基于 OpenStack 资源监控系统①

梁 宇[1,2], 杨海波[2], 李鸿彬[2], 兰国亮[1,2]

[1] (中国科学院大学, 北京 100049)
[2] (中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘 要: 资源监控是提高云平台可靠性的重要手段. 本文结合 OpenStack 云平台的特点, 设计并实现了一个全面、智能、高效的资源监控系统, 完成了资源监控系统整体架构的设计以及各模块的功能划分, 并给出了实例监控方式、数据存储模型等具体的实现方法. 最后, 通过测试结果说明资源监控系统的有效性.

关键词: 云计算; OpenStack; 资源监控系统

Resource Monitoring System Based on OpenStack

LIANG Yu[1,2], YANG Hai-Bo[2], LI Hong-Bin[2], LAN Guo-Liang[1,2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology of Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: Resource monitoring is an important way to improve the reliability of cloud platforms. In this paper, we design and implement a comprehensive, intelligent, efficient resource monitoring system based on the OpenStack cloud platform features and complete the overall architecture design and function of each module division of this system, and then we give the case of monitoring methods, data storage model and other specific implementations. Finally, the test results show the effectiveness of resource monitoring system.

Key words: cloud computing; OpenStack; resource monitoring system

随着互联网技术的发展, 计算机行业不断涌现出一些新的技术, 从分布式计算、并行计算逐渐发展到网格计算, 又随着资源虚拟化技术的成熟, 孕育出了云计算.

云计算(Cloud Computing)是一种新兴的商业计算模型, 云计算平台把庞大的基础设施、数据存储、软件组成相互共享, 构成了一个庞大的资源池, 使用户能够按需要获取计算力、存储空间和信息服务[2]. 并在此基础上抽象出层次化服务, 以付费使用的方式向用户提供诸如基础架构(IaaS)、平台(PaaS)、软件(SaaS)等服务. 云服务使人们不必关心底层的具体实现, 只需要把大量的计算、存储放到“云”中处理, 返回处理结果, 使用起来简单方便. 近年来, Google、亚马逊、IBM 等公司都在积极的推自己的云服务, 云计算已经从概念落实到实际的应用中, 已发展为可个性化定制、伸缩可发展、面向服务的公有云或私有云.

监控是云计算平台的重要组成部分, 它是云计算平台中网络分析、系统管理、作业调度、负载均衡、事件预测、故障检测以及恢复操作的前提, 可以帮助云计算平台动态量化资源使用、检测服务缺陷、发现用户使用模式、辅助资源调度, 对提高云计算平台的服务质量发挥着重要的作用[5].

现有的监控技术大多只针对网格集群监控而设计, 相对云计算平台的特点, 不能满足实际应用的需求. 同时, 有些专有的监控系统功能比较单一, 只能直对某个领域进行监控. 然而, 在云计算平台中, 用户暴露给虚拟服务的层次不同, 低层次的资源情况对使用高层次的用户是透明的, 导致用户不能自由部署他们自己的监控设施, 所以网格和集群系统上的很多监控方法不能有效的工作. 因此, 我们要设计一套属于云

① 收稿时间:2013-08-20;收到修改稿时间:2013-09-24

计算平台自己的资源监控系统.

1 OpenStack介绍

OpenStack 是一个旨在为公共及私有云的建设与管理提供软件的开源项目, 开发者将 OpenStack 作为基础设施即服务(IaaS)资源的通用前端. OpenStack 项目的首要任务是简化云的部署过程并为其带来良好的可扩展性, 帮助大家利用 OpenStack 前端来设置及管理自己的公共云或私有云.

OpenStack 是由 Rackspace 和 NASA 共同开发的云计算平台, 帮助服务商和企业内部实现类似于 Amazon EC2 和 S3 的云基础架构服务(Infrastructure as a Service, IaaS). OpenStack 包含两个主要模块: Nova 和 Swift, 前者是 NASA 开发的虚拟服务器部署和业务计算模块; 后者是 Rackspace 开发的分布式云存储模块, 两者可以一起用, 也可以分开单独用. OpenStack 是一个开源项目, 除了有 Rackspace 和 NASA 的大力支持外, 后面还有包括 Dell、Citrix、 Cisco、Canonical 这些重量级公司的贡献和支持, 发展速度非常快, 有取代业界内一些领先的开源平台的趋势.

OpenStack 云计算平台中有两种类型的物理节点, 分别为控制节点和计算节点. 控制节点包括网络控制、调度管理、api 服务、存储卷管理、数据库管理、身份管理和镜像管理等, 计算节点主要提供 nova-compute 服务.

在 OpenStack 的部署和运行中, 对实例的运行详情进行监控是很有必要的, 尤其是涉及到计费、调度和迁移等操作时, 谅解实例的即时信息和历史信息, 可以帮助系统进行合理的决策. OpenStack 主要通过 Horizon 来进行操作和查看实例, 在现有 Horizon 中只有一些简单的虚拟机概况, 如运行、暂停还是挂起, 如果想监控虚拟机详细的情况, 如 cpu 使用、内存使用、网络读写和磁盘读写, 就需要设计一个完整的资源监控系统来完成对 OpenStack 云计算平台的实时监控.

2 OpenStack资源监控系统的设计

2.1 整体架构的设计

资源监控系统是一个 OpenStack 的监控和度量工具, 它可以跟踪用户的资源使用情况, 包括虚拟机的 CPU 使用率、内存使用、网络流量和磁盘读写情况, 同时可以进行数据的汇总和统计. 下图显示资源监控系统的整体结构, 及各模块之间的联系.

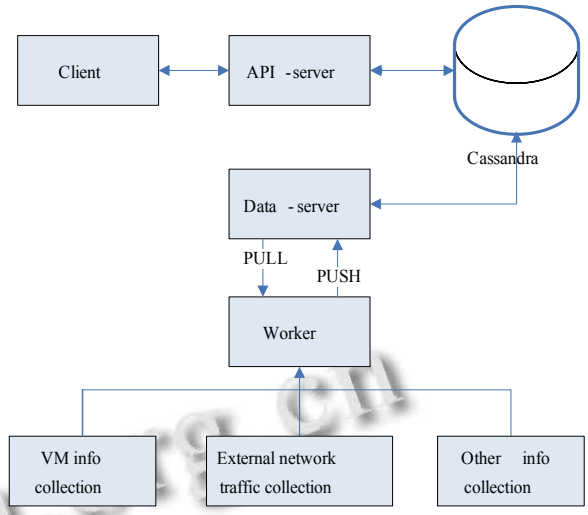

图 1 资源监控系统结构图

2.2 开发环境及通讯协议

本系统采用 python2.7 作为开发语言, Cassandra 作为系统数据库, 通讯协议全部使用 json 格式.

2.3 系统划分

资源监控系统分为五个模块, 分别是 worker、server、database、common、client.

其中 worker 和 server 是两大主要模块, 也是整个资源监控系统的核心模块. server 模块中包括 data-server 和 API-server 两大服务器模块. worker 模块负责采集监控数据并交由 data-server 模块进行存储, API-server 模块则负责提供进行查询和统计的外部接口.

2.4 各模块功能的设计

2.4.1 worker 模块

worker 模块运行在计算节点上, 负责对所在服务器上的实例进行数据采集. 采集有一个时钟周期, 每到一个时间进行一次采集, 并且通过 socket 发送给 server 服务.

worker 模块采集数据的主要形式:

{'instanceid1':('IP',time,'value'), 'instanceid2':('IP', time, 'value')} 其中 value 的单位为 KB.

虚拟机信息的采集:

[('cpu', 'total', (utc_time, cpu_usage)),
('mem', 'total', (utc_time, max, free)),
('nic', 'vnet8', (utc_time, incoming, outgoing(内网))),
('blk', 'vda', (utc_time, read, write)),
('blk', 'vdb', (utc_time, read, write))],

其中 cpu 和 mem 为实际用量.

worker 模块向 server 模块提交数据的形式:
self.send([msg_type, json.dumps(info)])
其中包括采集的 info 信息和本次发送信息的类型 msg_type

2.4.2 server 模块

server 模块运行在控制节点, 负责对 worker 发送来的信息进行处理, 并将其存储在 cassandra 数据库中.

该模块包括 data-server 和 API-server 两大服务器模块. data-server 模块负责将采集的监控数据进行存储; API-server 模块负责提供进行查询和统计的外部接口.

server 模块中数据接收和存储的方式方法:
msg_type, report = socket.recv_multipart()
数据接收使用;
plugins[msg_type](app=app, db=db, data=data)
获得数据类型和内容, 然后再使用.

2.4.3 database 模块

该模块给出对后台数据库进行增删改查的方法, 为所有与数据库相关联模块提供机制支持.

2.4.4 common 模块

common 模块主要是在数据的采集分类归并处理过程中, 对数据存储方式进行划分; 以及在数据的收集过程中遇到不可预知的 bug 时, 资源监控系统所采用的安全机制.

common 模块将数据的存储方式分为两种, 一种是以缓存的形式进行存储, 另一种是永久存储.

在数据采集返回过程中遇到不可预知的 bug 时, 资源监控系统所采用的安全机制为 loging.

common 模块中的 app.py 子模块的主要功能就是对 log 的处理. 通过几个主要函数来实现获取 logger 以及输出 debug、warning 等各类 log 信息.

2.4.5 client 模块

client 模块的功能是接收用户的查询信息, 并由 API-server 模块提供的外部接口提交给 server 模块进行各种处理, 最后返回用户需要的查询结果.

2.5 配置文件的设置

2.5.1 controller 端

```
[kanyun]
log: /var/log/kanyun/kanyun.log
[server]
host: 192.168.139.50
port: 5551
db_host: 127.0.0.1
log: /var/log/kanyun/kanyun-server.log

[api]
api_host: 192.168.139.50
api_port: 5552
db_host: 127.0.0.1
log: /var/log/kanyun/kanyun-api.log

[client]
api_host: 192.168.139.50
api_port: 5552
log: /var/log/kanyun/kanyun-client.log
```

2.5.2 compute 端

```
[kanyun]
log: /var/log/kanyun/kanyun.log

[worker]
id: worker1
worker_timeout: 60
dataserver_host: 127.0.0.1
dataserver_port: 5551
log: /var/log/kanyun/kanyun-worker.log

[client]
api_host: 192.168.139.50
api_port: 5552
log: /var/log/kanyun/kanyun-client.log
```

3 OpenStack资源监控系统的实现

3.1 实例监控方式

OpenStack 同时支持 Xen 和 KVM 虚拟化管理, 它通过 Libvirt 来操作这些虚拟机. Libvirt 库是一种实现 Linux 虚拟化功能的 Linux API, 它支持各种虚拟机监控程序, 包括 Xen、KVM 以及用语其他操作系统的一些虚拟产品. 想要进行虚拟机监控, 就需要调用已有 Libvirt 的功能来实现.

3.1.1 CPU 监控

Libvirt 不可以直接获得 CPU 使用率, 但是可以通过间接的方式计算. 首先通过 Libvirt 获得 CPU 的运行时长, 通过两次时长的差值和现实时间的差值比, 粗

略计算虚拟机对 CPU 的使用率.

3.1.2 网络流量和硬盘读写监控

通过解析 KVM 为每一个虚拟机建立的配置文件, 可以获得该虚拟机详细的网卡和设备编号, 然后通过编号调用 interfaceStats()和 blockStats()函数来获得具体设备的使用情况.

3.2 数据存储模型

采集到的数据是简单而大量的, 而且随着时间的推移和节点的增加. 操作也相当频繁, 使用 MySQL 数据库可能会对 OpenStack 其他功能产生影响, 所以在这里存储使用了 NoSQL 文档型数据库 Cassandra, 它在存储简单格式的数据上表现良好, 适合做分布式结构化存储.

数据模型如下:

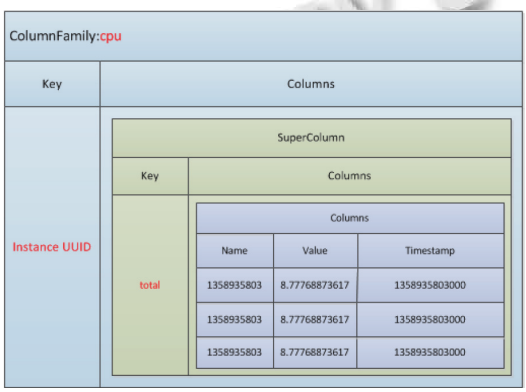

图 2 资源监控系统数据模型图

每一项统计都作为一个 ColumnFamily, 如 cpu、mem_read 等, 将 instance id 作为 key. 读取时需要依靠时间戳获取数据, 进行分析统计.

4 OpenStack资源监控系统测试及分析

4.1 资源监控系统的测试

在 OpenStack 云计算平台上安装部署资源监控系统, 并针对资源监控系统进行了相关性能的测试.

首先, 在 OpenStack 的计算节点上建立了 3 个测试节点, 配置信息如下:

表 1 测试节点配置信息表

	测试节点 1	测试节点 2	测试节点 3
操作系统	CentOs5.5	CentOs5.5	Ubuntu12.04
内存	2G	1G	2G
硬盘	20G	10G	20G

测试节点建立完成后, 对资源监控系统进行测试, 分别对 3 个测试节点进行 CPU 使用率及网络流量的测试.

对 CPU 使用率进行测试, 在 3 个测试节点上分别运行不同的进程, 实时统计每个节点的 CPU 使用情况, 统计间隔为 1 小时, 截取部分测试结果如下:

表 2 CPU 使用率测试结果统计表

	测试节点 1	测试节点 2	测试节点 3
9:00	28%	36%	32%
10:00	32%	40%	35%
11:00	29%	42%	38%
12:00	28%	53%	41%
13:00	33%	49%	38%
14:00	35%	46%	39%
15:00	31%	47%	36%

对网络流量进行测试, 在 3 个测试节点上分别在线进行下载、浏览网页等不同的活动, 实时统计每个测试节点上的网络流量情况, 统计间隔为 1 小时, 截取部分测试结果如下:

表 3 网络流量测试结果统计表

	测试节点 1	测试节点 2	测试节点 3
9:00-10:00	45M	31M	98M
10:00-11:00	52M	35M	110M
11:00-12:00	55M	33M	123M
12:00-13:00	50M	33M	141M
13:00-14:00	58M	28M	129M
14:00-15:00	49M	26M	118M
15:00-16:00	52M	30M	126M

资源监控系统还可以针对内存的使用等方面进行监控, 这些监控的内容是云计算平台系统管理、网络分析、故障检测等操作的前提, 一个稳定的资源监控系统对提高云计算平台的服务质量发挥着重要的作用.

4.2 资源监控系统问题的分析

资源监控系统发挥重要作用的同时也存在着一些问题, 由于资源监控系统本身是安装部署在云计算平台上, 所以监控系统会使用被监控对象的一些资源, 如 CPU 计算资源、存储资源、网络等, 那么不可避免

(下转第 16 页)

平. 从技术上看, 基于分片的虚拟技术和分布式重叠网络无疑是最受关注的技术热点, 异构网络的互联也是试验床建设着力解决的问题. 可以预期, 随着试验床建设的不断深入, 在不久的将来, 试验床一定会在新型互联网技术和服务的研究试验中扮演越来越重要的角色, 并极大的推动技术走向应用, 造福社会.

参考文献

1 Alberts D, Hayes R. Planning: complex endeavors. CCRP Publication Series. Washington. 2007.

2 Bordetsky A, Dougan A. TNT maritime interdiction operation experiments:enabling radiation awareness and geographically distributed collaboration for network-centric maritime interdiction operations. Defense Technology and Systems Symposium. 2006.

3 TNT MIO 08-4 After Action Report. http://cenetix.nps.edu. Naval Postgraduate School. Monterey. 2008.

4 Bordetsky A, Bourakov E, Looney J, etc. Network-centric maritime radiation awareness and interdiction experiments. 11th ICCRTS. Cambridge. 2006.

5 Satcliffe R.. Web 3.0: Are we there yet? The Next Wave, NSA Review of Emerging Technologies, 2009, 17(3).

6 Miller W. Cursor-on-target. Military Information Technology Online, 2004, 8(7).

7 Clement M, Bourakov E. Exploring network-centric information architectures for unmanned systems control and data dissemination. 5th AIAAC. Seattle. 2009.

8 Bordetsky A, Bourakov, E. Network on target: remotely configured adaptive tactical network. 10th Command and Control Research & Technology Symposium. 2009.

9 Gateau J, Bordetsky A. Extending simple network management protocol beyond network management: A MIB architecture for network-centric services. 13th ICCRTS. 2008.

10 Rideout B, Strickland J. Military application of networking by touch in collaborative planning and tactical environments. Naval Postgraduate School. 2007.

11 TNT 07-4 QLR, Quick Lookup Report. http://cenetix.nps.edu. Naval Postgraduate School, Monterey. 2007.

12 Conrad B, Tzanos I. A conceptual framework for tactical private satellite networks. Naval Postgraduate School. 2008.

(上接第 47 页)

的会导致被监控的平台或应用的使用效率下降. 所以, 如何做到大量监控数据的有效采集传输、高效的计算处理, 将是我们今后一段时间工作的重点, 这对提高云计算平台的使用效率起着至关重要的作用.

5 结语

本文基于OpenStack云计算平台设计一个资源监控系统, 阐述了系统的划分、功能的设计以及最终的实现. 一个稳定的资源监控系统可以使云计算平台更加完整, 同时更容易发现并解决平台中存在的一些问题. 另外, 资源监控系统也存在着会占用被监控对象一些资源等问题, 这些问题将是我们未来的主要研究方向, 解决现有问题得到一个更完整高效的资源监控系统.

参考文献

1 程辉.OpenStack 建设公有云平台实践. 2012 云计算架构师峰会.北京.2012.10.

2 Armbrust M, Fox A, et al. Above the clouds : A Berkeley view of cloud computing. Electrical Engineering and Computer Sciences, 2009.

3 Zanikolas S, Sakellariou R. A taxonomy of grid monitoring system. Future Generation Computer Systems, 2005, 21(1): 163-188.

4 Han FF, Peng JJ, Zhang W, Li Q, Li JD, Jiang QL.Virtual resource monitoring in cloud computing. Journal of Shanghai University (Engl Ed), 2011, 15(5): 381–385.

5 张棋胜.云计算平台监控系统的研究与应用[学位论文]. 北京:北京交通大学, 2011.

6 袁凯.云计算环境下的监控系统设计与实现[学位论文].武汉:华中科技大学, 2012.