目录

[一、环境需求 2](#_Toc440043731)

[1.1采集机A 2](#_Toc440043732)

[1.2 物理机B 2](#_Toc440043733)

[1.3 测试机C 2](#_Toc440043734)

[二、信息采集子系统 2](#_Toc440043735)

[2.1 系统设计 2](#_Toc440043736)

[2.1.1 信息采集系统流程 2](#_Toc440043737)

[2.1.2 信息采集系统架构 3](#_Toc440043738)

[2.2. 系统安装 4](#_Toc440043739)

[2.2.1 在A机上安装pip： 4](#_Toc440043740)

[2.2.2 在A机上安装依赖包： 4](#_Toc440043741)

[2.2.3 在A机上配置无密码登陆B物理机： 4](#_Toc440043742)

[2.2.4 建立项目数据库 4](#_Toc440043743)

[2.2.5 在cloud\_config表中添加配置项 4](#_Toc440043744)

[2.2.6 系统配置 4](#_Toc440043745)

[2.3 系统测试运行 5](#_Toc440043746)

[2.3.1 在采集机A上运行持续监控后台程序 5](#_Toc440043747)

[2.3.1在采集机A上运行API程序 5](#_Toc440043748)

[2.3.3页面展示 5](#_Toc440043749)

[2.4. API接口说明 6](#_Toc440043750)

[2.4.1 获取所有虚拟机列表 6](#_Toc440043751)

[2.4.2 设置是否抓取指定虚拟机 6](#_Toc440043752)

[2.4.3 删除指定虚拟机 7](#_Toc440043753)

[2.4.4 按时间起始结束，获取指定虚拟机数据 7](#_Toc440043754)

[2.4.5 设置抓取时间间隔（秒） 9](#_Toc440043755)

[2.4.6 设置获取虚拟机列表间隔（秒） 9](#_Toc440043756)

[2.4.7 添加物理机IP 9](#_Toc440043757)

[2.4.8 删除指定IP的物理机 10](#_Toc440043758)

[三、虚拟机录制回放子系统 10](#_Toc440043759)

[3.1. 系统设计 10](#_Toc440043760)

[3.1.1 noVNC系统设计 10](#_Toc440043761)

[3.3.2 noVNC存在的问题 11](#_Toc440043762)

[3.3.3 解决方案 11](#_Toc440043763)

[3.2 运行截图展示 12](#_Toc440043764)

[3.2.1 控制台连接与录制（windows） 12](#_Toc440043765)

[3.2.2 控制台连接与录制（linux） 12](#_Toc440043766)

[3.2.3 控制台回放列表 13](#_Toc440043767)

[3.2.4 控制台回放控制 13](#_Toc440043768)

# 一、环境需求

要求各机器之间可以ping

## 1.1采集机A

* Centos 6+
* Python 2.7+

## 1.2 物理机B

* Linux各版本
* Libvirtd

## 1.3 测试机C

* Windows

# 二、信息采集子系统

## 2.1 系统设计

### 2.1.1 信息采集系统流程



图2-1 虚拟机信息采集系统流程图

信息采集系统流程图如图2-1所示，系统开启四个线程维持抓取数据任务：

* **线程0：守护线程**

监控以下四个线程，每个0.5秒查看一次线程状态，若线程停止或被杀死则重新开启。

* **线程1：获取vhost列表**

每隔T1（interval\_travelsal）时间调用libvirtd远程qemu+ssh方式获取物理机vhost列表，更新数据库vhost表

* **线程2：读取DB中vhost条目**

每隔T2（interval\_check）时间读取数据库vhost列表一条，加入vhost队列尾部

* **线程3：采集vhost信息**

循环读取vhost队列，若不为空，则读出一条，并采集一次信息，若采集成功则将信息写入result队列尾部，若失败则删除数据库vhost表中该条vhost项（失效检测）。

* **线程4：Result写入数据库**

循环读取result队列，若不为空，则读出一条，并写入数据库result表

### 2.1.2 信息采集系统架构



图2-2 虚拟机信息采集系统结构图

虚拟机信息采集系统结构图如图2-2所示

采集系统搭建于采集节点，通过qemu+ssh通道与各物理机连同，采用访问物理机libvirtd服务方式获取虚拟机信息。其中虚拟机信息分为三部分：

1. **基本信息：**

包括UUID、虚拟机运行状态、CPU时间、CPU个数、最大内存、已使用内存、网卡标号列表、硬盘编号列表。

1. **硬盘数据信息：**

包括读请求数、写请求数、读数据大小、写数据大小、IO错误数、硬盘容量、硬盘使用量、物理磁盘用量

1. **网络数据信息**

下载包数、上传包数、下载数据大小、上传数据大小、下载错误数、上传错误数、下载丢弃、上传丢弃

获取信息时，首先通过基本信息模块获取基本信息以及网卡编号、硬盘编号，再将后两项分别送入硬盘数据模块和网络数据模块来获取硬盘和网卡数据。

## 2.2. 系统安装

### 2.2.1 在A机上安装pip：

安装python2.7+后执行以下步骤来安装pip

# wget https://bootstrap.pypa.io/get-pip.py

# python get-pip.py

### 2.2.2 在A机上安装依赖包：

# yum install python-libvirt

# pip install gevent

# pip install web.py

# pip install mimerender

# pip install mysql-python

### 2.2.3 在A机上配置无密码登陆B物理机：

1. 在A机下生成公钥/私钥对，按一次回车，它在~/.ssh生成id\_rsa和id\_rsa.pub

# ssh-keygen -t rsa -P ''

1. 把A机下的id\_rsa.pub复制到B机的~/.ssh/目录下，重命名为authorized\_keys

# scp ~/.ssh/id\_rsa.pub root@B机IP:~/.ssh/authorized\_keys

1. 测试无密码登陆： ssh B机IP 如果不需要输入密码则配置成功

### 2.2.4 建立项目数据库

在A机数据库中运行cloud\_monitor.sql文件

# cd libvirt\_monitor\_server

# mysql –u root –p

>source sql/cloud\_monitor.sql

### 2.2.5 在cloud\_config表中添加配置项

|  |  |  |
| --- | --- | --- |
| Key | value | 默认值 |
| interval\_check | 抓取数据间隔（秒） | 200 |
| interval\_travelsal | 连接物理机，获取虚拟机列表间隔（秒） | 200 |
| host | 物理机IP | 无 |

其中host项可添加多条数据，每条数据一个IP

### 2.2.6 系统配置

在code/cloud\_monitor\_setting.py中填写各项配置，示例如下

# 数据库配置

db\_engine = 'mysql'

db\_server = '数据库IP'

db\_username = '用户名'

db\_password ='密码'

db\_database = 'cloud\_monitor'

# API端口配置

api\_server\_port = 9898

## 2.3 系统测试运行

### 2.3.1 在采集机A上运行持续监控后台程序

# cd libvirt\_monitor\_server

# chmod 755 start-monitor.sh

# ./ start-monitor.sh

监控程序日志将输出到log目录下的daemon.log和monitor.log文件，运行后查看日志，若无错误输出说明运行成功。

### 2.3.1在采集机A上运行API程序

# chmod 755 start-api-server.sh

# ./ start-api-server.sh

API程序日志将输出到log目录下得api.log文件，运行后查看日志，若无错误输出说明运行成功。

### 2.3.3页面展示

完成上述步骤后在测试机C上

1. 修改html/index.html，第29行，修改为采集机A的IP及端口

var api\_url = "http://173.26.100.211:9898/instances";

1. 修改html/graph.html，第80行，修改为采集机A的IP及端口

var api\_url = "http://173.26.100.211:9898/instances";

1. 在浏览器中打开index.html，可以看到云平台所有虚拟机列表，点击每条最后一项graph链接，进入虚拟机信息图表页面，展示如下说明配置成功。
2. 在graph页面中可选择时间



## 2.4. API接口说明

以下localhost表示采集机A的IP

### 2.4.1 获取所有虚拟机列表

|  |  |
| --- | --- |
| http://localhost/instances | |
| HTTP method | GET |
| 返回值 | id: 虚拟机编号  uuid：虚拟机唯一标识  ip：虚拟机所在物理机IP  enable：是否抓取数据,1-是，0-否 |
| 返回值json示例 |  |

### 2.4.2 设置是否抓取指定虚拟机

|  |  |
| --- | --- |
| http://localhost/enable/虚拟机uuid | |
| HTTP method | POST |
| 参数 | enable：是否抓取数据1-是，0-否 |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

### 2.4.3 删除指定虚拟机

|  |  |
| --- | --- |
| http://localhost/instances/虚拟机uuid | |
| HTTP method | DELETE |
| 参数 |  |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

### 2.4.4 按时间起始结束，获取指定虚拟机数据

|  |  |
| --- | --- |
| http://localhost/ instances /虚拟机uuid | |
| HTTP method | POST |
| 参数 | stime：起始时间  etime：结束时间 |
| 返回值 | time：数据抓取时间（精确到分）  uuid：虚拟机唯一标识  name：虚拟机名称  host：所属物理机IP  state：虚拟机状态   |  |  | | --- | --- | | 0 | no state | | 1 | the domain is running | | 2 | the domain is blocked on resource | | 3 | the domain is paused by user | | 4 | the domain is being shut down | | 5 | the domain is shut off | | 6 | the domain is crashed | | 7 | the domain is suspended by guest power management | | 8 | NB: this enum value will increase over time as new events are added to the libvirt API. It reflects the last state supported by this version of the libvirt API. |   number\_cpus：cpu数量  cpu\_usage：cpu使用量（%）  max\_memory：最大内存（KB）  memory\_usage：内存使用量（KB）  vir\_interfaces：rx\_packets：下载包数  tx\_packets：上传包数  rx\_bytes：下载数据大小（bytes）  tx\_bytes：上传数据大小（bytes）  rx\_errs：下载错误数  tx\_errs：上传错误数  rx\_drop：下载丢弃  tx\_drop：上传丢弃  vir\_disks： rd\_req：读请求数  wr\_req：写请求数  allocation：已分配  rd\_bytes：读数据大小（bytes）  wr\_bytes：写数据大小（bytes）  errs"：错误数  capacity：硬盘容量  physical：物理磁盘用量 |
| 返回值json示例 | {  "time": "2015-11-14 15:54",  "uuid\_string": "ba16b497-d49d-4a0b-a00b-246209076bfa",  "name": "instance-00000014",  "host": "173.26.100.211",  "state": 1,  "number\_cpus": 1,  "cpu\_usage": "4.667",  "max\_memory": 524288,  "memory\_usage": 524288,  "vir\_interfaces": {  "tap9d6c50ed-03": {  "rx\_packets": 0,  "tx\_packets": 0,  "rx\_bytes": 0,  "tx\_bytes": 0,  "rx\_errs": 0,  "tx\_errs": 0,  "rx\_drop": 0,  "tx\_drop": 0  }  },  "vir\_disks": {  "vda": {  "rd\_req": 0,  "wr\_req": 1.331264531878126,  "allocation": 4595712,  "rd\_bytes": 0,  "wr\_bytes": 2726.429761286402,  "errs": 0,  "capacity": 1073741824,  "physical": 4653056  }  }  } |

### 2.4.5 设置抓取时间间隔（秒）

|  |  |
| --- | --- |
| http://localhost/interval/check | |
| HTTP method | POST |
| 参数 | interval：时间间隔（秒） |
| 返回值 |  |
| 返回值json示例 | {'message':'success, new check interval is xxx' } |

### 2.4.6 设置获取虚拟机列表间隔（秒）

|  |  |
| --- | --- |
| http://localhost/interval/travelsal | |
| HTTP method | POST |
| 参数 | interval：时间间隔（秒） |
| 返回值 |  |
| 返回值json示例 | {'message':'success, new check interval is xxx' } |

### 2.4.7 添加物理机IP

|  |  |
| --- | --- |
| http://localhost/host/add | |
| HTTP method | POST |
| 参数 | host：物理机IP |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

### 2.4.8 删除指定IP的物理机

|  |  |
| --- | --- |
| http://localhost/host/虚拟机IP | |
| HTTP method | POST |
| 参数 |  |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

详细调用示例请查看index.html和graph.html

# 三、虚拟机录制回放子系统

## 3.1. 系统设计

### 3.1.1 noVNC系统设计

[noVNC](http://kanaka.github.io/noVNC/)提供一种在网页上通过html5的Canvas访问机器上vncserver提供的vnc服务，需要做tcp到websocket的转化，才能在html5中显示出来。

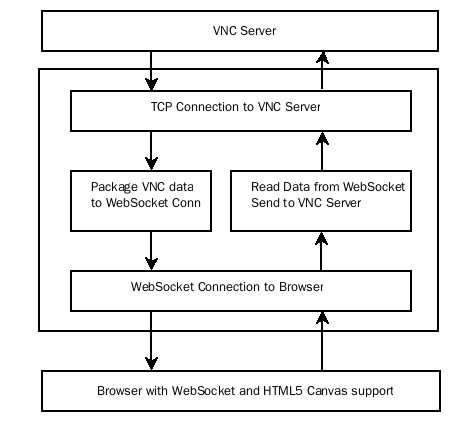


图3-1 noVNC系统架构与流程图

网页就是一个客户端，类似与vncviewer。但直接访问的不是VNCServer，而是noVNC服务器，它提供了VNC协议到WebSocket之间的转换。在noVNC代理服务器上要配置每个vnc服务，noVNC提供一个标识，去反向代理所配置的vnc服务。

如图3-1所示，是noVNC系统架构与流程图。

可以很清楚的看到，noVNC服务器是在将VNC协议的数据打包到WebSocket协议中，发送给浏览器。 并将从浏览器WebSocket连接中读取到的数据发送给VNC服务器。可以说，noVNC服务器是一个VNC代理，在浏览器和VNC服务器之间架起了一个桥梁。

### 3.3.2 noVNC存在的问题

noVNC并不是完美的，它还存在以下问题：

* 基于进程的并发，每个在独立的进程中被处理
* 自带的对VNC数据的记录和回放比较原始，容易产生性能问题

### 3.3.3 解决方案

* 将基于进程的并发改为基于轻量Greenlet的并发，使用Gevent支持，提高并发性能
* 将VNC数据的记录通过TCP协议发送到远端服务器记录，并增加数据管理
* 将VNC数据的回放通过远端服务器提供

虚拟机录制回放子系统架构与流程图如图3-2所示：

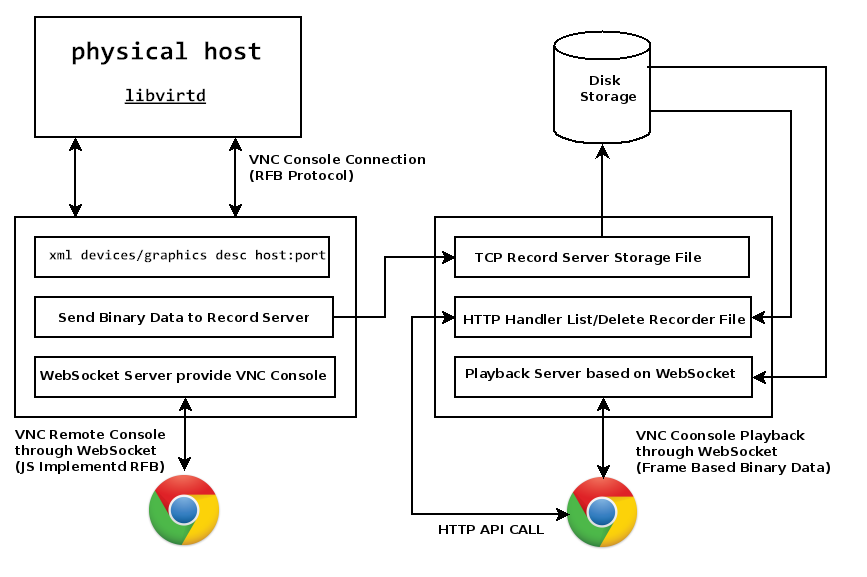


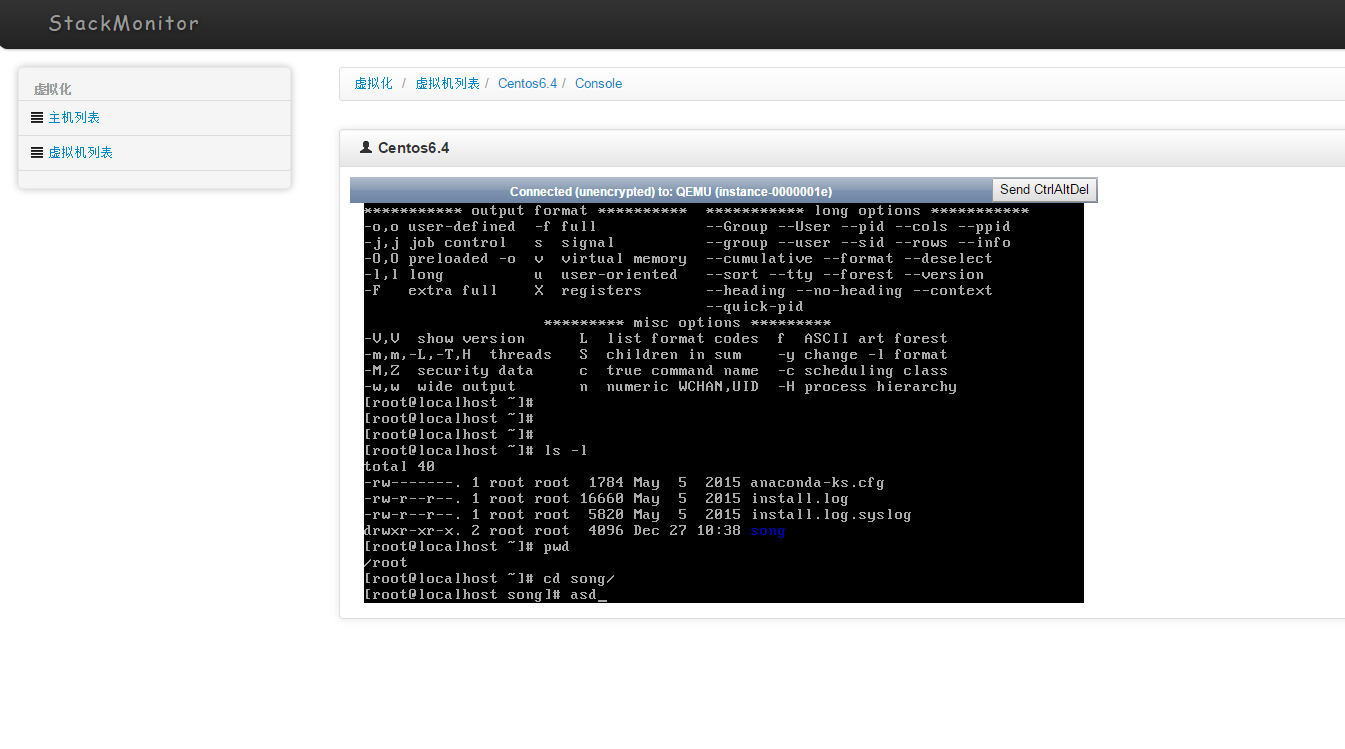
图3-2 虚拟机录制回放子系统架构与流程图

## 3.2 运行截图展示

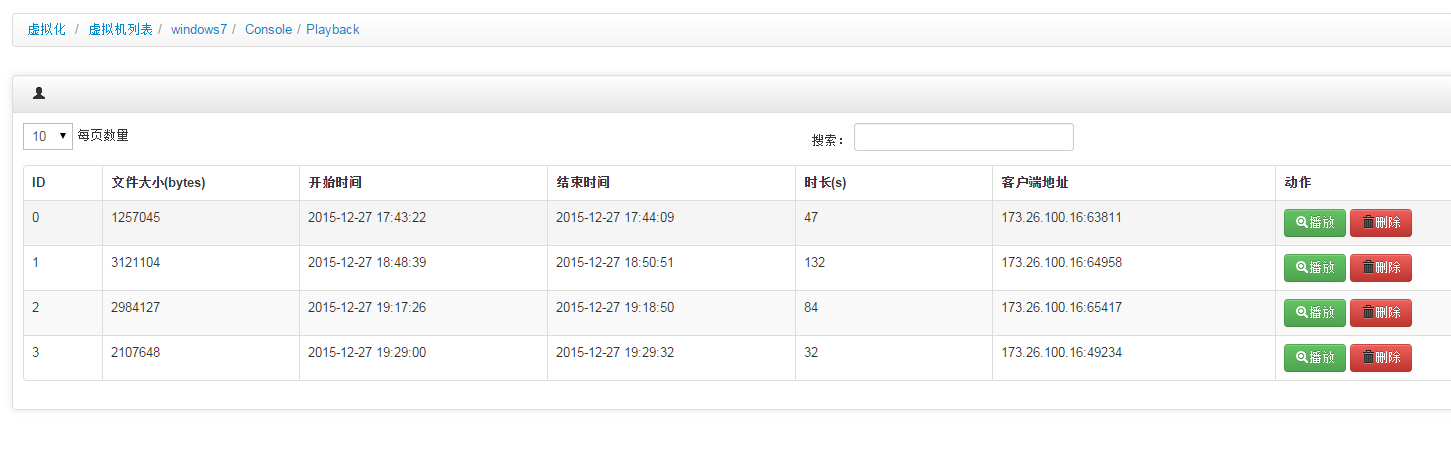
### 3.2.1 控制台连接与录制（windows）



### 3.2.2 控制台连接与录制（linux）



### 3.2.3 控制台回放列表



### 3.2.4 控制台回放控制

