



基于 OpenStack 体系的多融合管道式 服务云监控系统的研制^{*}

李 磊¹, 金连文¹, 黎德生¹, 唐万万²

(1. 华南理工大学 广州 510641; 2. 香港云龙科技有限公司 深圳 518000)

摘 要: 为了保证在云计算大集群规模下服务的稳定性和可靠性, 针对目前云监控的需求, 阐述了基于 OpenStack 工程体系结构研制的云监控系统, 并详细论述了系统结构和关键技术点, 包括所采用的工厂模式、Paste Deploy、管道、Eventlet 协线程、面向对象的数据库 SQL Alchemy 等编程模式, 使得系统具备了模块化、层次化、流水线化等结构特点, 具备多模式融合、部署简易、高扩展性和二次开发简易等优势。测试结果表明, 系统具备了功能服务的高稳定性、采集数据的高准确性以及与第三方客户端的易融合性, 能有效地实现云计算平台的信息化监控。

关键词: 云监控; 管道结构; 协线程; 对象化数据编程; OpenStack

doi: 10.3969/j.issn.1000-0801.2014.01.004

Research on Multi-Function Pipeline Integrate Cloud Computing Monitoring System Based on OpenStack Structure

Li Lei¹, Jin Lianwen¹, Li Desheng¹, Tang Wanwan²

(1. South China University of Technology, Guangzhou 510641, China;

2. Hong Kong Cloud Dragon Internet Information Co., Ltd., Shenzhen 518000, China)

Abstract: To deal with the increasing of computer clusters which the increasing needs of cloud computing brings and guarantee stability and reliability of the service, a cloud computing monitoring system based on OpenStack structure to satisfy the needs of cloud computing monitor was designed. The system structure and key technologies were described, including factory model, Paste Deploy, pipeline, Eventlet green thread and object-oriented SQL Alchemy model, so the system has modularity, layering and pipeline. The advantage of the system includes multi-function integrating, easy deploy, high expansibility and redeveloping easily. The experimental result proves that the system has high stability, accurate data acquisition and high integration with the third client, and it can implement cloud computing monitoring informatization more effectively.

Key words: cloud computing monitoring, pipeline structure, eventlet green thread, SQL Alchemy, OpenStack

1 引言

信息化建设的加快、电子商务的流行以及云计算的推

行, 让 IT 服务像水和电一样普及和重要^[1]。集群的构建为云计算的服务提供必要的计算支撑平台, 通过虚拟化的技术将分离在不同主机的计算资源, 包括 CPU、内存、存储等

^{*} 国家科技支撑计划基金资助项目 (No.2013BAH65F01-2013BAH65F04), 广东省战略新兴产业核心攻关基金资助项目 (No.2012A010701001)

进行逻辑抽象,在提供服务时可按照用户的需求进行资源的重构建,实现弹性、按需使用的能力^[2]。目前随着云计算技术的不断成熟,因其具有高效用比、低投入成本的特点,云计算已经逐渐成为互联网企业部署网络服务的首选基础设施,然而面对不断增长的业务支撑需求,如何保证云计算服务的稳定性成为评价其质量好坏的重要依据^[3]。如参考文献[4~6]所述,虽然随着硬件技术的不断完善、软件漏洞的不断修复,云计算集群内物理机出现故障的概率已经逐步地减小,但仍无法保证绝对的服务稳定和安全。因此对云计算平台的运行信息进行实时获取,能提前对故障事件进行判断,做出及时的处理,包括用户告警、数据备份、虚拟主机的迁移,这是保障云计算服务质量的关键基础。

目前监控云计算集群的第三方软件种类繁多,如 Zabbix、Ganglia、Nagios 等,可通过直接安装和配置获取物理机的各项运行信息,但在功能扩展时需要编写插件,包括计费、采集信息种类的扩展等,难以与包括 OpenStack 等的云计算平台进行融合,且获取元数据格式固定,二次开发难度较大;而已进入 OpenStack 的核心项目 Ceilometer 工程虽然属于开源软件,符合 OpenStack 代码体系标准,但代码冗余度大、配置复杂、依赖插件较多,目前尚未具备可用性部署的条件。因此研制适合具备高扩展性、部署简易的紧凑云监控成为目前云监控服务部署的必要基础条件。

按照 OpenStack 代码体系标准,研制具备多模式融合、部署简易、高扩展性的云监控软件,包括提供对集群物理主机、虚拟机运行信息的多模式融合采集、管道式 API 等。

2 OpenStack 工程体系和云监控需求

OpenStack 是目前开源云计算平台架构中发展速度最快的平台之一,采用了多种目前软件开发中的最新技术,包括工厂模式、Paste Deploy、管道、Eventlet^[7]协程、面向对象的数据库 SQL Alchemy^[8]等编程模式,使得工程具备了模块化、层次化、流水线化等结构特点,且所有的 OpenStack 工程都可基于相同基础代码进行构建,保证了与 OpenStack 版本的同步^[9]。

云计算平台监控主要实现两类数据的获取,具体如下。

- 物理主机信息的获取,包括物理信息(如 CPU 个数、内存总数、磁盘容量、网卡个数等)、运行统计信息(如 CPU 使用率、内存使用率、磁盘使用率以及网卡的流量统计等)。

- 虚拟主机信息的获取,包括运行信息(如 CPU 使用率、内存使用率、磁盘使用率以及网卡的传输效率^[10])等。

云监控的本质仅仅是对云计算平台的信息进行采集以及存储,但面对不断增长的业务需求以及集群数量的快速增长,如何保证监控系统的功能高扩展性、高稳定性,成为云监控系统面临的主要挑战。就工程性而言,功能扩展性的保证主要是依靠结构的模块化、流水线化等设计机制实现,使得开发者可以按照需求对监控系统进行二次开发,能快速地将自己设计的模块嵌入现有系统中;而稳定性则需要工程结构简洁,对第三方依赖的模块数较小^[11]。

3 多融合管道式服务云监控结构

综合考虑,设计了一套基于 OpenStack 开源工程体系结构的云监控系统,其结构如图 1 所示。

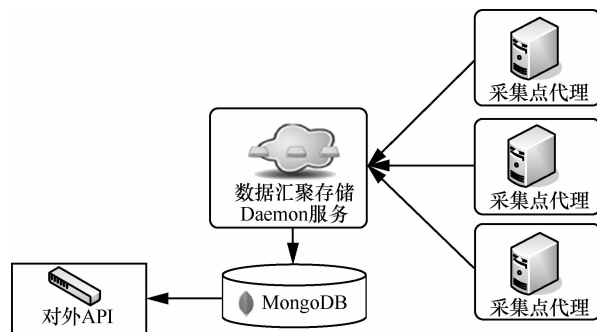


图 1 基于 OpenStack 开源工程体系结构的云监控系统结构

该监控系统主要采用 Python 语言进行开发,主要结构包括以下几个方面。

- 数据库存储支撑。为了提高大容量监控数据存储/读写的效率,采用了 NoSQL 数据库——MongoDB 作为系统的采集信息后台存储。
- 对外 API 的服务。该服务采用 Paste Deploy 结构进行设计,实现管道化信息处理和模块化组合,向用户提供基于 Wsggi 结构的数据接口服务。
- 数据汇聚存储 Daemon 服务。该服务采用面向对象的数据库 SQL Alchemy 设计,实现对各个数据采集节点的数据汇聚以及数据库的信息写入。
- 采集点代理服务。该服务采用协程结构进行设计,将多项数据采集服务进行有序的融合,即提供物理服务器、虚拟云主机各个信息的统计和采集。

在数据的采集服务上,系统按照采集数据的种类、获



取方式分为如表 1 所示的 3 类。

表 1 数据采集种类

数据采集类型	采集内容	获取方式
物理主机运行信息	物理主机各项运行信息	Python 库
虚拟云主机运行信息	CPU、内存、磁盘 I/O	虚拟化管理库 libvirt
虚拟云主机运行统计信息	磁盘使用量、网络传输量	虚拟化管理库 libvirt

由于 OpenStack 的基础工程是基于 Wscgi、Eventlet 等结构,为了保证监控与 OpenStack 进行无缝整合,在工程结构的开发上,系统以 OpenStack 基础工程——Olso 为基础进行开发,该工程是对 OpenStack 常用的一些工程结构和代码框架,依据各个服务的功能结合相对应的模块库进行模块化设计,代码层次结构如图 2 所示。

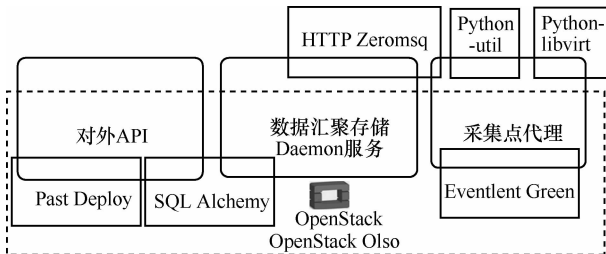


图 2 本系统代码框架层次

4 多融合数据采集代理服务设计与研制

云计算平台的数据采集服务主要负责获取集群内各个物理主机和虚拟主机的相关运行信息,并按照统一的通信协议和数据格式向云监控的数据汇聚端上报采集信息。

考虑到云监控的数据采集信息种类的多样化,在系统中将信息分为 3 类,数据采集服务需要同时满足这 3 类数据采集的任务,同时考虑到采集任务多任务融合的可控性和扩展性,需要采用类似多线程、信号锁等机制。但由于线程运行过程中的管控能力较差,对内存的使用率较低,因此系统设计的数据采集服务以协程为基础,即以 Eventlet 架构为基础。Eventlet 的基础是 greenlet,这是实现

“协程”的基础。协程又称作“微线程”,简言之就是在一个原生线程上通过复制和切换堆栈帧数据实现执行多个工作^[10]。其结构具备以下优点:

- 在具备多线程特点的同时,避免了传统的函数调用栈,使得无限递归成为可能;
- 用户态的线程调度,极大地降低上下文切换的开销,使得近乎无限并发的“微线程”成为可能;
- 由于可以在用户态进行手工线程调度,这样可以避免锁机制,提高了服务的管控能力。

数据采集服务由 1 个主线程代理、3 个数据采集代理

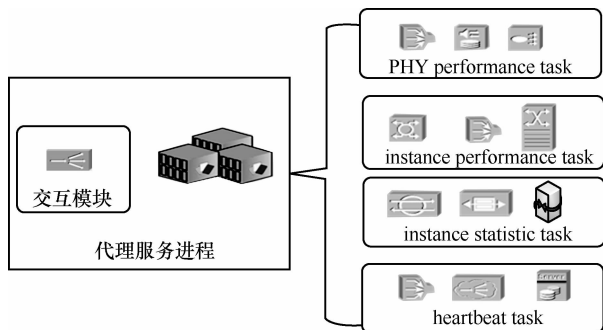


图 3 数据采集服务结构

表 2 协线程信息

协线程名称	采集内容	采集模块
PHY performance task	物理主机各项运行信息	Python-psutil
instance performance task	CPU、内存、磁盘 I/O	Python-libvit task
instance statistic task	磁盘使用量、网络传输量	Python-libvit
heartbeat task	采集点服务健康心跳信息	无

和 1 个心跳协线程组成,如图 3 所示,每个协线程按照其自身的数据采集类型采用不同数据获取模块进行,各个协线程信息见表 2。

其中,Python-psutil 直接提供了物理主机运行的各项数据信息,而 host_infor.PHY performance task 对所采集信息进行 JSON 数据格式的封装;对虚拟机信息采集的 Python-libvit,未向 Python-psutil 提供包括 CPU 使用率的信息,因此 instance performance task 采取了如式(1)所示的方式进行 CPU 使用率的计算^[12]。

$$\text{虚拟机 CPU 使用率} = \frac{\text{libvirt 获取虚拟机 CPU 使用时间 } T_1 - \text{libvirt 获取虚拟机 CPU 使用时间 } T_2}{\text{系统采集点时间 } t_1 - \text{系统采集点时间 } t_2} \quad (1)$$

为了实现多交互的多模式融合,在服务与数据汇聚存储 Daemon 服务的交互模式上采用了动态加载功能设计结构——IMPL,该结构可依据配置文件的内容

加载相应的功能模块,其结构如图 4 所示。服务目前支持 HTTP 接口交互模块和基于消息队列的 Zeromsg 模式交互模块。

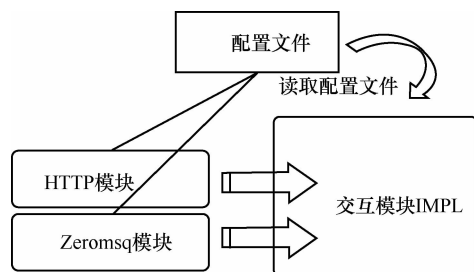


图 4 动态模块加载结构示意图

5 面向数据对象的数据汇聚 Daemon 服务的设计与研制

数据汇聚服务主要是将各个采集点上报的数据进行汇聚并分类存入数据库中,由此可知,数据汇合服务主要是完成对数据库的读写操作。传统的数据库读写操作主要是按照数据库的操作语言格式完成对数据库的操作,在实际开发过程中,数据库的操作指令格式较为固定,开发效率低,扩展性差。目前将数据库的表抽象为对象,按照抽象后的对象方法完成对数据库的读写操作开发模式成为数据库程序的首选开发框架。基于上述考虑,系统所设计的数据汇聚服务以 SQL Alchemy 为框架, MongoDB 为数据库支撑平台进行设计和开发,其结构如图 5 所示。

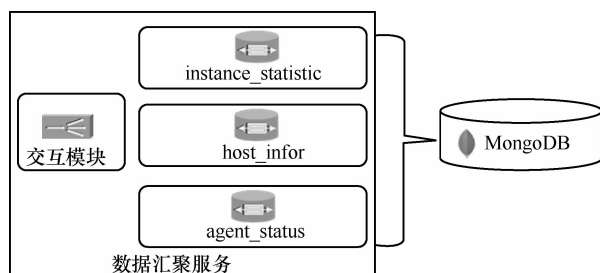


图 5 数据汇聚服务结构框架

SQL Alchemy 是一个能兼容众多数据库 (如 SQLite、MySQL、Postgres、Oracle、MS-SQL、SQL Server 和 Firebird) 的企业级持久性模型,是一种对象关系映射器(object relational mapper, ORM),用 ORM 等独立 SQL Alchemy 的优势在于它允许开发人员首先考虑数据模型,并能决定稍后可视化数据的方式^[8]。

在数据汇聚服务的设计上,系统直接将整个 MongoDB 进行抽象,其关键部分代码如下:

```
# 将系统 MongoDB 数据对象——wlee_db 抽象为对象
def get_database(db_name = 'wlee_db'):
    conn = get_connection()
```

```
return conn[db_name]
```

再按照所需要存储的数据表类型,以 SQL Alchemy 为基础将所需要存储的数据表抽象为 4 个数据映射对象,关键代码如下:

```
# 虚拟云主机运行信息存储表对象
```

```
ensure_collection ('instance_perf_recently', db_name =
'wlee_db',
```

```
indexes = [['name', pymongo.ASCENDING], ('timestamp',
pymongo.DESENDING)],
```

```
options = {"capped": True, "size": INST_RECENTLY_
PERF_SIZE,
```

```
"max": INST_RECENTLY_PERF_SIZE))
```

```
# 虚拟云主机运行统计信息数据存储表对象
```

```
ensure_collection('instance_statistic', db_name='wlee_db')
```

```
# 物理主机运行信息存储表对象
```

```
ensure_collection('host_infor', db_name = 'wlee_db')
```

```
# 物理机心跳运行信息存储表对象
```

```
ensure_collection('agent_status', db_name = 'wlee_db')
```

在实际存储操作时,服务通过对采集数据类型的分类,调用数据库对象的数据插入方法实现数据的写入,以存储虚拟云主机运行信息为例,部分关键代码如下:

```
# 获取数据库对象
```

```
db = get_database()
```

```
# 获取对应的虚拟云主机运行信息数据表对象
```

```
perf_recently = db.instance_perf_recently
```

```
# 插入数据信息,完成写入
```

```
return perf_recently.insert(perf_data)
```

在与数据采集交互模式上,为了与数据采集服务保持一致,服务同样采用动态加载功能设计结构——IMPL。

6 对外管道式 API 服务模块的设计

对外接口服务是云监控与用户进行信息交互的窗口,主要按照用户发送的数据请求完成对数据库信息的查询,并将结果返回到用户端。

系统在对外接口服务上采用 Paste Deploy 方式进行设计,主要结构如图 6 所示。该结构采用了管道模式,即将用户的网络信息处理分为首尾相连的多个子模块,每个子模块负责自己的信息处理,一般而言,每个过滤器用以实现对用户数据请求的预处理,包括网络地址端口的检查、用户信息认证等,位于管道终端的是应用服务处理模块,提



供最终的用户请求处理^[12];在扩展性方面,管道的连接可以通过配置文件进行灵活的构建,采用 Paste Deploy 方式可以灵活地实现中间件的请求处理方式^[13],从而能够与第三方软件进行整合。

如图 6 所示,目前主要提供两条管道,即服务版本查询管道和用户数据处理管道,对应的配置文件内容如下:

```
[pipeline:wlee_versions] # 服务版本查询管道
pipeline = wlee_versions_app

[pipeline:wlee_api_v1] # 用户数据处理管道
pipeline = fault_wrap wlee_api_app
```

因此实际 API 服务管道地址为 hostIP:port + 管道地址,其管道地址配置如下:

```
[composite:wlee]
use = egg:Paste#urlmap

/: wlee_versions # 服务版本查询管道地址
/v1: wlee_api_v1 # 用户数据处理管道地址
```

服务的用户数据处理管道主要由 1 个滤波器和 1 个应用组成。其中,filter fault_wrap 用于对用户请求数据进行处理,对用户数据请求的格式做检查,包括 HTTP 请求方式、JSON 数据格式等,application api_app 采用路由器进行设计,按照用户不同的请求分发到相应的数据处理服务。由于路由器结构可以按照用户的数据请求的子地址进行数据的分发,因此可将不同用户数据处理服务指定到多个处理模块,例如,用户查询物理服务主机信息的数据请求地址为: http://HostIp:port/v1/phyinfor/{host_name}/performance; 用户查询虚拟主机运行信息的数据请求地址为: http://HostIp:port/v1/instance/{instance_id}/performance, 即 application api_app 按照用户的数据请求类型转发到相应的数据处理模块中完成对数据库信息的查询,实现对用户数据的处理响应。

服务查询管道提供用户对当前监控软件版本信息的查询,由于该服务仅仅返回一个固定的 JSON 结构的数据——{"version": "1.0"}, 因此该管道不需要对用户的数据进行检查,仅由自身的应用服务——wlee_versions_app 组成,当用户向该管道地址发送 HTTP 请求时,该应用直接返回版本 JSON 数据。

7 实验测试

云监控系统采用 OpenStack 平台进行测试,采用 Grizzly3 版本,分布式部署,计算节点为 10 台 x86 工作站。测试方案采用立体纵向功能测试和横向数据对比的模式,即采用 HTTP 指令和 Web 客户端分别对系统的功能进行纵向测试;在监控数据的准确性方面,将云监控采集的数据和被监控系统自身的运行采集信息进行横向对比。

(1) 系统功能测试

分为如下 2 个步骤。

第 1 步: 直接采用 HTTP 指令实现 API 数据的交互,以 Linux 系统下的 curl 工具模拟客户端进行交互,curl 是利用 URL 语法在命令行方式下工作的文件传输工具,它支持的协议有:FTP、FTPS、HTTP、HTTPS、GOPHER、Telnet、DICT、FILE 以及 LDAP^[15]。主要测试 3 类数据的获取,以验证系统功能服务的正确性,指令格式见表 3。

表 3 curl 测试指令格式

curl 指令格式	指令说明
PHY performance task	Python-psutil
instance performance task	Python-libvit
instance statistic task	Python-libvit

在测试过程中,系统对表 3 的各个指令进行 100 次数据获取,运行时间和正确性统计结果部分的测试返回结果如图 7 所示。

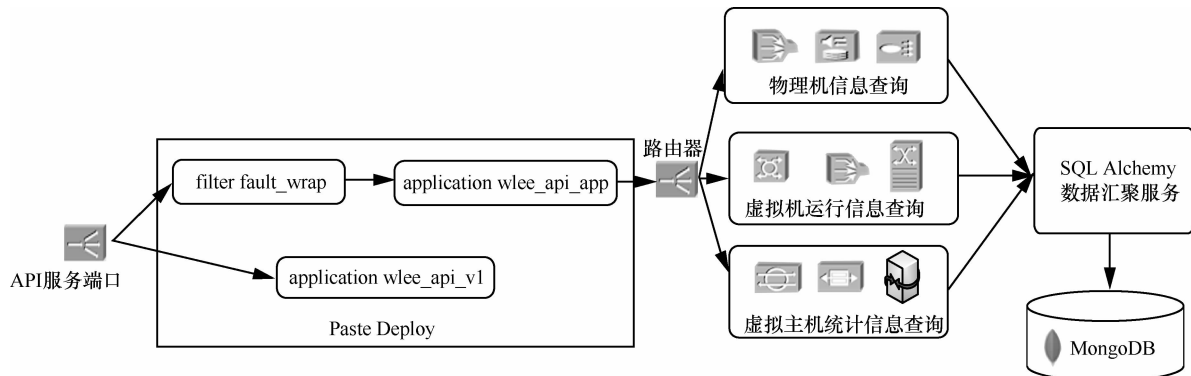


图 6 外管道式 API 服务模块

```
</body>
</html>test@test-OptiPlex-990:~$ curl http://127.0.0.1:8998/v1/phyinfor/test-OptiPlex-990/performance
{"phy_cpu_num": 8, "eth0_send": 70, "disk_usage_percent": 1.3, "disk_io_write_percent": 502, "eth1_rcv": 0, "phy_net_num": 1, "disk_usage_tot
ble": "2013-06-30 14:14:40.973478", "phy_cpu_percent": 24.3, "eth0_rcv": 53, "ram_available": 9940299776, "timestamp": 1372572880, "host_name
": 12476309504, "vm_num": 1, "eth2_rcv": 34433}test@test-OptiPlex-990:~$ curl http://127.0.0.1:8998/v1/instance/instance-00000014/statistic
{"host": "instance-00000014", "disk_stats": {"vda": {"total": 32212254720, "used": 252510208}}, {"name": "instance-00000014", "last_updated_at"
27.0.0.1:8998/v1/instance/instance-00000014/statistic
{"host": "instance-00000014", "disk_stats": {"vda": {"total": 32212254720, "used": 252510208}}, {"name": "instance-00000014", "last_updated_at"
27.0.0.1:8998/v1/instance/instance-00000014/statistic
{"host": "instance-00000014", "disk_stats": {"vda": {"total": 32212254720, "used": 252510208}}, {"name": "instance-00000014", "last_updated_at"
27.0.0.1:8998/v1/instance/instance-00000014/statistic
{"host": "instance-00000014", "disk_stats": {"vda": {"total": 32212254720, "used": 252510208}}, {"name": "instance-00000014", "last_updated_at"
27.0.0.1:8998/v1/phyinfor/test-OptiPlex-990/performance
{"phy_cpu_num": 8, "eth0_send": 70, "disk_usage_percent": 1.3, "disk_io_write_percent": 502, "eth1_rcv": 0, "phy_net_num": 1, "disk_usage_tot
ble": "2013-06-30 14:14:40.973478", "phy_cpu_percent": 24.3, "eth0_rcv": 53, "ram_available": 9940299776, "timestamp": 1372572880, "host_name
": 12476309504, "vm_num": 1, "eth2_rcv": 34433}test@test-OptiPlex-990:~$
```

图 7 curl 功能测试部分数据返回结果

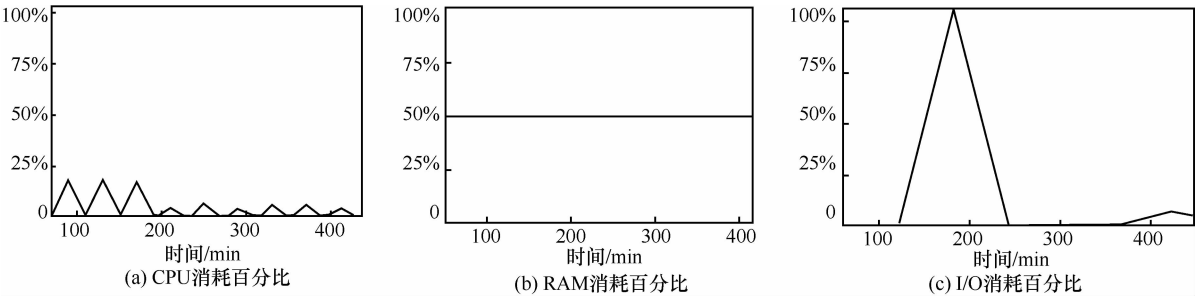


图 8 Web 端测试效果

第 2 步：考虑到监控一般以可视化的方式呈现给用户,因此采用 Web 客户端工程进行测试,在对客户端服务整合进行验证的同时,进一步完善监控系统的构建。客户端采用基于 Python 结构的 tornado 工程进行设计,同时结合 JavaScript 图形库实现 Web 图标绘制。

在实际测试过程中,随机选取了集群内 Hostname 为 op-01 的物理机和名字为 instance-000000x 的虚拟云主机为对象进行测试,Web 客户端的运行测试效果如图 8 所示。

在运行 24 h 后,Web 服务仍然能正常服务,可知监控系统在功能服务上具有正确性、稳定性以及与第三方等 Web 终端的功能服务稳定。

(2)系统监控数据准确性对比

为了能有效地对系统采集信息的准确性进行验证,依据系统调用采集模块的特点进行了类比分析。流程如下：

- 物理机信息采集模块采用了 Python-psutil 模块,该模块可直接获取宿主机的完整运行信息,因此无需对该项指标进行对比测试;
- 虚拟云主机信息采集模块通过 Python-libvirt 获取包括 CPU 使用时间等中间数据,然后按照相应的计算公式转换为对应的运行信息,因此需要对虚拟机的实际运行信息和系统换算的监控信息进行比较。

将在虚拟机内部直接采用的 top shell 指令与系统获取的信息进行对比,结果见表 4。

表 4 测试数据准确性对比

采集信息类型(每小时平均值)	系统采集信息	top 指令
CPU 占用率	15%	13%
内存使用量	512 MB	505 MB
硬盘 I/O 压力(写入)	43 555 667 byte	43 567 889 byte
硬盘 I/O 压力(读出)	29 968 829 byte	29 668 552 byte
网卡流量	32 263 byte	32 568 byte

结果表明,在系统数据采集的正确性方面与 top 指令的最大相对误差小于 5%,能较为准确地表现云计算平台的运行情况。

8 结束语

云监控系统以 OpenStack 工程体系为基础进行开发,结合包括工厂模式、Paste Deploy、管道、Eventlet 协程、面向对象的数据库 SQLAlchemy 等多种编程模式,具有管道结构、流水线、动态模式加载、协同程等特点,与现有的 OpenStack Ceilometer 监控软件相比,由于都是基于 Oslo 工程进行设计,在性能和扩展方面具有相同的特点,同时在设计时专门面向云监控需求进行开发,去除了 Ceilometer 对于计费等功能冗余代码,具有代码紧凑、配置简易的



特点,降低了代码二次开发的难度;与 Zabbix 等第三方监控软件相比,由于采用了模块化设计,体现了高扩展性、二次开发简单的优点。系统所采用的采集模块未针对任何特定的云计算平台架构,经测试表明,系统具有网络适应性强、安全性及可靠性高的特点。

参考文献

- 1 沈青,董波,肖德宝. 基于服务器集群的云监控系统设计与实现. 计算机工程与科学, 2012,34(10):73~77
- 2 Mikyung K, Dong-In K, Stephen P. *et al.* Design and development of a run-time monitor for multi-core architectures in cloud computing. *Sensors*, 2011,11(4):3595~3610
- 3 Shao J, Wei H, Wang Q X, *et al.* A runtime model based monitoring approach for cloud. *Proceedings of IEEE 3rd International Conference on Cloud Computing, Florida*, 2010: 313~320
- 4 赵海川,吴江,王鹏等. 面向电信业务的基础设施云监控系统. 西北大学学报(自然科学版), 2012,42(5):729~734
- 5 Ruan L, Peng J B, Xiao L M, *et al.* Distributed virtual machine monitor for distributed cloud computing nodes integration. *Lecture Notes in Computer Science*, 2013(7861): 23~31
- 6 DeChaves S A, Uriarte R B, Westphall C B. Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, 2011 (49):103~137
- 7 Eventlet. <http://eventlet.net/>, 2012
- 8 Rick C. *Essential SQL Alchemy*. O'Reilly Media Inc, 2008
- 9 Stacklab:documentation:OpenStack-hacker. <http://wiki.stacklab.org/doku.php?id=stacklab:documentation:OpenStack-hacker>, 2013
- 10 Thamarai S, Somasundaram, Kannan G. Cloud monitoring and discovery service (CMDS) for IaaS resources. *Proceedings of the*

- Third International Conference on Advanced, India, 2011:340~345
- 11 董波,沈青,肖德宝. 云计算集群服务器系统监控方法的研究. 计算机工程与科学, 2010,34(10):69~72
- 12 Get CPU usage with get system times. <http://www.codeproject.com/Articles/9113/Get-CPU-Usage-with-GetSystemTimes>, 2011
- 13 Paste Deployment. <http://pythonpaste.org/deploy/>, 2012
- 14 Zhao W B, Melliar-Smith P M, Moser L E. Fault tolerance middleware for cloud computing. *Proceedings of IEEE the 3rd International Conference on Cloud Computing, Florida*, 2010:67~74
- 15 Curl. <http://baike.baidu.com/view/1326315.htm>, 2011

[作者简介]



李磊,男,华南理工大学工程师,主要研究方向为云计算。

金连文,男,华南理工大学教授、博士生导师,主要研究方向为移动互联网、云计算、图像处理、机器学习。

黎德生,男,华南理工大学硕士研究生,主要研究方向为云计算、移动互联网终端技术、图像处理。

唐万万,男,香港云龙科技有限公司工程师,主要研究方向为云计算技术、网络集群化服务、公有云服务技术。

(收稿日期:2013-07-30)

(上接第 23 页)

- 3 石莎. 移动互联网网络安全认证及安全应用中若干关键技术研究. 北京邮电大学硕士学位论文, 2012
- 4 杜久升,陈宜金. 计算机工程与应用. 移动数据传输安全性研究, 2011, 47(34):11~14
- 5 Butter T, Aleksy M. MixVM-An approach to service isolation and data protection in mobile context-sensitive applications. *Proceedings of 2009 International Conference on Availability, Reliability and Security, ARES 09, Fribourg, Switzerland*, 2009:322~328
- 6 Nkosi M, Mekuria F. Improving the capacity, reliability & life of mobile devices with cloud computing. *Proceedings of IST-Africa Conference, South Africa*, 2011:1~9
- 7 Wonjoo P, Sun J K, Kee S. Framework for security management of mobile devices. *Proceedings of Communications and Information Technology, ISCIT 2009, Korea*, 2009:1219~1222
- 8 Wu J J, Zhou J, Ma J, *et al.* An active data leakage prevention

model for insider threat. *Intelligence Information Processing and Trusted Computing (IPTC)*, 2011(2):39~42

[作者简介]



杨育斌,男,蓝盾信息安全技术股份有限公司高级工程师、副总经理,主要研究方向为云计算。

程丽明,女,蓝盾信息安全技术股份有限公司高级工程师、高级研究员,主要研究方向为云计算。

(收稿日期:2013-07-01)