一个与它原先所期待的运行环境完全一致的接口，最终使得上层软件可以直接运行在虚拟环境上。

在虚拟化中，物理资源通常有一个定语称为宿主机（Host Machine），其上如果运行操作系统，则称为宿主操作系统(Host OS)。虚拟出来的资源通常有一个定语称为客户机(Guest Machine)或虚拟机(Virtual Machine），其上运行的操作系统称为客户操作系统（Guest OS）。

虚拟机可以看作是物理机的一种高效隔离的复制，其蕴含三层含义：

（1）同质。同质指的是虚拟机的运行环境和物理机的运行环境在本质上是相同的，但是在表现上可以有一些差异。例如，虚拟机所看到的处理器个数可以和物理机上实际的处理器个数不同，但是物理机上和虚拟机中看到的处理器必须是统一类型的。

（2）高效。高效指的是虚拟机中运行的软件有接近在物理机上直接运行的性能。

（3）资源受控。资源受控指的是虚拟化中间件需对系统资源有完全控制能力和管理权限，包括资源的分配、监控和回收。

2.2 主流形态

虚拟化中间件（也即虚拟化层或Hypervisor软件）主要表现为两种典型的类型：类型1或者类型2的中间件。在类型1下，虚拟化中间件直接管理调用硬件资源，而虚拟机（包括客户操作系统）直接运行在物理硬件上，所以又被称为裸机（Bare Metal）型。比较典型的例子有VMWare的ESX以及、Citrix的XenServer等。在类型2下，虚拟化中间件运行在宿主操作系统上对硬件资源进行管理，而虚拟机（包括客户操作系统）也运行在宿主操作系统之上，所以又被称为托管型或者宿主型。典型的例子包括VMWare的Workstation，微软的Virtual PC以及甲骨文的VirtualBox等。不过，对于某些中间件系统而言，其类型的差别并不明显，例如，KVM通常既被当作裸机型中间件，又可以被当成宿主型中间件。

此外，目前还出现了一种容器型的中间件方案，虚拟机仍然运行在宿主操作系统（通常是Linux系统）之上，模拟出运行应用程序的容器，所有虚拟机共享宿主操作系统的内核空间。

如图2-1所示，给出了裸机型，托管型的中间件和与容器型中间件的形态结构。

图 2-1 虚拟化中间件的多种形态

不同形态的中间件在性能，安全性和灵活性方面存在差异。

•性能：裸机型中间件因直接工作在硬件层面，性能和效率最高；容器型因虚拟机（应用）共享宿主操作系统核心对硬件资源进行操作，性能和效率较高；而托管型中间件在调用和控制硬件资源时，必须经过客户操作系统和宿主操作系统的核心，相对性能较低。

•安全性和可靠性：裸机型和托管型中间件创建和管理的虚拟机相互之间是隔离的，一个虚拟机故障不会影响到其它虚拟机；容器型中间件因共享宿主操作系统内核，虚拟机之间会产生相互影响。

•灵活性：裸机型和托管型中间件创建和管理的虚拟机可以运行不同的操作系统；而容器型中间件只支持和宿主操作系统相同的虚拟机和应用。

目前没有在性能、安全性和灵活性上都占据绝对优势的虚拟化中间件，无论何种形态的虚拟化中间件，都应该根据NFV实际应用场景满足相应的功能和性能的技术要求。

2.3 NFV 研究现状

在标准化和开源方面，虚拟化中间件并没有形成较为统一的要求规范和部署实践。在ETSI NFV ISG中，与虚拟化中间件标准化相关的立项如下表2-1所示，技术要求规范项目EVE001进展缓慢，而且当前并没有针对虚拟化中间件的相关接口进行立项和规范。在开源组织OPNFV中，针对NFV对中间件电信级需求，进行了相关立项，但是对中间件的功能和性能指标要求，并没有明确定义和规范。

表 2-1 ETSI NFV ISG 中相关立项及进展

立项	项目名称	进展情况
EVE001	虚拟化技术的要求（规范）	进展缓慢
EVE004	虚拟化技术的报告（研究报告）	已发布
IFA003	vSwitch的评测指标和加速要求（规范）	已发布

3 NFV 对虚拟化中间件的特殊需求

尽管NFV的虚拟化中间件大多数是基于云计算技术实现的，但是上层的电信级网络服务和IT服务却不同，电信级网络服务和IT服务的主要区别如下：

•在资源消耗方面，IT服务以计算为中心，其性能主要取决于CPU和内存的数量和能力；而电信级网络服务则以网络和转发能力为中心，I/O的带宽和内存大小是主要的影响因素。

•在业务模型方面，IT服务器一般使用数量较多的小虚拟机(VM)，且虚拟机之间相关性较小；而NFV虚拟化中间件上的电信业务VNF则往往使用数量相对较少的大虚拟机，且各个虚拟机之间相关性较强。这里的大和小指的是虚拟机所的配置的vCPU，内存，虚拟硬盘和虚拟网卡等资源的耗用量。

•在性能要求方面，IT服务只关心可达性，对时延和QoS的要求不高；而电信级网络服务则需要保障端到端的时延和QoS。

由于电信级网络服务和IT服务的差异性，这也导致NFV虚拟化中间件与IT云计算中的虚拟化系统相比较，存在着相当程度的不同。

3.1 高可靠性要求

虚拟化系统自身的可靠性要求。运营商的网络需要保持7x24小时的在线时间和5个9的到可靠性要求。传统电信设备通过软硬件的紧密耦合能够保证系统的可靠性。而采用NFV虚拟化之后，业务、虚拟化平台以及硬件服务器可能都来自不同的厂商，要确保它们搭配在一起能够获得电信级的可靠性就需要每个层面都需要满足一定的可靠性要求。对于虚拟化平台来说，其本身也必须具有高可用设计，以满足整个虚拟化系统的电信级可靠性，比如沿用传统设备的主备模式来设计双控制节点的高可用，实现控制节点多种系统服务主备的独立切换以及关键软件进程故障自我修复。能够快速探测控制节点，计算节点和存储节点的故障，并做相应的业务自动迁移。

对上层业务可靠性机制的支持要求。系统的可靠性也同样依赖于业务本身的可靠性。为此，除了业务设计上的鲁棒性考虑以及主备设计，也需要虚拟化中间件提供能力来保证业务最终的高可靠性。比如：提供计算节点反亲和性功能，避免业务的主备跑在同一个计算节点带来单点故障等；能够快速感知业务虚机的系统故障，并且提供监控虚机内部应用的机制以探测虚机应用级粒度的故障。

对底层硬件监控的监控要求。虚拟化中间件对于物理服务器关键运行参数的监控也有利于整个虚拟化系统的可靠性。通过监测比如CPU温度和风扇速度等指标提前发现硬件潜在问题，从而允许对有潜在问题的服务器提前作出干预，及时迁移业务，同时虚拟化中间件应该支持对业务的快速热迁移，并支持基于DPDK技术的业务的热迁移，以辅助支撑数据面转发加速业务的可靠性需求。

3.2 高性能要求

对加速技术的支持要求。传统电信设备通过应用专用处理芯片以及软硬件紧耦合的设计方式来提升性能指标。而虚拟化之后，采用了通用硬件，业务转而依赖虚拟化平台来提供相应的性能加速，比如报文的加速收发，报文的加解密处理等。有些需要虚拟化平台提供对底层加速芯片的支持，比如加解密，而有些则要求虚拟化平台提供灵活的接口供上层业务使用，比如PCIE设备的pass-through和SRIOV功能，以及基于纯软件虚拟交换机的虚拟化接口的DPDK报文加速处理。

虚拟交换机vSwitch的性能要求。vSwitch应具备动态可扩展的交换能力，以适应不同业务应用的需要。如果上层业务要求平台提供更多的交换能力，那么虚拟交换机可以占用更多CPU核来满足这种需求；反之如果业务主要是用于运算，对网络吞吐没多大要求，则可以通过减少虚拟交换机占用的CPU核数来释放计算能力。另外，增减CPU核数应该可以获得可预测且接近线性的交换能力增减。比如虚拟交换机如果占用一个CPU核可以获得特定报文长度10Gbps的吞吐能力，那么占用两个CPU核的情况下应该可以达到接近20Gbps的吞吐能力。

虚拟化中间件的性能优化要求。虚拟化平台除了在网络处理上要满足业务的性能要求，在运算能力，内存访问能力，以及存储访问能力方面，也需要尽量减少虚拟化平台带来的能力损耗，同时提供策略保证不同的虚机在平台上特别是在同一个计算节点上不会互相干扰，确保业务能获得可预测的，一致性的表现。比如支持CPU核绑定，消除因为CPU核切换带来的性能不稳定因素，提高业务的处理能力；支持分配给虚机专用大页表内存，减少TLB的cache miss带来的虚机性能降低；提供共享和本地方式的虚拟化存储，优化虚拟化存储带宽，以满足业务对磁盘读写性能的不同要求。

虚拟化中间件的策略配置要求。现在主流的COTS服务器都配置双路CPU，而CPU本身也支持超线程，这就要求虚拟化平台能够提供丰富的策略配置如NUMA亲和、PCIE设备NUMA亲和以及超线程亲和等选项。NUMA的策略配置一方面避免跨NUMA对虚拟机性能造成不利影响，另一方面也支持跨NUMA，以充分利用系统的计算资源，避免造成空闲CPU核的浪费。而对于超线程亲和性来说，因为同一物理核上的两个超线程会竞争运算资源，所以希望在虚拟机创建和部署时，平台应该支持把运算量比较大的两个虚拟核分别绑定到不同物理核的超线程上，以保证虚拟机的运算性能。

3.3 高实时性要求

电信网络服务中有一些实时确定性要求强的业务比如实时语音通话，传统上采用实时嵌入式操作系统比如vxworks等来满足对业务实时调度和低延时的需要。但虚拟化以后，在通用硬件以及上层业务应用之间还隔着一个虚拟化平台。如果虚拟化平台不具有实时确定性，这些业务很难实现虚拟化。

虚拟化平台的实时确定性可以从两个方面来强化。一方面，对虚拟化平台的操作系统进行增强，当前虚拟化中间件一般运行在开源的Linux操作系统，linux内核本身实时性较差，虽然社区有实时补丁可供强化，但结合虚拟化，平台提供商还是需要额外做一些开发工作来实现适配。另外一方面对虚拟化中间件自身进行增强，通过优化来提高自身的实时性和调度确定性。

3.4 运维管理的要求

标准规范的要求。运营商网络虚拟化后，除了保证业务的平滑过渡，决定成败关键的还在于整个虚拟化环境的运营维护。为此可能需要运营商花大量的时间来优化内部人员结构，增加软件开发和维护人员的比例，以及设计一套适合虚拟化环境的运维系统。这样一套运维系统构建于ETSI NFV参考架构之上，需要从硬件、中间件系统、业务管理系统到虚拟化协调层以及OSS/BSS等各个层面建立一套标准接口。而这其中，

承上启下的虚拟化中间件系统接口的标准化显得尤为关键和迫切，因为它关系到整个虚拟化生态链是否能够真正打开，运营商是否能够真正避免厂家锁定。目前主流虚拟化中间件系统的虚拟化基础架构管理（VIM）大多基于开源的OPENSTACK项目。OPENSTACK定义了一整套标准REST API用于外部操作，前期主要由IT云公司推动，所以不是特别注重标准的连贯性以及兼容性，同时也存在着在与运营商网络落地的适配性问题。这需要虚拟化中间件提供商以及运营商合力主动推进OPENSTACK REST API的电信标准化。而在这个目标达成之前，要求虚拟化中间件系统尽量遵循已发布OPENSTACK官方版本的标准REST API接口来提供基本和扩展功能，并且保持不同版本接口的兼容性。

服务器的管理要求。虚拟化中间件系统直接管理着规模大小不一的由COTS服务器组成的资源池。为此，需要提供统一的管理界面方便运维人员了解整套物理环境以及虚拟资源的总体状况。能够在不中断业务运行的情况下方便批量添加和更换服务器；能够及时通报物理及虚拟资源故障；能够实时提供网络流量计量数据以及平台和虚机资源动态消耗情况；能够提供详细的故障和运行日志信息以及支持虚接口级别的报文抓取及分析，方便运维故障定位。

升级要求。虚拟化中间件系统的升级必须支持在线升级，可以在基本不影响业务的情况下实现软件大小版本的升级。并且应该做到在最少人工干预的情况下自动完成整个资源池的软件系统升级。而且由于运营商网络的敏感度，升级时间必须能够做到可预测可评估。

3.5 其它要求

电信业务网元比较多，例如包括移动核心网、深度报文探测(DPI)、宽带远程接入服务器(BRAS)、会话边缘控制器(SBC)、安全网关、负载均衡、广域网加速和视频流服务等。不同的网元可能对虚拟化中间件提出不同的要求，除了上面提及的一些比较重要的要求和指标外，还有下面的一些相关要求

DPDK的版本兼容性要求。DPDK技术在NFV中得到了广泛的应用，因为DPDK技术的演进也非常快，很有可能在虚拟交换机上运行的多个业务会采用不同的DPDK版本，这就要求虚拟化中间件提供的虚拟交换机能够支持虚机跑不同版本的DPDK，并能做到加速。

多种组网的灵活配置要求。有些传统电信设备网元可能采用了vlan的子接口方式来共享一个物理端口，虚拟化后就要求虚拟交换机能够收发业务带vlan标签的报文，比如采用QinQ封装等。又或者网元无法直接使用virtio虚接口，这也要求虚拟化平台能够直接模拟这些不同的物理接口类型比如e1000等，减少网元虚拟化的开发工作量。

动态扩缩要求。NFV中电信网络服务的动态扩缩容，需要虚拟化中间件支持并提供横向和纵向动态扩缩容能力，实现在不中断业务的情况下，动态的为业务添加和剥离资源。

4 虚拟化中间件的测试

由于NFV对虚拟化中间件的特殊需求，中国联通网络研究院联合风河、思博伦、惠普和华为等厂家在实验室开展了一系列的验证测试工作。本章节以相关验证测试工作为基础，重点阐述相关测试场景和测试方法，并对典型测试结果进行小结和分析，以作为测试预期的参考。

4.1 测试网络拓扑

如下图4-1所示，为本次测试的逻辑拓扑。思博伦通信的相关测试工具和软件部署在惠普DL360Gen9的机架服务器上，虚拟化中间件系统安装在惠普BL460刀片服务器上（详细配置见4.2.1章节）。两者通过网络连接，思博伦通信的相关软件与中间件系统上的虚拟机进行交互，以执行相应的测试。同时，思博伦通信的物理仪表与C7000背板上的6125XLG交换机连接，用于测试网络性能。

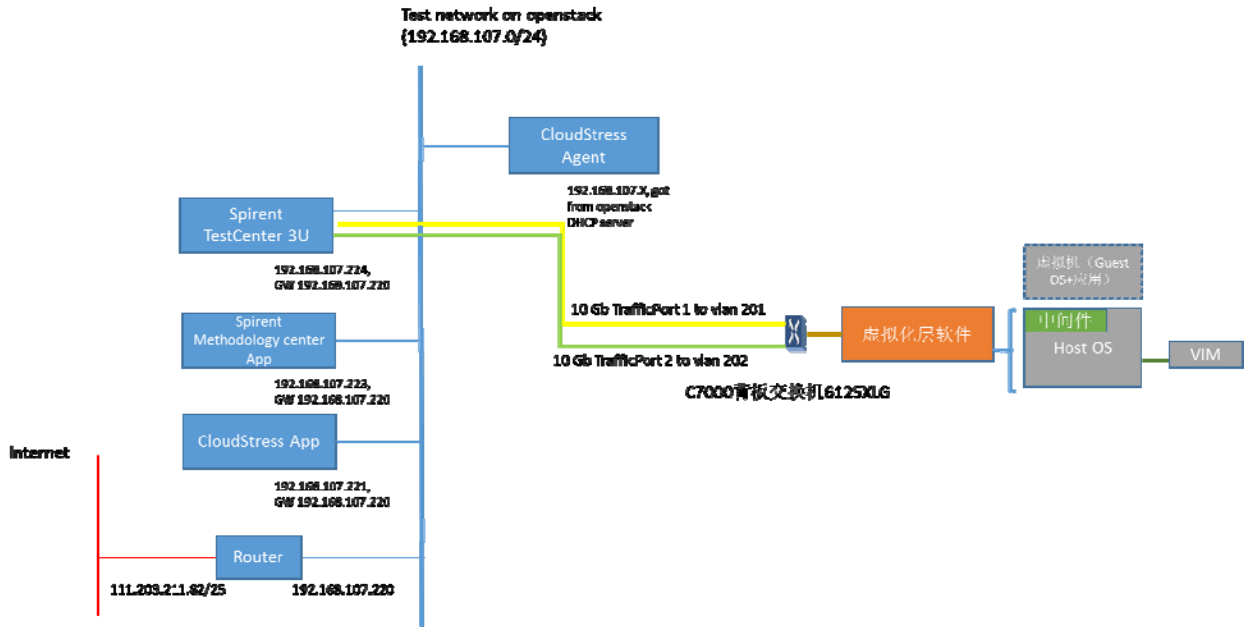

图 4-1 测试的逻辑拓扑图

4.2 测试平台说明

4.2.1 硬件平台说明

本次测试中使用的硬件主要是用于安装虚拟化中间件的BL460刀片服务器(配置相应的C7000刀框)。BL460服务器的配置如表4-1所示。

表 4-1 BL460 服务器配置

实配机型要求	服务器形态	刀片
	实配 CPU 型号	Intel E5-2650 v3 10core
	CPU 数量（实配/最大扩展）	2/2
	总线/通道架构	QPI
	实配内存类型	单条 DDR4-2133 ECC Registered DIMM_16GB
	实配内存容量	192GB
	整机扩展能力	PCI 扩展槽数≥2
实配机型指定配件	硬盘配置	2×600GB/10000/SAS
	RAID 配置	支持 RAID0/1 功能
	网卡配置	2×HP Ethernet 10GB 2-port 560FLB Adapter，支持 DPDK 和 SR-IOV
	光驱配置	无

	电源、风扇配置	热插拔冗余交流电源、热插拔冗余风扇
	USB 接口	无
	SAN HBA 卡	无
	远程控制系统及接口	独立 1 口 10/100 自适应电口,KVM over IP

4.2.2 测试软件说明

本次性能测试中使用的工具包括基本性能测试工具和网络性能测试工具。基本性能测试主要使用Spirent CloudStress，另外辅助使用了开源/免费工具FIO和Intel Memory Latency Checker。网络性能测试使用Spirent TestCenter 3U物理设备。

CloudStress是虚拟机资源性能测试和压力仿真软件工具，可以部署在多种中间件上，通过仿真CPU、内存、存储和网络IO四种指标，来测量虚拟化中间件和NFV基础设施的性能。CloudStress工具由安装CloudStress应用的虚拟机和安装CloudStress Agent的虚拟机两部分组成，采用SAAS方式进行部署和使用。CloudStress Agent部署在测试环境中的计算节点上，CloudStress应用部署在虚拟化环境之外的服务器，并保持与外网Internet的联通。

开源软件FIO用于Guest OS上的各种存储性能测试。免费软件Intel Memory Latency Checker用作Guest OS上的内存干扰测试。这两种测试均以Linux内核虚机为载体。

Spirent TestCenter 3U为物理测试仪表，用于vSwitch以及网卡Passthrough和SR-IOV模式的网络性能测试。该设备为模块化设计，本次使用了10Gb光口作为测试接口，用于测试多种场景下的vSwitch网络性能。

4.3 测试用例及内容

针对NFV对虚拟化中间件的特殊需求，本次测试中对虚拟化中间件的功能、基本性能和网络性能进行相关测试，测试项和需求的对应关系如表4-2所示。

表 4-2 测试项和需求对应列表

测试类别	测试项	对应需求
功能测试	虚拟机亲和性/反亲和性的支持	高可靠性要求
	控制节点故障恢复	高可靠性要求
	虚拟机的高可用性	高可靠性要求
	虚拟机的热迁移（带DPDK）	高可靠性、高性能要求
	故障告警的北向发送	高可靠性要求
基本性能测试	CPU测试	高性能要求
	内存测试	
	存储测试	
	虚拟机干扰测试	
网络性能要求	vSwitch性能测试	高性能要求

4.4 功能测试

4.4.1 虚拟机亲和性/反亲和性的支持

测试方法：通过虚拟化中间件系统设置亲和性/反亲和性策略配置，在同一个计算节点或不同计算节点上创建并启动多个虚拟机。

结果预期：在商业化Openstack中一般可以通过server group的功能来支持该策略的配置。具体地，可以创建相应的server group，指定其策略为Affinity或Anti-Affinity，在虚拟机创建时指明其所属的server group，属于同一个server group的虚拟机会遵照相应的Affinity或Anti-Affinity策略进行启动。

4.4.2 控制节点故障恢复

测试方法：控制节点以1+1/N+1主备或分布式方式部署后，通过软件或硬件模拟相应的故障，导致主控制节点失效。

结果预期：单个主控制节点失效后，其它备控制节点会接管虚拟化资源的管理成为新的主控制节点，同时会恢复主控制节点，恢复期间虚拟机运行不受影响。

4.4.3 虚拟机的高可用性

测试方法：测试虚拟化平台提供的虚拟机高可用机制。测试如下四级故障恢复机制：通过模拟计算节点故障，验证该节点上的虚拟机是否会自动重新启动；通过软件方式关闭计算节点上虚拟化中间件的某个虚拟机进程（异常关闭虚拟机），验证该故障虚拟机是否会重新启动；通过在虚拟机内部模拟内核故障，验证该故障是否会被中间件探测，并触发虚拟机的重启；通过在虚拟机内部模拟业务进程的故障，验证平台是否有能力协助业务的故障检测，并可根据配置的策略来进行相应的动作。

结果预期：虚拟化平台可提供四级虚拟机高可用防护。

4.4.4 虚拟机的热迁移（带 DPDK）

测试方法：利用pktgen工具向虚拟机（启动DPDK）打双向流量，设定相应的速率和包大小，通过虚拟化中间件进行虚拟机的热迁移，待迁移完毕后，停止打流。然后根据热迁移过程中的丢包数计算虚拟机迁移时间。

结果预期：虚拟机热迁移的时长在一个被允许的范围内。

4.4.5 故障告警的北向发送

测试方法：虚拟化平台支持通过SNMP协议进行系统告警信息的发送。在故障告警接受和发送两方设置相应的SNMP tap。另外，也验证平台是否提供REST API接口供外部监控提取系统故障告警信息，利用curl工具来模仿外部监控验证REST API交互正常进行。

结果预期：改变虚拟机的状态，在故障告警接收方可以看到相应的SNMP告警消息，通过curl工具也可主动拿到当前的故障告警信息。

4.5 基本性能测试

4.5.1 测试场景

虚拟化中间件基本性能的测试包括四个场景：CPU测试、内存测试、存储测试和虚拟机干扰测试。具体场景如下：

（1）CPU测试

在给定抖动限制条件下，测量同一物理服务器上多个虚拟机的多个vCPU能同时达到的最高CPU利用率，且验证不会因为CPU压力过大引起虚拟化中间件的故障。

（2）内存测试

在给定抖动限制条件下，首先测量单个虚拟机内存在常见读写比例情况下的读写带宽。作为对比，还测试了另外两个场景：内存大小不同的虚拟机配置和vCPU个数不同的虚拟机配置。

（3）存储测试

首先测量单个虚拟机存储在常见读写块大小、读写比例情况下的读写带宽及IOPS。作为对比，还测试了另外两个场景：

1.存储容量大小不同的另一种虚拟机配置，进行存储测试；

2.vCPU个数不同的另外两种虚拟机配置，进行存储测试。

考虑到存储系统的复杂性和多样性，本次仅测试了虚拟机磁盘位于本地磁盘的情况。对于虚拟机磁盘位于共享存储的情况，类型较多且复杂，不在本次测试内容之内。

（4）虚拟机干扰测试

为了测试虚拟机之间的相互干扰，选取了3个场景：测试多个虚拟机的内存性能指标在出现资源竞争情况下的性能表现情况；测试多个虚拟机的磁盘性能指标在出现资源竞争情况下的性能表现情况；模拟正常运行的一台虚拟机在受到同一物理服务器上多个噪声邻居干扰的情况下的性能表现情况。

4.5.2 测试方法

考虑到NFV环境中对性能和稳定性要求较高，因此每个vCPU和物理CPU的逻辑核进行绑定，从而提高vCPU的性能。本次基本性能测试选取的典型虚拟机配置为4个vCPU（核绑定），16G内存，4G磁盘存储。所有的CPU，内存和存储测试都基于此配置。

虚拟化环境中，同一物理服务器上的多个虚拟机共享物理CPU、内存、存储和网络等多种设备，因此可能会出现资源争用的情况。此时各种性能指标可能会出现较大抖动，甚至影响上层业务稳定运行。所以在测试条件中，加入了性能指标抖动方面的限制要求。

（1）CPU测试

本次CPU测试中，同一物理服务器上部署了3个虚拟机，每个虚拟机的4个vCPU均位于同一个NUMA节点上。使用压力测试工具将3个虚拟机共12个vCPU同时加压，压力的持续时间不小于60秒。观察其能达到的最高CPU利用率，并注意观察CPU利用率的抖动情况。

这种多个虚拟机同时高CPU压力的情况对于底层的虚拟化中间件和物理服务器也是非常大的考验。因此，本测试在测量虚拟机CPU利用率的同时，还可以验证虚拟化中间件及物理设备在高压力下的稳定性和兼容性。

（2）内存测试

本次内存测试主要考察典型配置虚拟机（4个vCPU，16G内存，4G磁盘存储）在1:1及3:1随机比例内存读写情况下的读写带宽。选取随机读写方式主要是为了尽量避免各种缓存带来的影响。

为了验证和展现虚拟机其它配置指标例如vCPU数量及内存容量大小对内存性能的影响，又选择了2个场景作为性能对比测试：

1.内存大小不同虚拟机配置。测试2G内存容量的虚拟机在1:1及3:1随机读写两种情况下的读写带宽；

2.vCPU数量不同虚拟机配置。测试10vCPU虚拟机在1:1及3:1随机读写两种情况下的读写带宽。

根据典型配置虚拟机以及对比虚拟机的内存测试结果，可以对现实中内存敏感型虚拟机给出配置方面的参考建议。

（3）存储测试

本次存储测试主要考察典型配置虚拟机（4个vCPU，16G内存，4G磁盘存储）在4k/64k/256k块大小、1:1及3:1随机存储读写两种情况下的读写带宽。读写块大小，读写比例均选择比较典型的配置。

由于VNF的存储容量通常不会太大，因此本次存储测试的容量大小也比物理存储测试的容量要小很多。为了验证虚拟机其它配置指标例如vCPU数量及存储容量大小对虚拟机存储性能的影响，又选择了2个场景作为性能对比测试：

1.存储大小不同（16G本地磁盘）的另一种虚拟机配置，测试虚拟机存储在4k块大小、1:1及3:1随机读写两种情况下的读写带宽；

2.vCPU数量不同（10vCPU）的另两种虚拟机配置，测试虚拟机存储在4k块大小、1:1及3:1随机读写两种情况下的读写带宽。

本次测试的虚拟机存储磁盘位于本地磁盘（1块600G,10k转速SAS硬盘做RAID0，硬件RAID卡缓存1G），主要考虑的是方便不同虚拟中间件厂商的测试结果做对比。在真实的部署环境中，虚拟机磁盘一般位于各式各样的共享存储中，各家厂商的解决方案也呈现多样化，因此很难进行简单的横向比较。

本次测试的物理存储介质为SAS硬盘，因此在虚拟机性能方面也会体现出一定的SAS硬盘特征。对于物理存储介质为完全SSD硬盘、SSD硬盘混合SAS硬盘、对象存储、磁盘阵列等诸多不同场景，测试结论可能又会截然不同。

（4）虚拟机干扰测试

虚拟化的主要特征之一是资源池化和共享。资源共享带来的问题之一就是可能会出现资源争用和干扰。这一问题对于NFV的影响非常大。VNF的性能表现是否稳定，VNF之间的过度资源共享会带来什么样的结果，是这本测试项的主要研究目标。

虚拟机干扰测试的3个场景具体情况为：

第一个场景为内存干扰测试：先将2个典型配置虚拟机部署于同一NUMA节点。同时执行内存性能测试，测量每个虚拟机能获得的最大读写带宽，和单独一个虚拟机运行时的数据做对比；再将2个典型配置虚拟机部署于不同NUMA节点。同时执行存储性能测试，测量每个虚拟机能获得的最大读写带宽，和单独一个虚拟机运行时的数据做对比。

第二个场景为存储干扰测试：先将2个典型配置虚拟机部署于同一NUMA节点。同时执行存储性能测试，测量每个虚拟机能获得的最大读写带宽，和单独一个虚拟机运行时的数据做对比；再将2个典型配置虚拟机部署于不同NUMA节点。同时执行存储性能测试，测量每个虚拟机能获得的最大读写带宽，和单独一个虚拟机运行时的数据做对比。

第三个场景为噪声邻居测试：先将1个典型配置虚拟机同时施加特定的CPU，内存，存储及网络压力，用于模拟一个VNF正常运行时的状态。再使用另外3个虚拟机，用于模拟噪声邻居。其中1个模拟CPU消耗型应用，将其CPU加压到接近100%；第2个模拟内存消耗型应用，将其内存读写加压到较高值；第3个模拟存储消耗型应用，将其存储读写加压到较高值。这3个虚拟机同时加压，此时观察第1个虚拟机的CPU，内存，存储及网络性能指标是否有变化。

4.5.3 结果预期

（1）CPU测试

预期一：虚拟机的CPU利用率可以同时达到90%以上；

预期二：虚拟机CPU的性能波动在可接受的范围内；

预期三：整个测试过程中，虚拟化中间件表现平稳，没有出现死机或故障重启等故障。

（2）内存测试

预期一：虚拟机的内存容量大小对内存带宽性能无明显影响；

预期二：虚拟机vCPU数量对内存带宽性能较为明显，vCPU数量越多，内存带宽性能越好。

（3）存储测试

预期一：vCPU个数对存储性能无明显影响，存储容量大小对存储有一定影响（底层介质为机械磁盘的情况下）；

预期二：存储测试受到了比较明显的RAID卡缓存的影响，随机写的延迟较低。

（4）虚拟机干扰测试

预期一：同一NUMA节点的两个虚拟机同时执行内存随机读写测试时，内存读写带宽与单个虚拟机性能相比有所下降，相应的读写延迟也有所增加；不同NUMA节点的两个虚拟机同时执行内存随机读写测试结果与单个虚拟机性能基本相同。总地说来，同一NUMA节点的虚拟机内存性能存在相互影响；

预期二：同一NUMA节点的两个虚拟机同时执行存储随机读写测试时，存储读写带宽与单个虚拟机性能相比有所下降，相应的读写延迟也有所增加；不同NUMA节点的两个虚拟机同时执行内存随机读写测试结果与同一NUMA节点的两个虚拟机相似。总地说来，NUMA节点位置对于存储性能影响不明显；

预期三：虚拟机的CPU和内存性能指标在噪声邻居运行前后不易出现变化和波动，但存储和网络指标易出现变化和波动。

4.6 网络性能测试

4.6.1 测试场景

虚拟化平台提供了多种网络加速技术，例如基于DPDK的软件虚拟交换机技术（vSwitch）、网卡直接透传技术（PCI-passthrough）和网卡SR-IOV技术等。基于DPDK的软件交换技术（vSwitch）因为其部署灵活以及高性能而受到更多的关注，所以这次测试主要验证这种加速技术，同时作为对比，又基于DPDK应用，测试一组网卡透传技术的数据，作为软交换技术的参照对比。

在vSwitch和PCI-Passthrough的网络性能测试中，使用思博伦通信的Spirent TestCenter 3U机箱及万兆接口测试板卡加载流量，上层VNF由DPDK开源驱动所带的l2forward进行模拟。vSwitch测试所用VNF由windriver提供,基于DPDK2.0.0的layer2forward进行模拟；

Spirent Testcenter 使用两个万兆测试接口，分别连接到惠普BL460刀片服务器上的两个万兆网络接口，这两个物理接口属于同一个物理网卡，测试流量经过背板交换机6125X进入到刀片服务器的82599物理网卡及刀片服务器内部，vSwitch将测试流量转发到VNF进行处理。同样连接到惠普BL460刀片服务器的Spirent TestCenter测试接口对收到的流量进行统计。

在本次测试中，考虑到VNF实际部署中存在单个VNF和多个VNF同时使用组成SFC的应用场景，这里设立了单VNF转发和双VNF转发两种场景。同时，由于物理服务器上存在两个NUMA节点，即物理网口、vSwitch以及VNF之间存在跨NUMA节点的情况，这对网络性能会产生影响，所以在测试中需要考虑跨NUMA和不跨NUMA的情况。对于不跨NUMA的情况，其测试拓扑较为简单，物理网口、VNF以及vSwitch都属于同一个Socket。对于跨NUMA的情况，单VNF和双VNF转发场景的拓扑结构分别如图4-3和4-4所示。

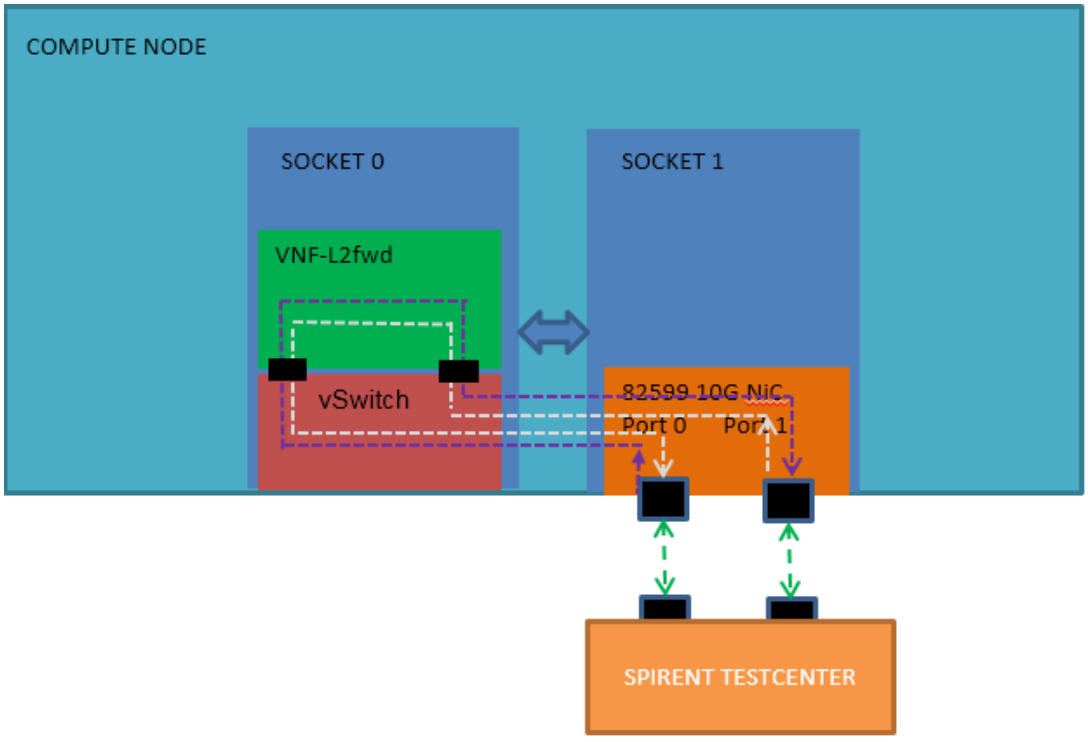

图 4-3 单 VNF 以及跨 NUMA 场景下 vSwitch 的性能测试拓扑

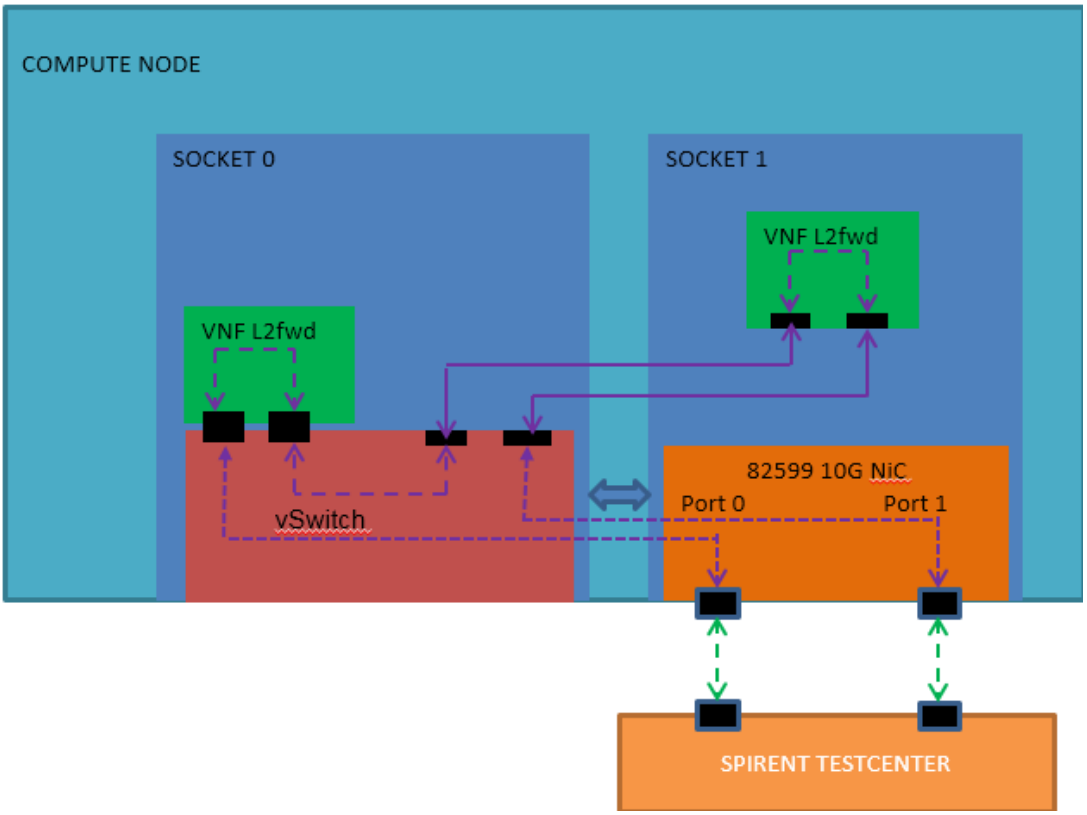

图 4-4 双 VNF 及跨 NUMA 场景下 vSwitch 的性能测试拓扑

此外，为了更好的对比和测试，在场景构建中还需要考虑如下因素：

（1）设定单一VNF下PCI-Passthrough的测试场景，以方便比较vSwitch与PCI-Passthrough场景的性能差异；

（2）考虑到不同电信网络服务对网络QoS要求的差异性，建议设立不同QoS场景的考虑；

（3）考虑到真实环境中多报文的共存，建议增加对混合包长场景的考虑；

（4）考虑到vCPU数量对vSwitch的性能影响，建议增加多个vCPU数量的场景对比测试。

综上所述，测试场景总体设立如表4-3所示。每一种场景下，单VNF、双VNF以及PCI-Passthrough中，均要测试在跨NUMA和不跨NUMA两种情况下的相关网络性能指标和数据；单VNF和双VNF中，均要额外测试给vSwitch分配不同vCPU数量（比如单核和双核）时的相关网络性能指标和数据。

表 4-3 vSwitch 性能测试中的场景设置

场景编号	场景描述	单一VNF	双VNF	PCI-Passthrough
A1	双向单流，单一包长，0.1% 丢包容忍度	√	√	√
A2	双向单流，单一包长，0.01% 丢包容忍度	√	√	√
B1	单向单流，单一包长，0.1% 丢包容忍度	√	√	√
B2	单向单流，单一包长，0.01% 丢包容忍度	√	√	√
D1	双向单流，混合包长，0.1% 丢包容忍度	√	√	√

D2	双向单流，混合包长，0.01% 丢包容忍度	√	√	√

4.6.2 测试方法

参考RFC2544测试方法， 验证由vSwitch处理转发的使用场景下，以80%最大转发处理能力为测试基准，测试平均时延和最大时延指标。在测试过程中，建议对vSwitch进行数据转发所引起的时延抖动情况进行分析，抖动的测试方法依照RFC3393标准，在测试过程中实时进行采集。结合时延和抖动数据结果所进行的分析，为部署不同VNF应用所要求的基础网络情况找到依据。

（1）测试步骤及方法：

1.双向数据流量IO性能测试采用标准RFC-2544测试方法学建议配置，单向数据流量IO性能测试采用可以单向迭代的非对称吞吐量测试方法完成测试，为避免MAC地址老化问题，测试过程中采用反向数据帧持续发送，发送速率控制在1fps；

2.每次迭代测试数据流发送时间设定为60s；

3.测试过程采用64B、128B、256B、512B、IMIX混合包长共计5组包长情况进行测试；

4.IMIX混合包长比例参考核心路由交换机测试中使用的包长混合比： 64Bytes 15% ，128Bytes 20% ，512Bytes 20% ，1514Bytes 45%；

5.分别测试丢包率容忍度为0.1%和0.01%两种情况下的数据IO性能；

6.吞吐量测试过程中同时采用RFC3393方式，记录数据包的抖动及时延信息。

（2）测试指标：

1.丢包率：在测试周期内，测试流量总丢包个数与总发送报文个数的比例。

2.吞吐量：在预先设定的可接受丢包率范围内，被测设备所能处理的最大数据帧/包转发速率的最大数据流量。通常使用每秒钟通过的最大的数据包数或者比特速率数。为直观体现测试结果，本报告使用吞吐量与理论最大速率的比例数值表述。

3.最小时延：测试流量报文经过测试环境所经历的最小时间。

4.平均时延：测试周期内全部测试流量经过测试环境所经历时间的算数平均值。

5.最大时延：测试流量报文经过测试环境所经历的最大时间。

6.最小抖动（RFC-3393）：任意两个相邻报文之间转发时延的最小变化量。

7.平均抖动（RFC-3393）：测试周期内测试流量统计的抖动数值的算数平均值。

8.最大抖动（RFC-3393）：任意两个相邻报文之间转发时延的最大变化量。

4.6.3 结果预期

（1）所有场景下单VNF，vSwitch分配不同核数量的转发能力测试

预期一：在单VNF配置场景下，vSwitch 配置核数量多少与转发性能的正相关性；

预期二：PCI-Passthrough模式对于在256字节以下小包转发性能提升效果明显， 在128字节长度报文转发上即可以实现90%及以上的线速转发性能；

预期三：当给vSwitch分配足够数量vCPU时,DPDK对512字节大包和混合包长转发能力与PCI-Passthrough性能接近。

（2）所有场景下单VNF，vSwitch分配不同核数量的时延和抖动测试

预期一：增加vSwitch CPU核数，能够降低数据的转发处理时延；

预期二：PCI-passthrough 在平均转发时延和平均抖动方面较DPDK具有优势，在最大抖动指标上由于中断处理机制可能导致出现较大时延；

（3）所有场景下双VNF，vSwitch分配不同核数量的转发能力测试

预期一：数据IO性能测试结果能够体现出vSwitch 配置核数量多少与转发性能的正相关性；

预期二：双VNF下，如果vSwitch分配的vCPU数量不足时，会表现出转发性能下降。

（4）所有场景下双VNF，vSwitch分配不同核数量的时延和抖动测试

预期一：数据转发时延与单VNF情况比较将增加不少于30%；

预期二：80%最大吞吐量下运行时，时延及抖动指标仍能保持稳定。

（5）所有场景下跨NUMA和不跨NUMA的网络性能对比测试

预期一：对于小包如64字节，不跨NUMA能够比跨NUMA的情况转发能力提高20%以上

预期二：不跨NUMA能够比跨NUMA的情况有更短的时延

5 启示与展望

5.1 中国联通虚拟化中间件测试的启示

当前，中国联通在实验室环境中对虚拟化中间件（同时还包括VIM）进行了功能和性能测试。其中在功能方面，重点关注其可靠性，以保证上层电信网络服务的高可用；在性能方面，重点关注vSwitch的网络转发性能和时延抖动，以保证上层电信网络服务的端到端QoS。

根据当前对风河等厂家的测试经验，有如下启示：

（1）测试工具和参数配置的统一十分重要。在测试中，可选的开源和商用工具很多，对应的测试和度量方法差异较大，为避免出现结果分歧，保证测试工具和配置的统一对测试的公平性和准确性来说十分重要。本次测试中同时采用了商用和开源工具，并对具体的版本和配置做了要求。

（2）刀框背板交换机存在网络性能瓶颈和丢包。本次测试中，采用刀片服务器，测试仪表需要与背板交换机的物理端口连接后才能与服务器进行通信，由于背板交换机自身性能会对测试产生影响，进而给测试带来不确定性。本次测试中通过单独对背板交换机进行量化评估，以确定和剔除其影响。

（3）磁盘IO性能的实际测试结果与预期不符。一定条件下，读相对写的比例越大，磁盘IO性能表现越好，但在实际测试中出现和预期相反的结果，初步诊断为磁盘类型和RIAD卡的问题。

5.2 中国联通对未来中间件的考虑

通过对业界主流虚拟化中间件的已有相关调研和测试，虚拟化中间件未来工作的重点建议如下：

（1）在功能方面

对底层设备的管理能力有待加强。当前VIM对底层设备的管理并没有较好的支持，这给NFV部署后的运维管理带来复杂性，需要虚拟化中间件厂家扩展对异厂家底层设备的基本信息、故障和性能的管理能力，便于NFVI基础设施的运维管理和容量规划等。

网络虚拟化能力的增强。同计算和存储虚拟化技术相比，网络虚拟化技术还比较落后，SDN尚不成熟，目前网络虚拟化技术类型繁多，如何整合到NFVI中是一个较大风险。对于电信业务网络来说，由于通常是一个分布式网络，因此需要配置较大的网络资源来承载，这种网络资源需要分解到数据中心内部的局域网络资源和数据中心间的承载网络资源，业务网络与接入网络间的承载网络资源等，承载网络资源分配又可能涉及到传送网络资源分配；这些资源的分配都需要做到虚拟化，自动化，目前这种分配尚需要通过承载网/传送网网管来进行，距离自动化尚有较大距离。

（2）在兼容性方面

虚拟化中间件向下需要与多厂家的硬件解耦，实现对底层不同硬件的虚拟化；向上则提供统一的接口，使得上层电信业务对底层硬件资源访问的透明化和使用的无感知。

（3）在性能方面

vSwitch性能有待进一步提升。虽然当前商业化产品对vSwitch做了相关优化，但其性能在大部分场景中还是远落后与SR-IOV和PCI-Passthrough，SR-IOV这种方式存在虚机无法迁移的问题而PCI-Passthrough则因为网卡独占带来资源利用率低下的问题，vSwtich虽然更为灵活，但在性能上需要进一步提升。

存储性能需要进一步关注和验证。当前存储类网元，例如HSS网元等，其NFV化尚不成熟，也是虚拟化中间件当前尚未严格认证和进一步优化的地方。

6 缩略语

缩略语	英文	中文
CPU	Central Processing Unit	中央处理器
DPDK	Data Plane Development Kit	数据面转发工具集
DPI	Deep Packet Inspection	深度包检测
EM	Element Manager	网元管理器
HA	High Availability	高可靠性
HBA	Host Bus Adapter	主机总线适配器
NFV	Network Function Virtualisation	网络功能虚拟化
NFVI	NFV Infrastructure	网络功能虚拟化基础设施
NUMA	Non Uniform Memory Access Architecture	非一致性内存访问
MANO	Network Function Virtualisation Management and Orchestration	网络功能虚拟化管理及（服务）编排
NFVO	Network Function Virtualisation Orchestrator	网络功能虚拟化编排器
NIC	Network Interface Controller	网络接口控制器
OS	Operating System	操作系统
PoP	Point of Presence	接入点/服务提供点
SFC	Service Function Chaining	业务链
SLA	Service Level Agreement	服务水平协议
SPOF	Single Point of Failure	单点故障
SR-IOV	Single-Root I/O Virtualization	单根 I/O 虚拟化
vCPU	Virtualised CPU	虚拟化的中央处理器
VIM	Virtualised Infrastructure Manager	虚拟化的基础设施管理器
VM	Virtual Machine	虚拟机
VNF	Virtualised Network Function	虚拟化的网络功能模块
VNFC	Virtualised Network Function Component	虚拟化的网络功能模块组件
VNFM	Virtualised Network Function Manager	虚拟化的网络功能模块管理器
vSwitch	Virtualized Switch	虚拟交换机

7 参考文献

[1] RFC 2544, https://www.ietf.org/rfc/rfc2544.txt
[2] ETSI Network Functions Virtualisation (NFV); Architecture Framework
[3] http://www.opnfv.org/
[4] RFC 3393, https://www.ietf.org/rfc/rfc3393.txt
[5] ETSI GS NFV-TST 001;Pre-deployment Testing; Report on Validation of NFV Environments and Services
[6] IETF RFC 6985：IMIX Genome.

8 鸣谢

本文撰写和测试工作中，风河、惠普和华为等提供相关测试设备和技术支持，思博伦等提供相应的测试方法建议和测试仪表工具，对这些单位的合作与支持表示衷心感谢！

附录

附录一：DPDK 编译

1.从DPDK官方网站(dpdk.org)下载dpdk 16.07开源软件，然后拷到centos 7.2 guest OS系统上。

2.解压缩到指定目录比如/root/dpdk-16.07，可执行/root/dpdk-16.07/tools/dpdk-setup.sh，然后选择x86_64-native-linuxapp-gcc进行配置编译。

3.创建/root/ dpdk-16.07-env.sh如下，然后执行source /root/ dpdk-16.07-env.sh。以方便后续编译l2fwd样例。

```
[root@host-192-168-222-11 l2fwd]# cat /root/dpdk-16.07-env.sh
export RTE_SDK=/root/dpdk-16.07/
export RTE_TARGET=x86_64-native-linuxapp-gcc
```

附录二：l2fwd 程序编译

1.修改编译l2fwd代码（这一部分供在同一张10G接口卡上的两个物理接口做PCIE pass-through性能测试时用）

用下面替换原来l2fwd的l2fwd_simple_forward函数：

```
static void
l2fwd_simple_forward(struct rte_mbuf *m, unsigned portid)
{
        struct ether_hdr *eth;
        void *tmp;
      unsigned dst_port;
      int sent;
      struct rte_eth_dev_tx_buffer *buffer;
      uint16_t vid[2]={htons(1001),htons(1101)};

      dst_port = l2fwd_dst_ports[portid];
        eth = rte_pktmbuf_mtod(m, struct ether_hdr *);
```

```
        /* 02:00:00:00:00:xx */
//      tmp = &eth->d_addr.addr_bytes[0];
//      *((uint64_t *)tmp) = 0x000000000002 + ((uint64_t)dst_port << 40);

        tmp = &eth->ether_type;
        *((uint32_t *)tmp) = 0x00000081 + ((uint32_t)vid[dst_port] << 16);

/* src addr */
//      ether_addr_copy(&l2fwd_ports_eth_addr[dst_port], &eth->s_addr);

        buffer = tx_buffer[dst_port];
        sent = rte_eth_tx_buffer(dst_port, 0, buffer, m);
        if (sent)
                port_statistics[dst_port].tx += sent;
}
```

与原来代码的区别如下：

```
[root@host-192-168-222-11 l2fwd-tweak]# diff ../l2fwd/main.c ./main.c
195a196
>       uint16_t vid[2]={htons(1001),htons(1101)};
201,202c202,203
<       tmp = &eth->d_addr.addr_bytes[0];
<       *((uint64_t *)tmp) = 0x000000000002 + ((uint64_t)dst_port << 40);
---
> //    tmp = &eth->d_addr.addr_bytes[0];
> //    *((uint64_t *)tmp) = 0x000000000002 + ((uint64_t)dst_port << 40);
204,205c205,209
<       /* src addr */
<       ether_addr_copy(&l2fwd_ports_eth_addr[dst_port], &eth->s_addr);
---
>       tmp = &eth->ether_type;
>       *((uint32_t *)tmp) = 0x00000081 + ((uint32_t)vid[dst_port] << 16);
>
> /* src addr */
> //    ether_addr_copy(&l2fwd_ports_eth_addr[dst_port], &eth->s_addr);
```

注意：Vid[2]需要根据实际的每个端口的划分vid来改，并且在绑定的背板交换机该vid需要配置为tagged模式；范例表示port 0以tagged方式接入VLAN 1001，而port 1以tagged方式接入VLAN 1101。

然后如下编译，

```
source /root/dpdk-16.07-env.sh
cd /root/dpdk-16.07/examples/l2fwd
make
```

2. 执行dpdk l2fwd

创建下述文件/root/run.16.07.sh，然后执行chmod a+x /root/run.16.07.sh，执行这个脚本配置DPDK执行环境(假定需要用到两个DPDK数据端口ens5和ens6)：

```
[root@host-192-168-222-11 ~]# cat /root/run.16.07.sh
#/bin/sh
modprobe uio
insmod /root/dpdk-16.07/x86_64-native-linuxapp-gcc/kmod/igb_uio.ko
/root/dpdk-16.07/tools/dpdk-devbind.py --bind=igb_uio ens5 ens6
mkdir /mnt/huge
mount -t hugetlbfs nodev /mnt/huge
echo 512 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

然后进入l2fwd目录，执行l2fwd如下:

```
cd /root/dpdk-16.07-rc4/examples/l2fwd
./build/l2fwd –c 7 –n 4 -- -p 0x3
```

可能的问题：l2fwd在编端口序号有一定的随机性，也就是第一个物理端口（如ens5）有可能是port 1，而第二个物理端口有可能反而是port 0。这个需要根据执行l2fwd时打印的log中显示的端口MAC地址来确定映射关系。