计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(8):71−76 [doi: 10.15888/j.cnki.csa.005875]

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

# 基于私有云平台的云主机资源监控方案①

张　倩, 何汉东

(中电海康集团有限公司, 杭州 310013)

**摘　要**: 针对传统网格计算中的监控工具已无法满足云计算平台中虚拟资源监控的问题, 本文基于 OpenStack 云平台设计并实现一套完整的私有云平台中云主机资源监控方案. 实验表明该方案可以有效地进行虚拟资源的监控, 满足企业私有云中对虚拟资源的监控需求, 且系统方案具有良好的可扩展性.

**关键词**: 私有云; OpenStack; 云主机监控

引用格式: 张倩,何汉东.基于私有云平台的云主机资源监控方案.计算机系统应用,2017,26(8):71–76. http://www.c-s-a.org.cn/1003-3254/5875.html

## Cloud Host Resource Monitoring Scheme Based on Private Cloud Platform

ZHANG Qian, HE Han-Dong

(CETHIK GROUP Co. Ltd., Hangzhou 310013, China)

**Abstract**: The traditional monitoring tools in grid comping have been unable to meet the virtual resource monitoring requirements in cloud computing platform. To solve this problem, this paper designs and implements a complete scheme of cloud host resource monitoring in private cloud platform based on OpenStack. Experimental results show that the proposed scheme can effectively monitor the virtual resources, and meet the requirements of the enterprise private cloud monitoring of virtual resources, and the system has good scalability.

**Key words**: private clouds; OpenStack; cloud host monitor

当前互联网技术发展迅猛, 云计算已然成为最热门的研究领域之一, 各种各样的作为 IaaS 层服务的云计算平台也相继涌现, 几种主要的开源 IaaS 软件有 Eucalyptus[1]、OpenStack[2]、CloudStack[3]和 OpenNebula[4]等. 在私有云平台[5]中, 虚拟资源的数据监控和展示是云平台中重要的部分, 随着节点数目和所需监控对象数量的增加, 一个强大、健全、可扩展的监控系统才能满足各类用户的需求. 它是一个云平台非常重要的特性, 也是评估一个 IaaS 的可运维程度的参考. 因此, 在私有云应用中云主机监控系统是必不可少的.

虚拟化技术作为云计算平台的基础, 云主机的性能及稳定性直接影响了整个云平台的性能以及用户的体验. 虽然虚拟化技术不断成熟, 然而在云计算平台中基于云主机资源监控的监控技术还在不断发展, 并没有通用的监控系统可以使用. 目前比较成熟的监控技术大多数是针对分布式网格集群设计的, 比如 Ganglia, Nagios[6], Zabbix[7], Zenoss 等, 通常这些监控系统需要在被监控服务器中部署监控代理, 监控服务器负责与监控代理进行通信, 实现监控需求. 这些成熟的监控系统更加适合于物理节点资源的监控, 并不适合在私有云平台中用于监控云主机资源. 主要原因包括以下几点:

首先, 在私有云平台中, 网络层通常需要实现安全隔离, 满足用户私有网络与系统管理网络相互隔离, 无法直接进行通信. 该情形下, 用户云主机在用户私有网络中, 监控服务器在系统管理网络中, 那么适用于物理机的监控工具, 无法实现云主机中监控代理与监控服务器的网络通信.

① 收稿时间: 2016-11-28; 采用时间: 2016-12-19

其次, 在私有云平台中, 假设已通过某种策略实现用户私有网络与系统管理网络之间的通信, 那么在云主机内部部署复杂的监控代理, 这将大大增加云主机的负载, 增加网络开销, 同时增加物理服务器的开销.

所以, 适用于物理服务器的监控方案并不适用于私有云环境下的云主机资源监控[8][9][10]. 同时, 如果直接采用物理节点上的云主机管理器 VMM 获取云主机资源信息, 监测到的资源有限, 无法获取准确的信息. 因此, 在私有云平台中急切需要设计一种云主机资源监控方案, 实现资源实时监控, 提供性能历史数据供平台分析使用, 这些数据信息可以用来找到当前系统性能瓶颈(如作为性能分析 performance analysis)和预测系统未来的 load(如容量规划 capacity planning).

针对私有云的应用, 在网络层安全隔离的需求下, 需要且尽可能减少监控软件对整个平台性能的影响, 本文采用宿主机与云主机直接进行通信, 通过在宿主机中收集云主机的资源使用情况. 本文基于业界流行的云计算平台 OpenStack, 设计并实现一种基于私有云平台的云主机资源监控方案, 并进行实验验证.

## 1 相关技术分析

### 1.1 OpenStack 云平台

OpenStack 是开源的 IaaS(基础设施即服务)云计算平台, 它由多个不同功能的子项目构成, 可以提供计算、存储、网络的资源的管理, 为服务商及企业本身提供了云基础架构服务. OpenStack 由一系列相互关联的项目提供云基础设施解决方案的各个组件. 伴随着各大 IT 公司的加入以及各级学者对社区的贡献, OpenStack 的功能在不断完善.

Nova 是 OpenStack 基础架构服务的核心子项目, 它创建一个抽象层, 通过支持不同的虚拟化管理程序, 对 CPU、内存、网络适配器、硬盘驱动器等服务器资源实现虚拟化, 提高系统的资源利用率和资源自动化管理功能, 该项目提供云主机的全生命周期管理, 包括启动、扩容、挂起、停止和重新引导等功能. Nova 组件虽然可以管理云主机的各项操作, 但是并不能提供监测云主机内部资源的使用情况数据.

### 1.2 云主机资源监控

Nova 组件支持多种 hypervisor, 主要包括 Xen, KVM, VMware vSphere, Hyper-V 等. 本文采用业界较为成熟开源的 Libvirt+KVM 作为 Nova 的后端虚拟化驱动, 为整个系统提供计算虚拟化功能. 在虚拟化技术中, 内存资源作为重要的资源, 如果可以监测到内存资源的使用情况, 根据实际情况进行动态地调整内存资源, 这将会提高云主机的性能, 从而提高业务效率. 再者, 对于云主机中磁盘的负载监控也是必不可少的, 尤其是在私有云应用场景下, 企业内对存储空间要求较高, 如果可以根据监测到的磁盘信息进行动态地调整存储空间, 将会使得平台更加适合存储要求高的应用场景.

OpenStack 平台中, Nova 组件提供了 RESTful API 通过调用 Libvirt 提供的 API 查询云主机实例信息, 包括云主机的状态信息, 分配时的 CPU、内存、磁盘等信息, 且这些都是静态的信息, 并不能动态获取云主机实际的资源使用情况. 如: vcpu 的信息是存储在 hypervisor 层的, 仅仅利用 virt-manager 无法获取较底层的信息, 在考虑平台负载均衡时可能会带来问题; OpenStack 平台中获取的仅仅是创建云主机时分配的内存大小, 无法获取实时的内存使用情况, 这将会影响用户对资源的直观判断, 也无法确定是否需要增加内存空间. 以上分析可见, 仅通过 OpenStack 平台中的 RESTful API 是无法获取云主机中实际的资源使用情况的.

Ceilometer 是 OpenStack 的一个子项目, 它可以实现监测云主机的功能, 它将监测到的数据主要用于计量使用, 该组件部署较复杂. 而且在私有云平台中, 计量功能相对弱化, 对租户网络隔离性要求较高. 综上分析, 采用现有的监控方案是不适合私有云平台的, 需要设计并实现一套满足私有云平台的云主机监控方案.

### 1.3 Collectd+InfluxDB+Grafana 监控系统框架

目前, 业界有一套比较流行的资源监控系统框架, 即: Collectd[11]结合 InfluxDB[12]与 Grafana. 其中, Collectd 是 C 语言开发的一个守护进程, 用来周期性地收集统计数据和提供各种存储方式来存储不同值, 它主要采用插件的形式来收集不同的资源数据; InfluxDB 是一个开源, 分布式, 时间序列, 事件, 可度量和无外部依赖的数据库, 非常适合存储指标、事件、分析等数据; Grafana 也是一个开源的, 具有丰富指标仪表盘的数据展示和图表编辑工具.

本文利用 Collectd+InfluxDB+Grafana 监控系统框架进行二次开发, 通过在宿主机中部署 Collectd 工具, 开发实现在宿主机中收集云主机资源信息的 Collectd

插件. 该方案主要优势有三点, 一是可以避免在云主机中直接部署 Collectd 工具, 减少监控工具对云主机性能的影响; 二是避免依赖网络通信实现云主机的资源监控功能; 三是云主机的信息采集通过 Collectd 的插件形式实现, 可扩展性强.

## 2 云主机监控系统方案设计

### 2.1 系统整体设计

本文将 OpenStack 云计算平台应用在私有云建设中, 要求可以监控到所有云主机的信息, 用户可以根据实际使用情况动态地进行资源扩展或者系统自动对私有云中的资源进行扩展; 还可以根据云主机占用宿主机资源情况进行云主机的动态迁移. 因此, 云主机的资源监控在私有云平台中是非常重要的部分.

本文提出的云主机监控系统方案可以实时监测用户创建的云主机资源使用情况, 包括云主机 CPU、内存、网络等, 同时将这些监控数据存储在数据库中, 提供数据分析及数据显示. 云主机监控系统总体架构如图 1 所示.

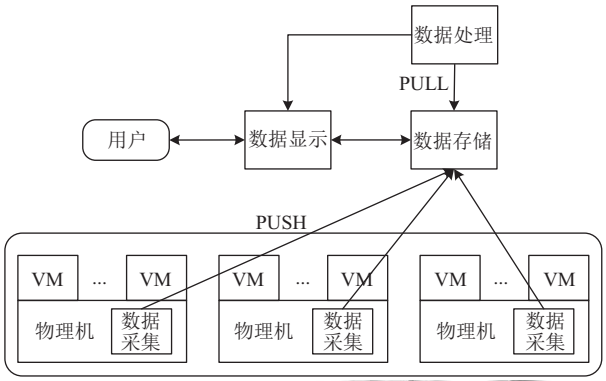


图 1 云主机监控系统架构图

整个监控系统分为四个模块, 分别为数据采集、数据存储、数据处理、数据显示. 数据采集模块部署在每个计算节点上, 负责收集每个计算节点上通过 OpenStack 创建的云主机的资源信息; 数据采集模块将收集的数据推送到数据存储服务器, 数据存储模块负责将数据存储到数据库中; 数据处理模块定时查询数据库历史信息, 进行数据分析判断云主机资源的使用情况, 如是否需要报警; 数据显示模块负责将云主机实时监测的数据显示给用户. 下面重点分析数据采集模块的设计与实现, 这也是本文的重点内容.

### 2.2 系统各模块详细设计

#### 2.2.1 数据采集模块

本文 Nova 采用基于内核的虚拟化技术 KVM 作为后端驱动, 基于 KVM 技术有一种工具 qemu-guest-agent(qga), 其可以实现宿主机与云主机之间的通信, 原理如图 2 所示. 宿主机(compute node)通过读写 socket file, 云主机实例(Instance)通过读写串口设备, 两者实现通信.


图 2 qga 通信原理图

在可以实现宿主机与云主机通信的基础上, 基于 Collectd 进行二次开发云主机资源采集插件(Python plugin). Collectd 工具部署在宿主机中, 插件程序首先筛选出 OpenStack 平台中活跃状态的云主机, 然后由 Libvirt[13]调用 qemuAgentCommand()函数获取云主机中资源文件, 分析得到资源使用情况信息.

数据采集模块满足私有云环境下网路安全隔离, 在租户网络与管理网络不进行网络通信的情况下, 可以监测到云主机内部资源的使用情况, 同时尽可能减少在云主机中部署重量级的监控代理服务, 减少监控功能对云主机资源的占用. 数据采集程序采取主动收集信息的方式将数据 PUSH 到数据存储层, 实时保存数据到数据库中提供数据处理和数据显示.

#### 2.2.2 数据存储模块

数据存储模块部署在数据库服务器中, 负责将数据采集模块发送的数据存储到数据库中, 本文采用 InfluxDB 时序数据库. 采用 Collectd 工具 Python 插件(collectd-python)的数据格式 PluginData, 云主机监控数据在 InfluxDB 中数据库表(openstack_vm_value)结构如表 1 所示.

表 1 云主机资源信息表

| time | host | instance | type | type_instance | value |
|---|---|---|---|---|---|
| 时间 | 云主机所在计算节点主机名 | 云主机所在租户 | 值类型(gauge) | 云主机属性 | 云主机属性值 |

#### 2.2.2 数据显示模块

数据显示模块采用开源工具 Grafana, 定时获取

InfluxDB 的数据自定义排版显示某段时间内的资源信息, 为用户提供直观的界面.

## 3 云主机监控系统方案实现

### 3.1 监控系统软件架构

整个云主机监控系统软件架构图如图 3 所示, 云主机中部署 qga 服务作为 socket 的服务端, 宿主机中封装一个 socket 的客户端; 采用 Python 编程实现 Collectd 的云主机监控数据采集插件, 插件作为 socket 客户端发送请求读取云主机当前资源信息, 云主机中 qga 服务负责获取资源文件信息返回. 每个计算节点会有一个 collecd的收集器, 将收集到的云主机资源信息发送给 InfluxDB, InfluxDB 提供了 Collectd 的插件负责将信息写到数据库中; 用户通过 Grafana 自定义获取 InfluxDB中的资源信息进行显示.

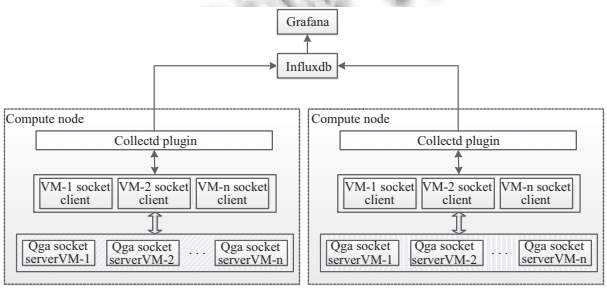


图 3 云主机监控系统软件架构图

### 3.1 监控系统数据采集模块实现

数据收集程序作为 Collectd 的插件部署在每个计算节点上, 通过在 Collectd 中配置定时时间来实现每隔一段时间获取云主机实例的资源使用信息, 从而满足资源监测.

开发语言采用 python2.7, 首先在云主机中需要部署好 qga 程序, 可以实现宿主机与云主机之间的通信, 然后通过调用 Libvirt API 查询 OpenStack 平台中活跃状态的云主机, 针对活跃状态的云主机在宿主机由 Libvirt 调用 qemuAgentCommand()函数获取云主机中资源文件, 分析得到资源使用情况信息. 本文主要监控资源包括 cpu 使用率, 内存使用率及网络流量.

① cpu 使用率

直接通过读取云主机内部文件/proc/stat 无法获取 cpu 的使用率, 本文采用间接方式采样两个足够短的时间间隔 cpu 快照, 时刻分别记为 $i$-1 和 $i$, 两个采样点分别读取文件/proc/stat 中总 cpu 记录, 根据以下公式计算 cpu 使用率:

$$cpu\_usage_i = (1-(idle_i - idle_{i-1}) \\ (total\_cpu_i - total\_cpu_{i-1})) * 100 \quad (1)$$

其中: $idle_i$-$idle_{i-1}$ 表示 cpu 空闲时间, $total\_cpu_i$-$total\_cpu_{i-1}$ 表示总的 cpu 时间片

② 内存使用率

根据云主机内部文件/proc/meminfo 获取云主机内存使用信息, 计算公式如下:

$$mem\_usage = (MemTotal - MemAvailable) / \\ MemTotal * 100 \quad (2)$$

其中 MemTotal 为总内存, MemAvailable 为已使用内存.

```
输入: OpenStack平台接口api
输出: 某计算节点(compute-x)所有云主机资源使用信息vm_stat_data
//调用nova接口获取节点compute-x上创建的云主机实例列表
openstack_instances
openstack_instances = get instance from nova where host=compute-x
//调用libvirt接口获取节点compute-x上创建的虚拟机实例列表
all_instances
all_instances = get instance form libvirt
//获取平台需要监控的云主机实例
for instance in all_instances
  if instance in openstack_instances
    if instance.status == “active”
      monitor_instances.add(instance)
    end if
  end if
end for
//调用qga command获取资源信息
for instance in monitor_instances
  cpu_usage = get_cpu_usage(instance)
  mem_usage = get_mem_usage(instance)
  network_flow = get_network_flow(instance)
  vm_stat_data.add(instance, cpu_usage, mem_usage, network_flow)
end for
return vm_stat_data
```

③ 网络流量

根据云主机内部文件/proc/net/dev 得到所有网卡收发数据包的信息, 如: Receive_Bytes, Transfer_Bytes.

数据采集插件代码流程如下.

将云主机数据采集插件 vm_plugin.py 放置在目录/usr/lib64/collectd/plugins 下, 云主机监控配置文件/etc/collectd.d/vm/conf 如下:

<LoadPlugin python>

```
    Globals true
</LoadPlugin>
<Plugin python>
    ModulePath "/usr/lib64/collectd/plugins/vm"
    LogTraces true
    Import "vm_plugin"
<Module vm_plugin>
    AuthApiServer "http://keystone_api:5000"
    NovaApiServer "http://nova_api:8774"
    AdminTenantName "admin"
    AdminUserName "admin"
    AdminPassword "xxx"
    Verbose "False"
    Debug "False"
</Module>
</Plugin>
```

## 4 测试结果与分析

### 4.1 实验设置

本文采用 4 个物理机节点进行实验,1 个作为 OpenStack 控制节点, 3 个作为 OpenStack 计算节点, 节点硬件及软件配置如表 2 所示.

表 2 物理机配置表

| | |
|---|---|
| 物理机配置 | Intel(R) Xeon(R) CPU E5-2643 v2@3.5GHz x 12<br>系磁盘1 TB<br>内存128 G |
| 操作系统 | Centos7.2 |
| Openstack版本 | Openstack L版本 |
| 第三方软件 | mariadb、rabbitmq<br>Collectd 5.5.1<br>InfluxDB 0.13.0<br>Grafana 3.0.4 |

为了验证本文提出监控系统方案的有效性, 利用上述物理机配置, 搭建 OpenStack 云平台同时在三个计算节点上部署 Collectd 程序(包括云主机数据采集插件), 在控制节点上部署 InfluDB 与 Grafana 进行实验. 分别在 OpenStack 平台上创建 30 个云主机实例, 观测资源使用情况, 云主机配置如表 3, 每种配置分别创建 10 个云主机.

### 4.2 实验步骤及结果

① 制作 OpenStack 镜像文件: 系统采用 centos7.2, 根据 OpenStack 官方镜像制作步骤完成满足 OpenStack 平台的镜像制作功能, 同时在镜像中安装配置 qemu-guest-agent(注意 BLACK_LIST 配置).

表 3 云主机配置表

| | 配置1 | 配置2 | 配置3 |
|---|---|---|---|
| 操作系统 | Centos7.2 | Centos7.2 | Centos7.2 |
| vcpu | 1 | 1 | 2 |
| 内存 | 1 G | 2 G | 4 G |
| 硬盘 | 20 G | 20 G | 40 G |

② 创建镜像: 属性"hw_qemu_guest_agent=yes"需要增加, 满足创建云主机时在宿主机建立 socket 文件, 在云主机中增加串口设备, 实现宿主机与云主机的 socket 通信.

③ 创建云主机: 指定上述镜像创建云主机, 分别创建 30 个云主机实例.

④ 修改 Collectd 配置文件, 启动 Collectd 以及 InfluxDB 服务进程.

⑤ 登录 Grafana 面板, 自定义取得资源信息, 观察云主机资源变化情况.

实验一: 测试云主机 cpu 使用率, 在云主机中分别选择运行不同的进程, 采用 grafana 定义显示面板, 选取某段时间的数据, 每种配置选择 1 个云主机实例显示结果如图 4 所示.

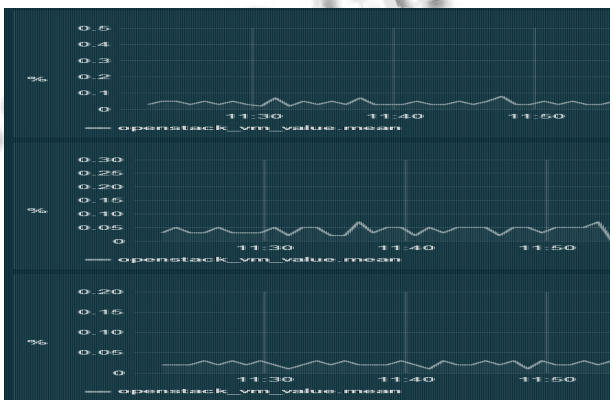


图 4 云主机 cpu 使用率

实验二: 测试内存使用率, 在云主机中分别运行同一程序, 程序持续运行一小时保持占用 100 M 内存, 采用 Grafana 定义显示面板, 选取某段时间的数据, 每种配置选择 1 个云主机实例显示结果如图 5 所示.

实验三: 测试网络流量, 在云主机中分别进行租户网络内数据访问操作, 统计一段时间内网络收发包情

况, 每种配置选择 1 个云主机实例显示结果如图 6 所示.


图 5 云主机内存使用率

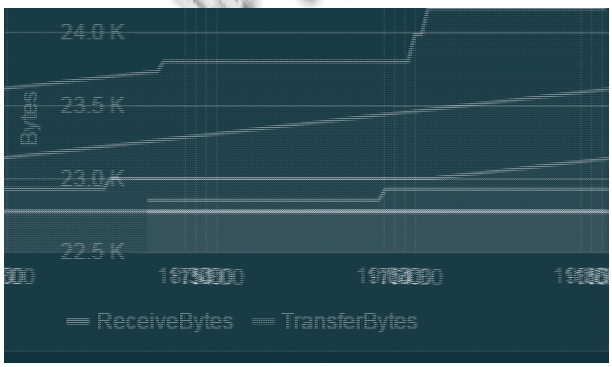


图 6 云主机网络流量

实验结果表明, 本文提出的云主机监控系统可以满足 OpenStack 平台中对云主机实例进行动态监控的需求, 可以根据监控信息判断一段时间内云主机的运行情况, 从而便于用户根据资源使用情况进行相应资源的扩展操作. 同时, 满足私有云环境下网络层安全隔离的要求. 虽然监控系统可以有效地解决云主机资源实时获取, 但该系统在云主机内部启用了 qga 服务, 即便该服务的资源消耗较少, 但是一定程度上也占用了云主机的资源.

## 4 结语

云主机监控系统负责对云主机资源进行数据采集、数据分析和数据显示等操作, 为运维人员提供了系统性能调优的依据, 可以进一步分析系统负载情况; 同时用户可以实时了解拥有的云主机的运行状况, 决定是否进行资源扩容. 针对目前 OpenStack 平台中云主机监控系统业界仍然没有统一的方案的问题, 本文根据实际业务需求, 提出一种基于私有云平台的云主机资源监控方案, 该系统方案可以有效地实现对云主机 cpu、内存及网络资源的监控, 且系统可扩展性较强, 只需要根据监控需求增加 Collectd 的插件即可. 但是, 系统本身也存在资源占用的问题, 后续工作将进一步完善; 在私有云中, 云主机磁盘信息监控同样有需求, 如根据云主机磁盘使用情况进行动态扩容提供业务存储能力, 同时监控系统中数据分析是非常重要的, 这都是下一步需要进行的工作.

## 参考文献

1 Nurmi D, Wolski R, Grzegorczyk C, *et al*. The eucalyptus open-source cloud-computing system. Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. Shanghai, China. 2009. 124–131.
2 OpenStack. 44% more deployments captured in OpenStack's ninth user survey. http://www.openstack.org/.
3 Apache CloudStack™. http://cloudstack.apache.org/.
4 OpenNebula. http://opennebula.org/.
5 Bist M, Wariya M, Agarwal A. Comparing delta, Open stack and Xen Cloud Platforms: A survey on open source IaaS. Proc. of IEEE the 3rd International Conference on Advance Computing Conference. Ghaziabad, India. 2013. 96–100.
6 Nagios. Nagios Open Source. https://www.nagios.org/.
7 Zabbix. http://www.zabbix.com/.
8 Han FF, Peng JJ, Zhang W, *et al*. Virtual resource monitoring in cloud computing. Journal of Shanghai University (English Edition), 2011, 15(5): 381–385. [doi: 10.1007/s11741-011-0755-1]
9 Rossigneux F, Lefevre L, Gelas JP, *et al*. A generic and extensible framework for monitoring energy consumption of OpenStack clouds. Proc. of IEEE the 4th International Conference on Big Data and Cloud Computing. Sydney, SW, Australia. 2014. 696–702.
10 Bermudez I, Traverso S, Munafo M, *et al*. A distributed architecture for the monitoring of clouds and CDNs: Applications to amazon AWS. IEEE Trans. on Network and Service Management, 2014, 11(4): 516–529.
11 Collectd. Collectd-the system statistics collection daemon. https://collectd.org/.
12 Influxdata. The modern engine for metrics and events. https://www.influxdata.com/.
13 Libvirt: the virtualization API. http://libvirt.org/.