目录

一、系统架构 2

1. 服务器集群 2

2. 数据与链路平面 3

3. 控制平面 3

4. 管理平面 3

5. 应用扩展 5

二、软件部署 6

1. Openflow Controller 6

2. Sodero Controller 6

3. Sodero GUI 6

4. 节点软件 7

5. 维护脚本 7

三、维护脚本 7

1. 批量维护脚本 7

2. 自动监控机制 9

附录1、FAQ 9

1. 光网络节点互相Ping不通 9

2. GUI无法访问 9

3. API无法访问 10

附录2、APIs 10

# 一、系统架构



## 服务器集群

服务器集群通过Arista交换机及OSU实现机柜内的以及跨机架的互通，目前绝大部分机器的光交换网络接口为eth4，IP地址为10.5.X.Y，其中X为机架号(1-12)，Y为机器号(1-15)。

服务器上部署Sodero Agent监控端，可以实时采集该节点的网络流量并汇报给Sodero控制器，同时设置了10.5.X.Y网段所有机器的静态ARP表，用来加速Openflow的转发速度(节省了每次向Openflow Controller查询ARP请求的时间)。

## 数据与链路平面

由12台Arista 10G交换机与12台Sodero OSU光交换机配对组成，Arista交换机负责所连接机柜内服务器的直接通信，以及根据Controller控制规则实现跨机柜的服务器之间转发通信。由此可见，该网络结构不再需要汇聚交换机进行跨机架转发。

Sodero OSU光交换机负责控制跨机柜的服务器连接通道。通过实时采集跨机柜的服务器 - 服务器带宽需求，自动安排最佳的转发路线，控制1)、自身光交换通道(对应Arista交换机上端口的物理连接)，2)万兆交换机的转发规则(对应Arista交换机上的Openflow规则)，从而保证和优化跨机架的服务器通信。

## 控制平面

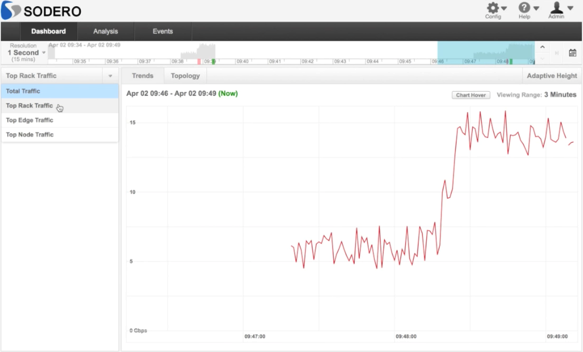
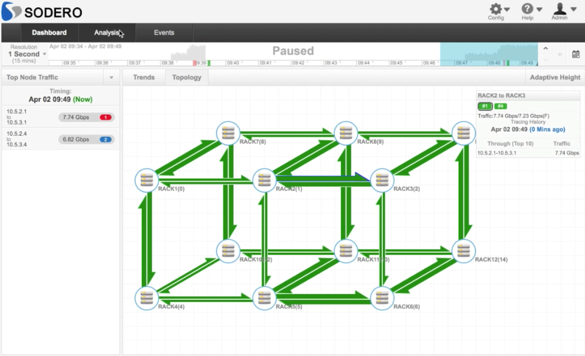
Sodero控制器是整个网络的大脑，负责将最底层的网络带宽需求，或者是通过API传入的带宽需求进行分析、解析、并转换为光交换机OSU及Arista交换机可识别的光交换协议规则以及Openflow协议规则，下发给OSU及Openflow交换机执行。

Openflow控制器负责接收来自Sodero控制器的命令，分发到12台Arista交换机的Openflow接收器中。

## 管理平面

GUI：

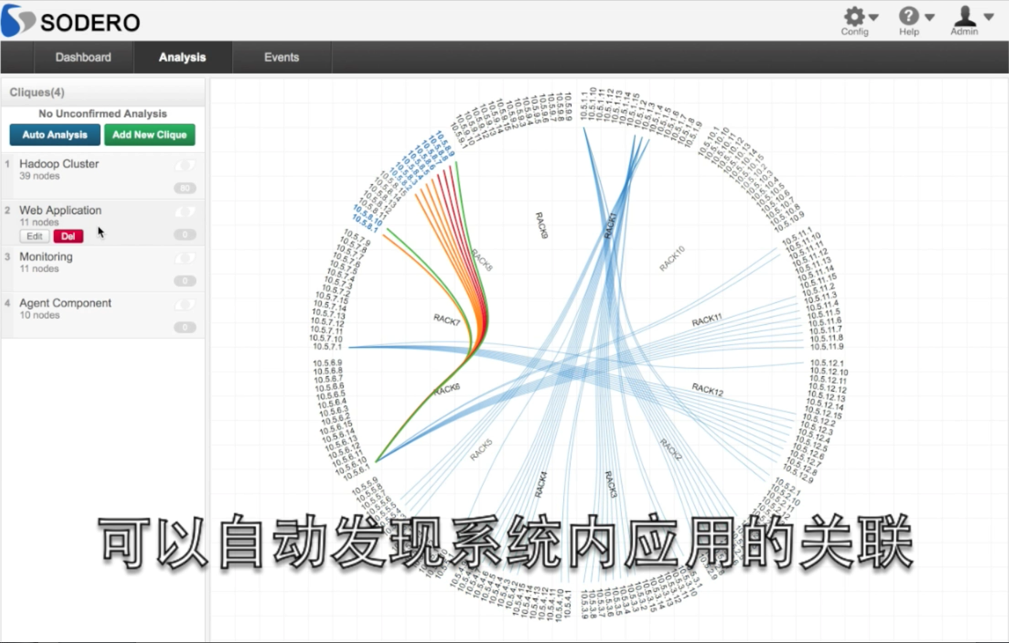
* 实时网络监测：

通过部署在服务器中的流量采集程序以及预先定义的网络拓扑结构，可以提供走势图与网络拓扑两种方式的监测能力，当网络拓扑中出现拥塞(某一条或多条边上的流量接近峰值)，系统可以第一时间发现并予以告警。

如果打开了Dynamic Equilibrium动态平衡选项，系统会自动调整拓扑中边的宽窄以适应流量要求。

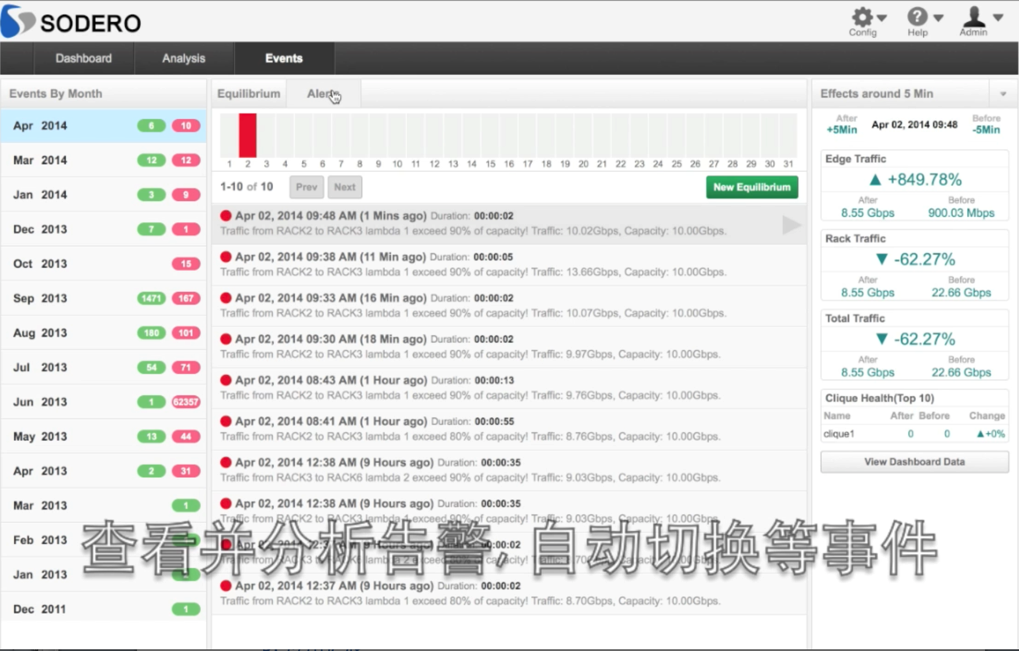
* 应用关联分析：



系统提供自动化的应用关联分析功能，通过分析历史的服务器－服务器流量及变化规律，可以找到相同时间序列规律的服务器组，自动或者手工干预定义出应用分组。

根据不同的应用业务特性，可以设定不同的网络流量优先级，在出现网络拥塞时，可以优先保证业务优先级高的应用获得更高的带宽保证。

* 事件记录分析：



系统会记录所有发生过的拥塞告警、自动平衡切换等事件，并提供自动分析功能，可以分析事件发生前后网络的吞吐量情况，网络优化等情况。

API

* 查询API：

查询预定义的配置信息如机架信息、节点信息、拓扑信息等，查询实时采集到的流量信息。

* 切换API：

目前支持两种拓扑切换API：

1）自动切换：根据系统采集到的或者接口传入的流量，自动实现拓扑切换和动态优化。

2）手动切换：根据接口传入的路由表信息，API函数将其转换为光交换可识别的命令进行动态切换。

详细的API列表及定义见《附录2》内容。

## 应用扩展

基于管理平面提供的API，可以在底层光交换网络基础上实现上层的应用扩展，比如预测Hadoop集群的网络需求或者OpenStack的虚拟机迁移流量，实现自动网络切换。

这部分内容在实际实现过程中还需要进一步设计和探讨。

# 二、软件部署

以下管理相关软件均部署在10.1.0.202服务器中

## Openflow Controller

* 部署路径

/root/sodero/floodlight-0.90

* 启动命令

/root/sodero/floodlight-0.90/startup.sh

* 停止命令

/root/sodero/floodlight-0.90/shutdown.sh

* 日志路径

/root/sodero/floodlight-0.90/floodlight.log

## Sodero Controller

* 部署路径

/root/sodero/controller/

* 启动命令

/root/sodero/controller/startup.sh

* 停止命令

/root/sodero/controller/shutdown.sh

* 日志路径

/root/sodero/controller/controller.log

## Sodero GUI

* 部署路径

/root/sodero/web/

* 启动命令

/root/sodero/web/startup.sh

* 停止命令

/root/sodero/web/shutdown.sh

* 日志路径

/root/sodero/web/web.log

* 访问地址

http://10.1.0.202:8888

## 节点软件

部署在各节点上的监控程序

* 部署路径

/root/sodero/agent

* 启动命令

/root/sodero/agent/run.sh

* 停止命令

/root/sodero/agent/stop.sh

* ARP设置命令

/root/sodero/arp\_install.sh

该命令用来设置光交换网络必要的静态ARP地址，即180台节点的ARP表。

## 维护脚本

* 部署路径

/root/sodero/ansiblescripts

* 运行命令

/root/sodero/ansiblescripts/sodero.sh

详情见下面章节。

# 三、维护脚本

## 批量维护脚本

系统使用Ansible作为批量运维工具，该工具使用Linux自带的Python脚本语言，支持ssh/scp及sudo操作，通过定义Inventory文件来存储待操作的对象(清华环境里即180台服务器节点)，通过编写Playbook脚本来实现具体的维护动作。工具详细介绍可参考其官方网站： http://www.ansibleworks.com。

Inventory文件是一个文本，结构类似于Windows下的Ini文件，用[]中括号区分不同的section，这里我们使用tsinghua\_host文件作为统一的Inventory文件， 具体内容说明如下：

* [localhost]

用于需要在本机执行的命令

* [agent:children]

是一个集合，包括Rack1到Rack12所有的节点。这是我们的批量操作默认的对象。也就是说默认执行批量操作是针对Rack1到Rack12的所有节点。

* [rack1] - [rack12]

从Rack1到Rack12对应的节点，方便单个Rack的动作执行。可以看到某些节点被注释掉了，因为这些节点无法访问或者有问题。

* [all\_servers]

所有的180台节点

* [notifiers]

告警事件接收人

其他的section不常用，这里不再敖述。

为了方便，我们提供了一系列的Playbook脚本以及一个统一的命令行脚本：sodero.sh来简化对大量Playbook脚本的调用，执行不带参数的sodero.sh可以查看脚本支持的动作种类。现将常用的命令说明如下：

* sodero.sh ping [nodes]

从管理服务器上Ping一个或多个节点

* sodero.sh cmd [nodes]

从管理服务器上远程执行命令

* sodero.sh cross\_ping

所有的180个节点互相ping，验证光网络所有节点的互通性

* sodero.sh arp\_install

所有的180个节点检查并设置静态ARP规则，ARP规则是保证节点间的联通的基础。

* sodero.sh agent\_run / agent\_stop

启动或停止流量监控程序

## 自动监控机制

目前提供了服务监控、网络监控及配置监控几类自动监控(Crontab + Ansible)，当监控出现异常时，会给Inventory文件中定义的告警接收人发送邮件通知：

* 服务监控

监控Openflow Controller、Sodero Controller、GUI等服务状态。

* 网络监控

监控180台服务器是否可以ping通，光网络环境互ping是否联通，12台OSU和12台Arista交换机是否可以联通等。

* 配置监控

监控服务器上的静态ARP是否配置正常等。

# 附录1、FAQ

## 光网络节点互相Ping不通

最有可能的原因是相关机器上缺少静态ARP设置，之前发现过ARP协议设置完成后又丢失的情况。可通过登录机器执行arp –n查看是否有10.5.x.x网段的静态ARP信息，Flags Mask为CM。

其次是光交换机的联通性问题，需要登录http://10.1.0.202:8888/osuadmin/查看OSU状态是否有非正常的。

最后可能是Openflow规则的设置问题，这种问题多数发生在controller刚启动或者重启的情况。可以通过查看controller日志定位。

目前新加入的监控机制可以自动监控此项配置并产生告警。

## GUI无法访问

GUI服务未启动，或者端口被防火墙屏蔽，可通过查看服务状态以及IPTables规则定位。

目前新加入的监控机制可以自动监控此问题并产生告警。

## API无法访问

目前API与GUI服务集成在一起的，因此定位方法同上。但切换执行类的API还要基于Sodero controller服务，因此还要检查controller服务是否运行正常。

目前新加入的监控机制可以自动监控此问题并产生告警。

# 附录2、APIs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URI** | **Method** | **Description** | **Input(list)** | **Output(list)** |
| /api/racks | GET | List all existing racks | None | {  "id":"<int>",   "name":"<string>" } e.g., {"id":1, "name":"rack1"} |
| /api/nodes | GET | List all existing hosts | None | {  "id":"<int>",   "name":"<string>",  "rack":"<int>",   "ip":"<IP>",   "mac":"<MAC>" }  e.g., {"id":1, "name":"host1", "rack":1, "ip":"10.5.1.1", "mac":"F4:52:14:0A:8D:F0"} |
| /api/traffic | GET | List all existing flows traffic between IP pairs | None | {  "src-ip":"<IP>",   "dst-ip":"<IP>",   "traffic":"<long long (bits/s)>" } e.g., {"src-ip":"10.5.1.1", "dst-ip":"10.5.2.1", "traffic":11786500} |
| /api/topology | GET | List all connections between racks, their bandwidth capacities and realtime traffic | None | {  "src-rack":"<int>",   "dst-rack":"<int>",   "capacity":"<long long(bits)>",   "traffic":"<long long(bits/s)>" } e.g., {"src-rack":1, "dst-rack":2, "capacity":20000000000, "traffic":1125768} |
| /api/equilibrium\_auto | POST | Execute dynamic equilbrium (change routes and bandwidths) according to input traffic or realtime monitoring traffic demand. Once input flow traffic exists, the corresponding realtime monitoring traffic will be replaced. | {  "src-ip":"<IP>",   "dst-ip":"<IP>",   "traffic":"<long long (bits/s)>" } e.g., {"src-ip":"10.5.1.1", "dst-ip":"10.5.2.1", "traffic":11786500} | {  "success":"<true or false>",   "message":"<string>" } e.g., {"success":"true", "message":"10 routes changed"} |
| /api/equilibrium\_manu | POST | Statically set routes according to input route list. Dynamically set bandwidth with a none zero capacity or remains old bandwidth with zero capacity. | {  "src-ip":"<IP>",   "dst-ip":"<IP>",   "routes":[<int>, <int>...]   "capacity":"<long long(bits)>" } e.g., {"src-ip":"10.5.1.1", "dst-ip":"10.5.3.1", "routes":[1,2,3], "capacity":"2000000000"} | {  "success":"<true or false>",   "message":"<string>" } e.g., {"success":"true", "message":"10 routes changed"} |