计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(2):51−57 [doi: 10.15888/j.cnki.csa.006163]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Openstack 虚拟化流量平台监控系统①

徐　毅, 曾文兵

(电子科技大学 电子科学技术研究院, 成都 611731)
通讯作者: 徐　毅, E-mail: 18384205780@163.com

摘　要: 云平台监控系统是有效保障云服务质量的重要环节, 本文在 Openstack 云平台上进行相应的虚拟服务器监控系统设计. 系统结合 Openstack 的开源性和强大的可扩展性的特点, 以 Openstack 中的网络组件 Neutron 面对庞大虚拟服务器流量转发时所出现的瓶颈问题为出发点, 结合 SDN 软件定义网络技术及其重要的南向接口协议 OpenFlow 解决流量瓶颈问题, 设计基于 Libvirt 和 sFlow 协议的监控系统, 负责获取虚拟机群数据流量信息, 并且向上层控制模块以及用户反馈最新的平台负载情况, 使用控制模块和流表控制流量转发, 最终使整个系统平台达到负载均衡.

关键词: Openstack; Neutron; SDN; 流量监控; Libvirt

引用格式: 徐毅,曾文兵.Openstack 虚拟化流量平台监控系统.计算机系统应用,2018,27(2):51–57. http://www.c-s-a.org.cn/1003-3254/6163.html

Openstack Virtualization Platform Traffic Monitoring System

XU Yi, ZENG Wen-Bing

(Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Cloud platform monitoring system is an important segment to effectively guarantee the quality of cloud services. In this study, the corresponding virtual server monitoring system is designed on the Openstack cloud platform. The system combines the characteristics of Openstack's open source and strong scalability. At the same time, it starts with the bottleneck problem which appears when Neutron, the network component in Openstack faces the huge virtual server traffic forwarding. This study combines SDN technology and openflow, its important southward interface protocol, to solve the traffic bottleneck problem. At the same time, it designs the monitoring system based on Libvirt and sFlow protocol responsible for obtaining data flow of virtual cluster, giving feedbacks of the latest platform load to the upper control module and user, and using control module and flow table to control traffic forwarding. Finally, the whole system platform can achieve load balancing.

Key words: Openstack; Neutron; SDN; traffic monitoring; Libvirt

云计算[1]是计算机领域继八十年代时期大型计算机向客户端-服务器转变后的又一次重要技术革新, 其不但继承了并行计算、分布式计算、网格计算等以往高效的计算方式的优点, 同时融入了越发成熟的虚拟化技术.

面对着信息时代再一次的大力度提速, 云计算能够整合大量普通的基础设施, 运用着虚拟化技术构成一个庞大的资源池, 用户不必要关心底层具体的实现方式, 即可按需获取计算能力和存储等各种服务. 经过接近十年的不断改革与发展, 云计算已经不再是

① 收稿时间: 2017-03-20; 修改时间: 2017-04-10; 采用时间: 2017-05-11

Google、亚马逊、IBM 等公司的专属产品了, 其正在走向普遍化, 并且渗透进入着各行各业的发展之中. 构建自己的云平台已经成为企业发展的必经之路.

Openstack 是当下相当流行和优秀的云平台建设开源项目, Neutron 是 Openstack 中处理网络流量的重要模块. 由于自身设计的局限, 当 Neutron 面对过于庞大的网络流量和来自客户对高效性的要求时, 其自身的网络流量瓶颈也就越发的明显. 现今成熟的云平台将面对着成百上千的物理服务器和更加庞大的虚拟机群, 其日常产生的通信流量将是非常的巨大. 如此, 解决 Neutron 的流量瓶颈就变得迫在眉睫. 与此同时, 合理运用对虚拟流量的有效监控和对 SDN 著名的 Openflow 协议的有效解读, 将为我们在理论上解决前面的问题提供坚实的基础.

面对着现今越发庞大的云系统架构, 对于监控系统的研究也逐步成为信息计算机行业的重点. 国内外现今较为流行的云监控产品十分丰富, 国外首当其冲的就是亚马逊的云监控服务 CloudWatch, 其次还有能够提供网络、服务、应用等多种监控的 Monitis 软件. 国内较为知名的云监控产品有阿里云监控、基于网站综合性能进行监控的监控宝、以及具有较好伸缩性的 CreCloud 云网管等等. 但是市场上这些具有成熟框架的监控产品, 大多是基于物理平台资源的, 或则是针对网站性能和特定云平台进行监控的. 同时对于虚拟化平台资源的监控并没有形成一个成熟的解决方案, 在虚拟化平台资源和 Openstack 云平台的监控上, 业内学者也进行了相应的研究, 但在监控方面主要是利用以往运用在传统物理平台上的监控软件进行二次配置, 如 Nagios, MRTG, colledtd 等等, 但是其中绝大多数方案都需要通过在被监控的虚拟机上布置监控代理等, 这无疑将加大虚拟机的负载, 降低服务质量.

本文系统中通过开源的 LibvirtAPI 直接部署在计算节点宿主机上的方式来获取虚拟服务器的状态信息, 替代了原来需要在各被监控虚拟机上布置插件的方式, 由此减轻了虚拟服务器的压力, 提高了其 QoS. 同时利用轻量级的开源虚拟交换机 Openswitch 对 sFlow 和 Openflow 协议的支持, 合理使用了 sFlow 协议对于流量信息的监控能力和 Openflow 流表对于数据转发的有效支持, 构建出了一套新型高效的云平台虚拟机监控系统.

1 Neutron 网络模块与 SDN 以及 LibvirtAPI 关键技术介绍

1.1 Neutron[2]模块概述

Openstack 项目中包含有计算 (Nova)、存储 (Swift)、网络 (Neutron)、身份服务 (Keyston) 等几个核心子项目. 其中 Neutron 是整个云架构的网络组件, 在 Openstack 发展的初期, 虚拟网络的创建和管理是由 Nova 项目来实现的, 叫做 Nova-network. 其可以提供简单的网络服务和基于 L2 的网络服务. 但随着云计算中对网络更为复杂和高级的要求, 社区中孵化了一个单独的网络项目, 称为 Quantum, 后来由于版权的问题, 更名为 Neutron. Neutron 本身架构由三个核心部件构成, Neutron Server 组件是最核心的一个组件, 其中含有守护进程 neutron-server. Neutron 整体框架组成上可以简单的定义为 Neutron Server+API+Plugin, 即提供给外部调用的功能接口和对内部扩充时运用的插件. 最后结合位于两者中间的 Nertron Server 就够成了我们的 Neutron 组件.

Neutron 中的各个组件在实际部署中通常根据各自的功能实现的不同, 分别布置在 Openstack 框架中的三个节点中, 分别是用于部署 Neutron Server 的控制节点、部署负责转发服务的 L3-agent 和提供 DHCP 服务的 DHCP-agent 等插件的网络节点、部署负责具体实现的 Plugin-agent(插件代理) 的计算节点. 在 Neutron 部署的网络拓扑图中还有三个关键的网络需要我们认识, 分别是负责 Openstack 中各个模块之间交互和连接数据库的管理网络 (Management Network)、负责虚拟机之间数据交互的数据网络 (Data Network)、最后是虚拟机连接外部或则外部调用 Openstack 的 API 都必须通过的外部网络 (External Network).

1.2 SDN 技术概述

SDN[3]技术即软件定义网络技术, 最初起源于 2006 年斯坦福大学的 Clean Slatey 研究课题, 并于 2009 年由 Mckeown 教授提出了核心的 SDN 概念. 不同于传统控制逻辑耦合在相应硬件模块上的形式, SDN 技术采用控制逻辑和实际工作硬件组相互分开的形式. 作为一种新型的网络创新架构, 是对传统分布式网络架构的一种应时更进. 由于现今网络更新和创建的速度已经达到了新量级, 快速的网络业务变更直接要求系统更频繁的更新网络设备的配置 (路由器、交换机、防火墙等). 但在传统分布式网络中, 网络设备不仅承担着网络数据处理还要承担相应逻辑控制, 所

以一旦在业务发生变更后再进行配置就将变得相当庞杂. SDN 将网络设备的控制逻辑从普通网络设备中剥离出来, 将其集中化, 再形成新的集中控制平台, 使设备只需要负责单纯的数据处理. 同时 SDN 将向上提供开放的 API 接口给用户, 既北向接口. 向下开发出接口连接普通网络设备层, 既南向接口. 同时集中数据转发逻辑与 SDN 控制器, 从而形成高效且不依赖转发设备的现代化网络架构.

1.3 LibvirtAPI 关键技术

LibvirtAPI[4]指的是 Libvirt 虚拟化库, 该虚拟化库中是一套面向虚拟机的开源 API. LibvirtAPI 主要应用在基于 Linux 系统的虚拟机管理上, 现在 LibvirtAPI 已经能够向 KVM、XEN、QEMU、VMWARE 等诸多主流虚拟机提供一套完整通用的 API 编程接口, 通过该接口可以忽略不同 Hypervisor 的差异实现高效的虚拟机管理. LibvirtAPI 所有 API 均采用 C 语言进行开发, 可以有效地切合 Linux 系统. LibvirtAPI 可以根据其功能分为五个 API 部分: 虚拟机监管程序连接 API、域 API、网络 API、存储卷 API 和存储池 API. 其中虚拟机监管程序连接 API 是前缀为 virConnect 的一套 API, virConnect 是使用 LibvirtAPI 其它 API 的基础, 既需要首先通过 virConnect 和 Hypervisor 建立连接,才能调用其它 API 进入虚拟机监控信息获取的使用流程.

LibvirtAPI 直接部署在物理机的 Linux 操作系统上, 同时, 对于物理机上不同的虚拟机, LibvirtAPI 为用户屏蔽了底层 Hypervisor 的差异, 通过对其提供统一的 virConnect 接口, 建立起与虚拟机群之间的连接. 其中, Hypervisor 负责统计和管理对应虚拟机群和其占有的所有资源. 与其建立连接后, 通过调用 LibvirtAPI 相应 API 接口便可以获取来自 Hypervisor 对虚拟机群所有有价值的资源统计信息. 利用 LibvirtAPI 技术不仅解除了传统方法中将虚拟机当作物理机, 在每个被监控虚拟服务节点部署监控代理时虚拟机承受的额外压力. 同时, 也使得用户和运维人员可以在单一物理机上实现对其上多台和多样的虚拟服务器的监控管理, 有效地提高了相应的工作效率.

2 Openstack 虚拟化流量监控系统主要功能模块设计与分析

2.1 系统整体架构设计思想及主要模块流程分析

本次设计的主题思想是利用有效的虚拟流量监控和建立基于 Openflow 流表的数据转发机制, 有效地解决开源项目 Openstack 云平台上存在的网络瓶颈问题. 系统将利用 SDN 技术中的 Openflow 协议与 Openstack 中的网络模块 Neutron 进行集成, 构建成系统的流量转发控制模块. 同时, 使用 Openflow 流表取代网络节点成为唯一的流量转发标准, 构建基于该流表的 Openflow 虚拟交换机模块进行流量转发和虚拟机选择. 在计算节点上布置 librvirt 和 sFlow 进行虚拟机普通数据和网络数据的采集, 同时构建监控模块获取其对虚拟机实时性能的采集信息. 获得的数据一方面直接通过用户界面反馈给管理员, 另一方面会将数据放入内存数据库 Redis 随时供控制模块调用数据, 控制模块根据调用的实时数据以及动态负载均衡算法制定着合适的转发流表, 从而使整个系统能够趋向于负载均衡.

系统整体采用分布式架构进行设计, 搭建在 Openstack 云平台之中. 自顶向下的设计中依次包含五个重要的功能部分, 第一个模块是 web 用户模块, 该模块是基于 Openstack 的 Dashboard 模块设计的, 主要用于与用户交互. 第二个部分是控制模块, 其核心部件是 Neutron 中的守护进程 Neutron-server 和 Openflow 控制器, 控制模块主要负责向上提供 API 接收用户的请求, 向下关联着流表, 接收虚拟交换机模块的反馈消息进行处理. 第三个功能模块是虚拟交换机模块, 其负责提取来自虚拟机的数据包并且维护流表, 比较数据包是否能匹配到对应的流表项, 若没有匹配到则向控制模块进行反馈. 第四个功能部分是虚拟服务器模块, 在此所有的虚拟服务器都是由 Nova 进行创建, Nova 是 Openstack 中的计算组件, 其负责虚拟机的创建和资源的调度等. 第五个功能模块是监控模块, 其负责通过 Livirt API 和 sFlow[5]进行虚拟机群的信息监控收集, 获得的数据通过 josn 的格式存储在 Redis 数据库中. 控制模块和用户可以从数据库中获得必要的信息分别进行负载均衡和及时向用户反馈.

图 1 是整个系统的架构图, 从图中可以看到各个模块之间的基本联系. 整个系统是 SDN 技术与 Openstack 的一个集成, 分别将 Openflow 控制器集成到 Openstack 的网络组件 Neutron 中形成控制模块, 控制模块向上提供给客户 API, 相当于北向集成. 控制模块根据来自监控模块的数据进行流表的制定, 以期达到负载均衡. 同时, 虚拟交换机模块桥接着计算组件 Nova 中的虚拟机群, 虚拟机群的所有交互流量都需要

通过虚拟交换机模块的处理, 虚拟交换机模块同时也维护着流表, 如果一段数据包并没有匹配到流表项, 则虚拟交换机模块将会反馈到控制模块, 控制模块会进行处理后制定新的流表完成数据包的转发. 监控模块为了获取虚拟机群的实时性能信息将采用 Livirt API 和 sFlow 协议, Libvirt API 负责获取物理信息, sFlow 负责获取虚拟机群网络运行状态. 数据会存储在 Redis 数据库进行保存, Redis[6]数据库是一种基于内存的数据库, 存储效率优秀.

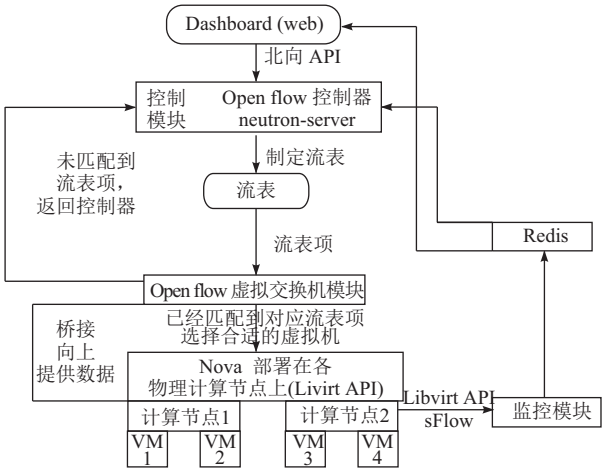

图 1 系统整体架构图

2.2 控制模块分析

控制模块是基于 Openstack 的网络组件 Neutron 和 Openflow 协议而集成的[7], 是系统中最为核心的功能模块. Neutron 负责整个 Openstack 的组网模式, 将 Openflow 控制器集成于其中而形成的控制模块向上对用户提供 API 接口, 使得用户的数据包能够到达控制模块, 向下连接着虚拟交换机模块以便间接控制流量转发. 同时控制模块负责制定转发流表, 每当需要根据 Openflow 协议制定流表时, 都会从监控模块中提取虚拟服务器负载数据并且使用相关负载均衡算法进行 流表制定. 下层的虚拟交换机模块将根据流表分发用户请求和进行虚拟机群数据流量转发, 以期系统性能达到最佳.

图 2 是控制模块的结构图, 从图中可以看到控制模块的基本工作流程. 其在北向连接上层用户, 南向下发流表给虚拟交换机模块控制数据转发. 控制模块在北向接口接收到来自用户的请求后, 其会首先提取用户数据包中的源 MAC 地址, 同时与所维护流表进行匹配操作, 如果存在有流表项与该数据包匹配, 则说明该条用户请求是一条旧的请求, 则由原虚拟机服务器继续未完成服务. 同时为了能顺利区分新旧用户请求, 控制模块会自行维护一张已提供服务流表项表用于匹配, 以期达到虚拟服务器对外提供服务期间的完整性. 在整个服务期间, 由于虚拟交换机模块工作于二层协议, 虚拟机的 IP 是不会暴露给用户的, 所以整个后台服务的处理程序对于客户来说都是透明的, 用户并不会知道为其提供服务的是哪一台虚拟机. 如果虚拟服务器没有该源地址 MAC 的记录信息, 则说明该条用户请求是新的, 控制模块将根据来自 Redis 数据库的监控信息和负载均衡算法[8]为用户选择合适的虚拟服务器.

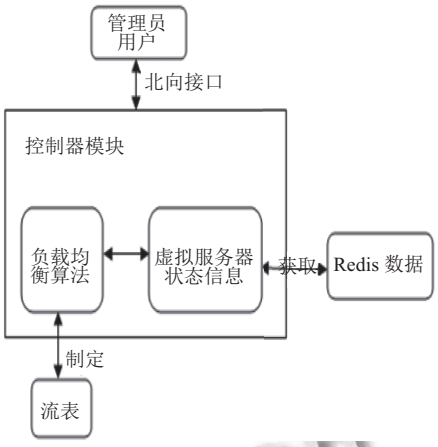

图 2 控制模块结构图

控制模块在南向与虚拟交换机模块相联系, 虚拟交换机模块直接与计算组件 Nova 构建的计算节点群相桥接. Nova 创建的虚拟机分布于各个计算节点上, 虚拟机相互之间产生的数据流量包和与外界交互的数据流量包都直接通过虚拟交换机模块进行转发, 而不用再去创立单一的网络节点. 每当虚拟交换机模块接收到来自底层虚拟机群的数据包的时候, 首先会提取数据包, 并且尝试匹配到相应的流表项, 如果匹配到相应的流表项, 则根据流表项中的目的地址进行下一步的数据转发. 如果没有能匹配到相应的流表项, 则将该来自虚拟机的数据包发往控制模块处理. 控制模块将提取数据包中相关信息, 并且为其定制相应流表并且下发新建流表到虚拟交换机模块进行再一次的数据包转发.

2.3 虚拟交换机模块分析

Openvswitch[9]是一款十分优秀的开源虚拟交换机,

Openswitch 虚拟交换机已经完全实现了传统物理交换机的功能, 并且 Openswitch 已经提供了对 sFlow 和 Openflow 协议的支持. 这些特性使得该虚拟交换机能够在 Openflow 的支持下作为 Openflow 交换机进行基于流表转发数据, 且与控制逻辑层解耦的新一代交换机, 相对于此, 将转发逻辑融合在内的传统物理交换机就显得沉重缓慢. 同时, Openswitch 主要是通过 C 语言实现, 因此对于在大多数平台上进行部署都会较好移植.

基于 Openflow 和 Openswitch 实现的虚拟交换机模块兼具着良好的控制性和扩展性, 具体的数据转发控制策略都是由控制模块进行制定, 因此也就实现了数据转发和逻辑控制的分离. 这种低耦合也是 SDN 技术的核心思想, 系统中各模块在这种低耦合的体系中, 只需要专注处理好自身的任务便可, 从而能成倍的提高运行效率.

图 3 是虚拟交换机模块在系统中的工作流程图, 从图中可以了解到虚拟交换机模块处在控制模块与虚拟服务器之间, 负责的任务就是基于流表的数据转发. 在系统中, 虚拟交换机模块只是负责实现普通虚拟交换机的数据转发功能, 对于数据包如何进行转发则毫不知情, 在整个 Openflow 虚拟网络中, 该组件只是一个执行者. 其只需要根据控制模块已经规划好并且下发流表的逻辑进行机械的操作便可. 也正是由于这种特点, 使得虚拟交换机模块具有了很好的扩展性, 如果需要更新或则改变扩展功能时, 我们只需要去改变控制模块端的相应规则便可.

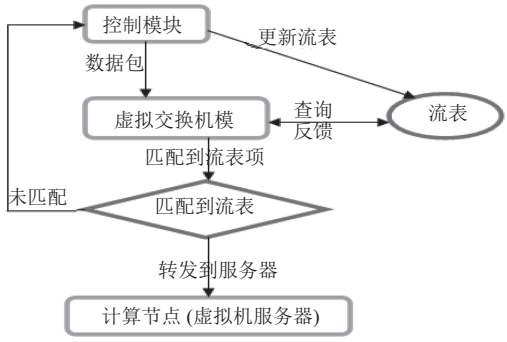

图 3 虚拟交换机模块工作流程图

虚拟交换机模块根据控制模块中的 Openflow 控制器制定的转发逻辑完成虚拟机数据流量的转发和用户请求的重定向. 在整个平台中, 虚拟交换机模块基于 Openflow 协议和控制模块交互, 虚拟交换机负责的基本流程为: 接收来自控制模块的用户请求并根据流表项进行虚拟机服务器选择、同时接收来自底层虚拟机群的数据包并且匹配到相应流表项、如果没有匹配到则反馈到控制模块建立新的流表. 虚拟交换机所维护的流表大多都是通过这种方式建立的.

2.4 监控机模块分析

监控模块是整个系统中联系底层计算节点和控制模块的逻辑中轴, 监控模块将密切关注处于计算节点上的各虚拟机的实时状态并且及时向控制模块和用户管理员反映虚拟机群的状态, 方便控制模块能够获得及时的数据以便在制定流表项的时候使整个系统达到负载均衡, 也使管理员用户能够及时感知整个系统的运行状态.

现今对虚拟机的监控最为常见的方式便是在虚拟机中部署 Agent 进行虚拟机状态信息的获取, 这种传统的监控方式是从对物理机的监控上移植过来的. 这种通过代理获取服务器状态信息的监控方式主要是基于 SNMP 协议进行实现的, 当下使用十分广泛的 nagios 监控系统即是一个基于 SNMP 协议的系统, nagios[10] 作为一款优秀的开源电脑系统和网络监视工具, 能够十分高效的完成对 Windows、Linux、Unix 等系统的主机状态和交换机路由器等网络配置的监控. 但是, 正如其它基于 SNMP 协议的监控一样, 负责管理的平台必须要维护一个服务器程序 nagios-server, 同时对于不同的被监控系统需要其维护不同的客户端程序, 如 Windows 系统下需要安装 nsclient++客户端程序, 而 Linux 系统则需要安装 nagios 的 nrpe 插件, 这无疑将是虚拟服务器的一项巨大负担.

因此, 本系统的在对监控模块进行设计时, 摒弃了原本在虚拟服务器上部署 Agent 的做法, 转而将采用在物理宿主机上直接部署 LibvirtAPI 的方式来设计监控模块. 系统中, 监控模块使用 LibvirtAPI 和 sFlow

协议去获取虚拟机的网络流量信息和虚拟机运行的基本信息. 不同于基于 SNMP 协议的监控程序, Libvirt 提供了一种虚拟机监控程序察觉不到的 API 接口, 其直接部署在物理机上, 通过 virConnect 接口与虚拟机管理器建立连接, 安全的运行于宿主机之上对虚拟机进行稳定的监控. 建立连接后可以通过 virNetwork 接口对虚拟机网络相关信息进行管理、通过 virDomain 接口可以获取虚拟机 CPU 使用的相关信息、通过 virStorageVol 接口对存储情况进行监控. 同时 sFlow 协议在 LbvirtAPI 的基础上获取网络相关信息, 其后传

递信息至管理员和数据库中. LibvirAPI 直接连接 Hypervisor 对虚拟机相关信息进行获取, 相较于通过分散在各虚拟机内部的代理获取的方式也将更具高可用性.

图 4 是监控系统相关模块结构图, 从图中可以看出整个监控系统中以监控模块为核心, 向下通过 LibvirtAPI 和 sFlow 协议获取虚拟服务器的运行状态和网络状态信息, 向上则可以直接反馈信息给管理员用户以便其可以内视系统的整体状况. 同时, 由于监控信息的实时性, 数据不必要进行持久化, 因此选择 Redis 数据库进行数据存储. 控制模块通过 Redis 数据库便可获取相关监控信息, 并利用监控实时信息对流表进行维护和制定, 最终使系统整体高效运行.

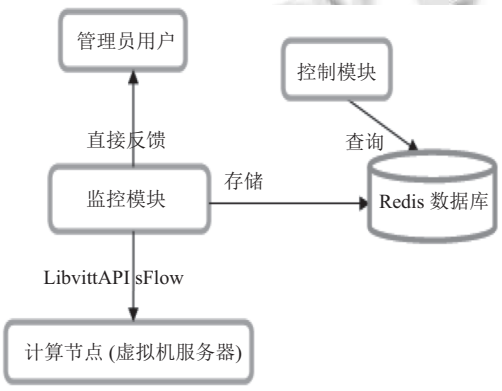

图 4 监控系统模块结构图

3 Openstack 虚拟化流量监控系统测试

系统设计是基于搭建的 Openstack 云计算平台之上的, 整体采用分布式架构进行部署, 在物理结构上云计算平台由计算节点、控制节点、网络节点构成. 控制节点是整个系统的核心部分, 系统的控制模块将与网络组件 Neutron 一起布置在控制节点, 其将负责联系着位于计算节点的虚拟机群和提供给用户的 web 界面. 为了便于搭建实现, 系统采用 Fuel Openstack 工具进行一键式部署, 经过全局考虑, 系统将部署两个计算节点和一个控制节点. 前期将准备的硬件配置有如表 1 所示.

表 1 测试节点配置情况表

	操作系统	内存(GB)	CPU(GHz)	硬盘(GB)
计算节点1	Linux(Ubuntu)	8	2.4	500
计算节点2	Linux(Ubuntu)	8	2.4	500
控制节点	Windows7	4	1.6	500

在基础系统搭建完毕后, 系统将分别在两个计算节点上各创建了三台虚拟机构成虚拟机群以便进行实时的监控. 其后在一段时间内对虚拟机群内的六个虚拟机进行网络流量的检测, 分别在控制节点由管理员用户下发进行游览网页、在线播放歌曲等与网络活动相关的任务指令到控制模块 (这些网络任务便于观察中断和延迟), 以便通过监测的数据流量情况判断整体负载 (当然也可以通过 CPU 等其它指标), 同时也将记录服务期间是否有中断和服务延迟. 控制模块根据监控数据和相关负载均衡算法制定恰当的流表, 并且使虚拟交换机根据流表转发数据流到计算节点的相应虚拟机服务器, 后者在完成任务期间将会产生网络流量, 同时进行监控数据获取和相关数据的分析. 测试时间间隔为一小时, 总共进行五次测试, 测试数据为各节点虚拟机的网络流量, 如表 2 所示.

表 2 各节点网络测试数据结果表 (M)

	VM1	VM2	VM3	VM4	VM5	VM6
8:00-9:00	52	50	47	49	50	48
9:00-10:00	60	62	61	59	60	59
10:00-11:00	55	53	51	52	54	52
11:00-12:00	51	50	53	51	50	52
12:00-13:00	62	63	60	61	61	63

系统测试使用从上层客户应用下达的网络命令, 该命令以数据包的形式交由控制模块进行处理, 控制模块利用负载均衡算法和从 Redis 数据库中获取的实时虚拟服务器状态信息制定适当流表, 该客户数据包根据流表的信息进行转发, 并且最终到达或则交给指定的最合适的虚拟机服务器.

对测试数据分析可得出, 系统主要实现了以下三个方面的效果.

1) 通过系统能够成功的获取计算节点中各虚拟服务器的阶段性时间内的数据流量信息;

2) 系统在进行网络活动时, 未出现网络服务中断或延迟的情况, 且整体运行良好;

3) 计算节点中各虚拟服务器在进行网络服务时, 未曾出现单机负载过大或则单机闲置的情况, QoS 质量良好且系统整体达到负载均衡.

通过测试结果可以看出系统能够有效的获取阶段性的虚拟服务器流量信息, 同时在实现对虚拟服务器的监控基础上, 系统在进行网络服务时未出现服务中断和时延且各虚拟服务器在多任务下发时负载情况基本接近, 使系统整体趋向负载均衡, 使系统 QoS 质量得

到了保证.

4 虚拟流量监控系统问题分析

该虚拟流量监控系统在一定程度上达到了虚拟服务器数据流量的监管和负载均衡控制, 但是由于仍需要在计算节点部署相关数据获取接口, 所以仍将消耗一部分计算资源. 同时, 测试环境较为简单, 当面对企业级的大型云平台系统时, 为了能够达到预期的效果, 其实时性和高效性仍然需要进一步的改进. 系统构建于 Openstack 云平台上, 虽然该平台在业内被普遍使用, 但是其它不同平台亦占有一定份额, 是否能够在其他平台上得到更加高效的结果, 将需要在移植到其它平台后做再一步的测试.

参考文献

1 陈康, 郑纬民. 云计算: 系统实例与研究现状. 软件学报, 2009, 20(5): 1337–1348. [doi: 10.3724/SP.J.1001.2009.03493]

2 李莉, 李纪成, 张超然, 等. 基于 OpenStack 云平台 Neutron 关键技术研究. 长春理工大学学报 (自然科学版), 2015, 38(6): 114–117.

3 左青云, 陈鸣, 赵广松, 等. 基于 OpenFlow 的 SDN 技术研究. 软件学报, 2013, 24(5): 1078–1097.

4 姚华超, 王振宇. 基于 KVM-QEMU 与 Libvirt 的虚拟化资源池构建. 计算机与现代化, 2013, (7): 26–29, 33.

5 范亚国. 基于 sFlow 的网络链路流量采集与分析[硕士学位论文]. 武汉: 武汉理工大学, 2008.

6 邱祝文. 基于 redis 的分布式缓存系统架构研究. 网络安全技术与用, 2014, (10): 52, 54.

7 Malik A, Ahmed J, Qadir J, *et al*. A measurement study of open source SDN layers in openStack under network perturbation. Computer Communications, 2017, 12: 139–149.

8 田浪军, 陈卫卫, 陈卫东, 等. 云存储系统中动态负载均衡算法研究. 计算机工程, 2013, 39(10): 19–23. [doi: 10.3969/j.issn.1000-3428.2013.10.005]

9 张若晨. 基于 OpenvSwitch 的代理虚拟交换机在 SDN 网络中的实现与应用[硕士学位论文]. 广州: 华南理工大学, 2016.

10 和荣, 肖海力. 基于 Nagios 的监控平台的设计与实现. 科研信息化技术与应用, 2014, 5(5): 77–85. [doi: 10.11871/j.issn.1674-9480.2014.05.011]