**概要设计文档**

[1引言 3](#_Toc898305431)

[1.1编写目的 3](#_Toc1678718881)

[1.2背景 3](#_Toc437955980)

[1.3术语定义 4](#_Toc401430678)

[1.4参考资料 4](#_Toc567066827)

[2架构设计 5](#_Toc680949521)

[2.1需求规定 5](#_Toc74276626)

[2.1.1功能需求 5](#_Toc1260308008)

[2.1.2其他需求 5](#_Toc787432248)

[2.2运行环境 5](#_Toc1348952054)

[2.3基本处理流程 5](#_Toc1682249153)

[2.4 系统架构 6](#_Toc1645283746)

[3接口设计 7](#_Toc2038875976)

[3.1用户接口 7](#_Toc1818063961)

[3.2外部接口 7](#_Toc546159028)

[3.3内部接口 7](#_Toc506742211)

[4运行设计 8](#_Toc1694005753)

[5系统出错处理设计 9](#_Toc1000206649)

[5.1出错信息 9](#_Toc1558485523)

# 1引言

## 1.1编写目的

本文档的编写目的是对Openstack云计算系统监控的架构进行说明，为后继的详细设计等工作提供参考和依据。

## 1.2背景

云计算使用共享的硬件基础架构提供计算服务，包括计算、存储、网络等资源。云计算使得客户不需要自己搭建基础设施，只需要通过按需付费的方式就能使用计算资源。这降低了客户的花费，也使得用户的应用程序更容易扩展、可维护性更强，加速了应用的开发速度。如今已经有众多的大型IT公司和创业型企业相继推出了自己的云计算服务，并实现了业务的快速增长，云计算市场呈现出欣欣向荣的景象。在这些公司里面有的使用自己开发的系统提供云服务，如亚马逊、微软、阿里云，但更多的公司则是使用Openstack作为提供云计算服务的基础架构。

Openstack起源于2010年，是Rackspace和NASA的合作发起的开源项目，通常被用于提供IaaS服务。目前Openstack在全球众多公司和开发者的支持下不断发展，成为最受欢迎的开源云计算管理系统。Openstack是一个管理、定义、使用云计算资源的框架。Openstack可以管理物理机和虚拟机、网络、存储系统，使用租户、配额、用户角色来使得管理更加高效，并提供了控制资源的统一接口。

监控是云计算重要的部分。云服务要求极高的可用性和稳定性，因而需要对各种资源进行监控。通过观察监控数据，运维人员可以对数据中心的资源使用情况有总体的了解、发现性能瓶颈、获知系统发生故障等意外情况，在这基础上才能优化系统、处理故障、提高系统可用性。此外，监控还能记录客户的资源使用情况，进而进行计费或者建议客户购买更大的配额。

## 1.3术语定义

1. Nagios：一个开源的监控框架，简单易扩展。

2. Ganglia：一个开源的监控工具，能够获取大量节点的性能信息。

3. Graphite：一个开源的性能数据存储软件。

4. Grafana：一个开源的展示时序图表的仪表盘。

5. flask：一个python web微框架。

## 1.4参考资料

1. Kevin Jackson, Openstack Cloud Computing Cookbook Third Edition, Packt Publishing Ltd., 2015
2. V. K. Cody Bumgardner, Openstack In Action, Manning Publications Co., 2016
3. David Josephsen，Nagios系统监控实践，机械工业出版社，2014
4. Massie，Ganglia系统监控，机械工业出版社，2013
5. 詹姆斯 特恩布尔,第一本docker书，人民邮电出版社，2014
6. 格林布尔，Flask Web开发：基于Python的Web应用开发实战，人民邮电出版社，2015
7. Lorin Hochstein, Ansible: Up and runing, O’Reilly Media Inc., 2015

Jason Myers&Rick Copeland, Essential SQLAlchemy, O’Reilly Media Inc., 2016

# 2架构设计

## 2.1需求规定

### 2.1.1功能需求

* 定义要监控的主机
* 定义主机上要监控的服务
* 在仪表盘上增加或减少监控的图表
* 调节图表的大小和样式

### 2.1.2其他需求

(1) 易用性：系统应该能够提供直观友好的界面、简单的交互方式。

(2) 可扩展性：系统应能允许使用者根据自身的需求进行扩展，监控更多的指标。

(3) 可靠性：系统应提供稳定的服务。

## 2.2运行环境

操作系统：Ubuntu server 14.04；

数据库：MySQL 5.6；

运行环境：Docker容器

## 2.3基本处理流程

管理员在web界面设置要监控的机器和服务，点击确认后，系统在后台对管理员的设置进行处理，把这些操作同步到监控系统中。监控系统按照配置获取对应的机器和服务的性能信息，并把信息存储起来。管理员要查看图表的时候，系统读取这些数据，并在页面上绘制图表。

## 2.4 系统架构

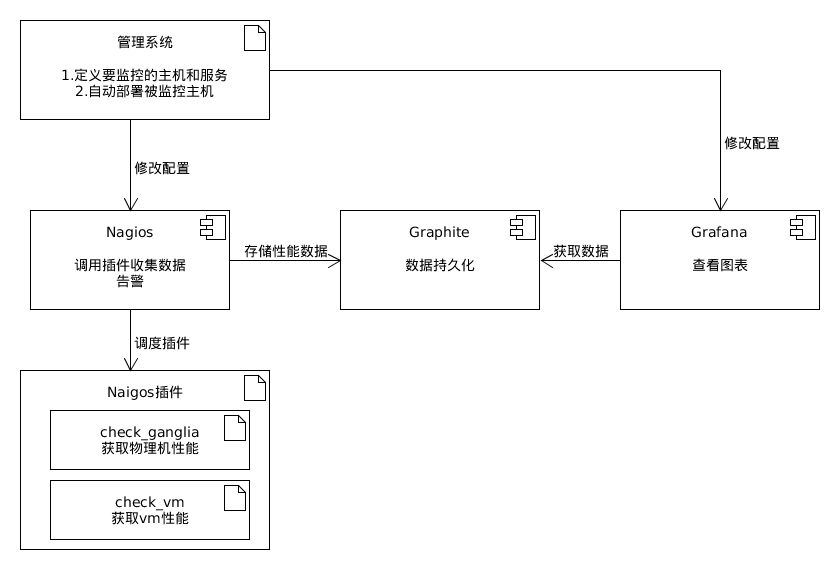


图4-1：架构图

管理系统提供了用户友好的界面方便用户定义要监控的主机和服务，并将用户的定义同步到Nagios的对象配置文件和Grafana的仪表盘。Nagios、Graphite、Grafana协作共同监控Openstack云平台的任务。Nagios负责调用插件check\_ganglia和check\_vm采集物理机和虚拟机的性能信息，并将性能信息存放到Graphite的数据库。Grafana负责将Graphite的数据展示到web页面的仪表盘上。

# 3接口设计

## 3.1用户接口

系统通过web页面为用户提供服务。Web页面设计风格简洁、重点突出、操作简单。用户可以通过web页面添加主机、服务，也可以批量添加一个IP段的主机，避免了大量的重复输入。用户可以在web页面查看仪表盘，还可以用所见即所得的方式编辑仪表盘的内容、样式。

## 3.2外部接口

该系统使用了多个开源软件，下面是系统和这些软件的接口：

Nagios：修改Nagios的配置文件，通过命令行重启Nagios服务。

Grafana：使用Grafana的API修改仪表盘。

Mysql：使用sqlalchemy来执行数据库操作。

## 3.3内部接口

内部基于功能划分为多个模块，向顶层隐藏了实现的细节。总共分为数据库操作模块、外部系统调用模块、web服务模块。数据库模块提供了数据库的操作、保证了管理系统内部的数据一致性。外部系统调用模块负责实现与外部系统的交互。Web服务模块负责渲染页面和处理用户的请求。

# 4运行设计

系统在Docker容器中运行，因此也可以运行在各种支持Docker容器的操作系统上。在Ubuntu系统没有自带Docker应用，需要用户自行安装。通过Ubuntu的apt-get安装的docker版本通常较旧，用户可以运行curl -sSL https://get.docker.io | bash命令安装最新版的docker。由于国内的网络环境不太好，下载docker镜像会相当地耗费时间，好在国内许多厂商都为我们免费地提供了docker镜像源。这里使用阿里云提供的docker镜像源。用户可以编辑/etc/default/docker文件，把DOCKER\_OPT改为 DOCKER\_OPTS="--registry-mirror=https://pee6w651.mirror.aliyuncs.com\"，并重启docker守护进程使得配置生效。

系统把建立docker镜像的Dockerfile和其他相关文件都放在了docker文件夹里面，用户进入到该文件夹后，运行docker build --build-arg LOCAL\_IP=${LOCAL\_IP} --tag mymonitor . 命令即可创建镜像。其中LOCAL\_IP是本机的IP。之后运行docker run -itd --privileged --net=host -v $(pwd)/../monitor:/opt/monitor --name mymonitor mymonitor 创建并运行容器。该命令使用特权模式创建容器，并与主机使用同一个网络名字空间，因此应该注意主机不能与容易运行相同的应用，以免端口占用造成容器内应用启动失败。此外，该命令还把管理系统的代码所在的目录挂载到了容器内。这使得修改管理系统的代码变得简单，不用每次都重新建立镜像，节省了很多时间。

# 5系统出错处理设计

## 5.1出错信息

系统中的各种提示如表5-1所示：

表5-1 系统出错提示

|  |  |
| --- | --- |
| 错误类别 | 错误处理 |
| 输入有误 | 提示错误的类型，要求用户重新输入 |
| 同步到其他软件失败 | 提示同步出错 |
| 未知的错误 | 返回500 server error |