salt-broker: 轻量级的Salt Proxy解决方案

基本简介

当前运维系统底层采用 Salt 进行实现, 由于节点分布在全国各地, 存在南北通畅问题, 为了解决这个问题, 之前采用了 Syndic 方案, 在实际使用中发现由于Syndic采用分治机制, 弱化了MasterOfMaster, 在某些网络状况较差的情况下, 会让结果变得不可控. 为了解决该问题, 借鉴ZeroMQ文档, 开发了轻量的Salt Proxy解决方案 salt-broker

前置阅读

- 0MQ - The Guide: Sockets and Patterns
- Salt中ZeroMQ那点事
- Salt中Syndic那点事

环境说明

- CentOS6.4
- Salt 2014.1.10 ,默认配置
- 由于本文为原理解析, 所以采用的代码为最初版的代码, 只描述了其功能逻辑

为什么会有salt-broker?

因为采用Syndic, 在网络链路不好的情况下, syndic架构将变得不可控. 详情请访问 Salt中Syndic那点事

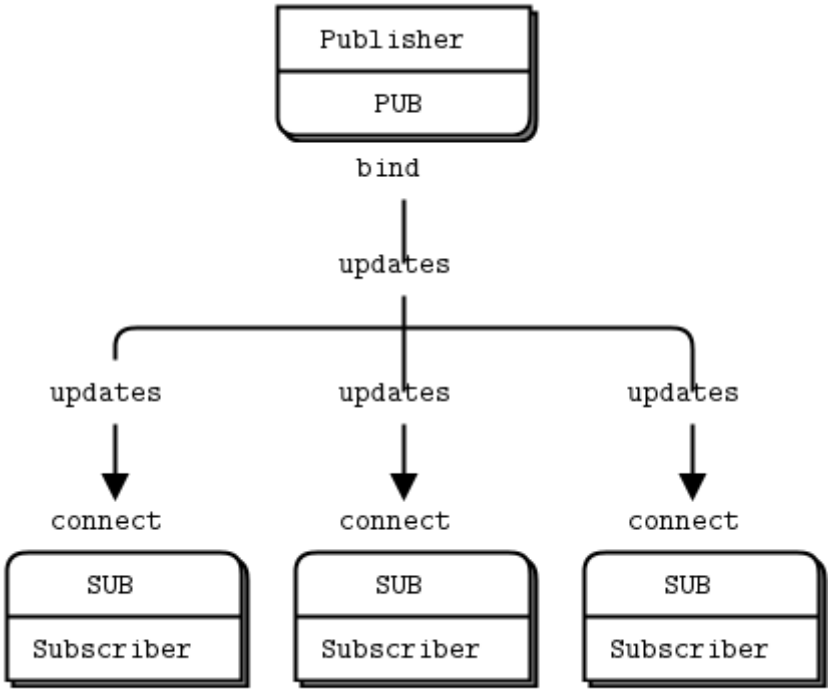
而由于业务系统采用Salt作为中心调度, 结果不可控将变得非常糟糕. 尝试在syndic基础上进行修改, 没有达到预期. 分析了需求, 其实自己需要的是一个强中心, 轻量级的Salt Proxy解决方案, 所以就有了 salt-broker .

salt-broker是什么?

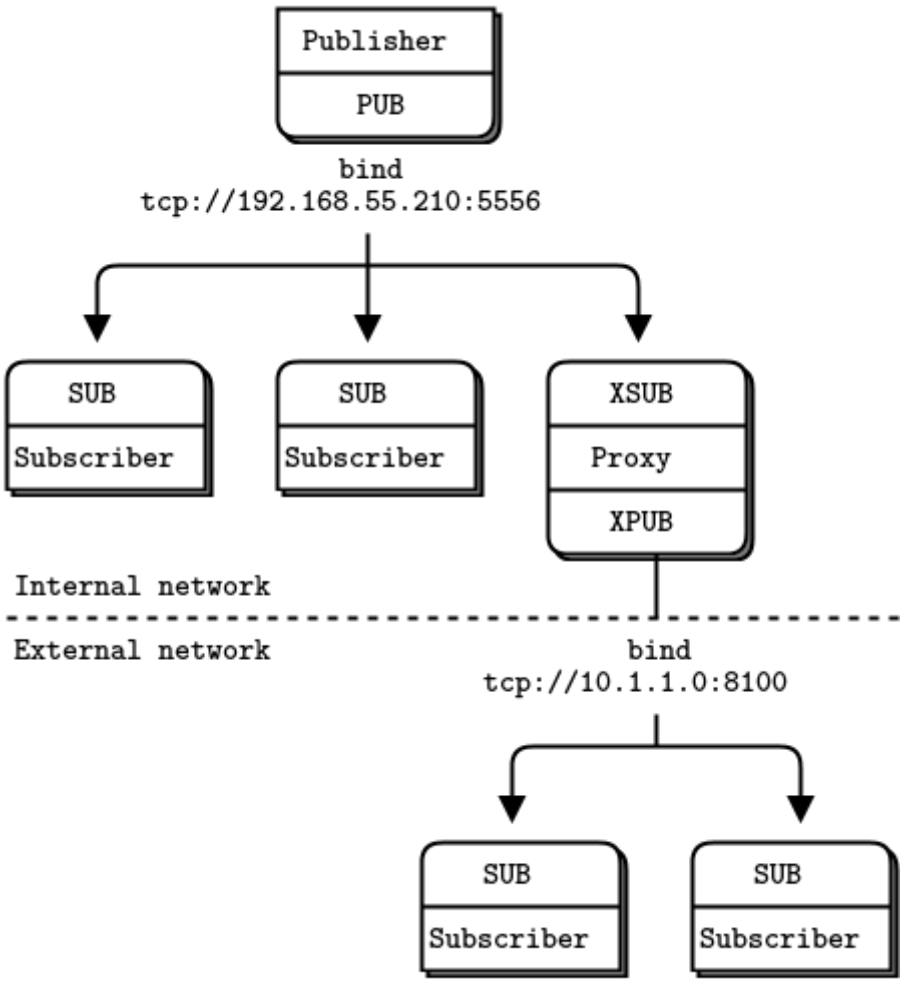
salt-broker是轻量级的Salt proxy解决方案, 只做数据转发, 不做额外的处理. 其工作原理如下:

PUB Broker

在Master/Minions结构中, 命令分发采用ZeroMQ PUB/SUB模式, 如下图:



salt-broker中的PUB Broker在中间增加了Forwarder Proxy层, 使架构变成如下:



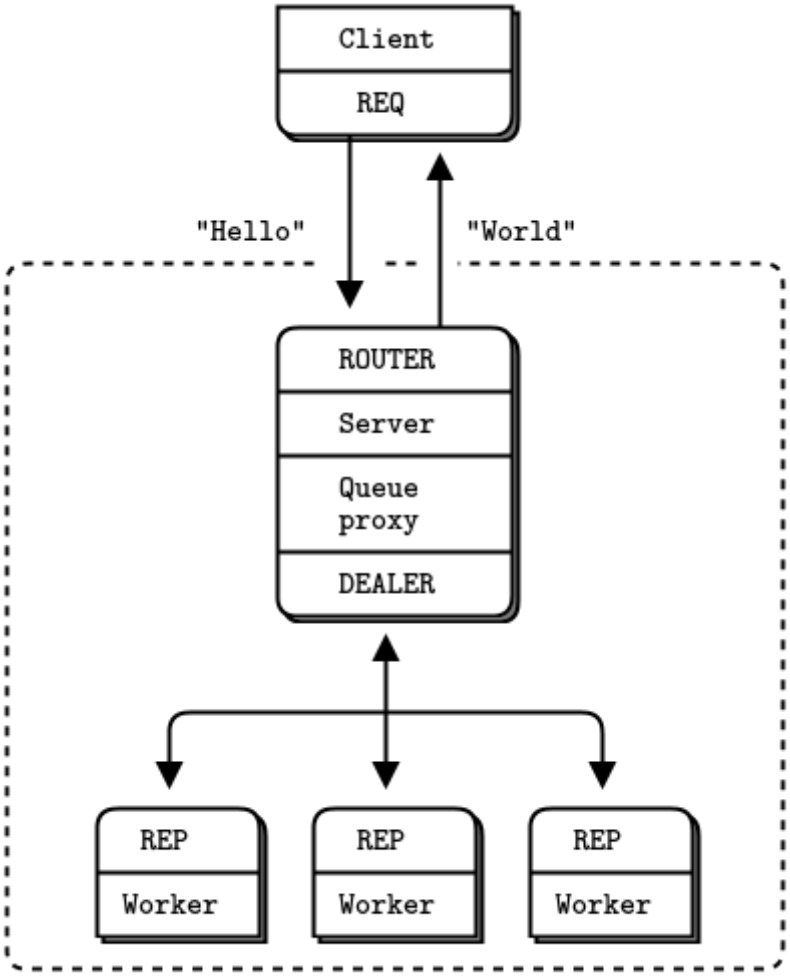
对应代码如下:

```
master_pub = "tcp://%s:%s" %(self.master_ip, self.pub_port)
context = zmq.Context()
frontend = context.socket(zmq.SUB)
frontend.connect(master_pub)
backend = context.socket(zmq.PUB)
backend.bind("tcp://0.0.0.0:%s" %self.pub_port)
frontend.setsockopt(zmq.SUBSCRIBE, b'')
while True:
    message = frontend.recv_multipart()
    backend.send_multipart(message)
```

本地建立PUB接口, 并连接Master的SUB接口, 订阅来自于Master的消息, 接收到后立马发送到本地的PUB接口. 由于Minions上指定的Master地址为salt-broker所在的地址, 所以Minions能够接受到该消息.

Ret Broker

在Master/Minions结构中, 认证,文件服务,结果收集等采用ZeroMQ ROUTER/REQ模式, 如下图:



salt-broker中的Ret Broker在原来的REQ/ROUTER之间再增加了一层ROUTER/DEALER Proxy层. 对应的代码如下:

```
master_ret = "tcp://%s:%s" %(self.master_ip, self.ret_port)
context = zmq.Context()
frontend = context.socket(zmq.ROUTER)
frontend.bind("tcp://0.0.0.0:%s" %self.ret_port)
backend = context.socket(zmq.DEALER)
backend.connect(master_ret)
zmq.device(zmq.QUEUE, frontend, backend)
```

本地建立ROUTER接口, 接收来自于Minions的REQ请求; 同时本地建立DEALER接口, 连接Master的Router接口, 将接收到的数据发送给远端

的Master ROUTER接口.

Broker VS Syndic

salt-broker与syndic一样, 都支持多层级架构. salt-broker相对于syndic, 更为轻量级, 只做数据转发. 在超大规模场景下, salt-broker并不能有效的降低master的压力, 而syndic能够降低.
syndic本地会维护auth及文件服务系统, 而broker会将所有请求转发给Master, 即所有的Minions的最终管理都是在Master上. 由于所有管理权均在Master上, salt-broker能够解决掉之前Syndic在网络不稳定时的不可控问题.

如何使用salt-broker?

全新安装

1. 安装salt(需提前配置EPEL)

```
yum -y install salt
```

1. 安装salt-broker

```
pip install salt-broker
wget https://raw.githubusercontent.com/pengyao/salt-broker/master/pkg/rpm/salt-broker -O /etc/rc.d/init.d/salt-broker
chmod +x /etc/rc.d/init.d/salt-broker
```

1. 配置salt-broker

/etc/salt/broker

```
master: master_ip
```

1. 启动salt-broker

```
service salt-broker start
chkconfig salt-broker on
```

1. 启动完毕后, 需要将minions配置文件中的master配置为salt-broker所在的主机, 同时重启minions服务
2. 在Master接收minions的key(如果之前已accept, 无需操作本步骤)

在Syndic主机上安装

1. 关于Syndic主机上的syndic及master服务

```
service salt-syndic stop
service salt-master stop
chkconfig salt-syndic off
chkconfig salt-master off
```

1. 安装salt-broker

```
pip install salt-broker
wget https://raw.githubusercontent.com/pengyao/salt-broker/master/pkg/rpm/salt-broker -O /etc/rc.d/init.d/salt-broker
chmod +x /etc/rc.d/init.d/salt-broker
```

1. 配置salt-broker

/etc/salt/broker

```
master: master_ip
```

1. 启动salt-broker

```
service salt-broker start
chkconfig salt-broker
```

1. 启动完毕后, 需要将原syndic下的minions配置文件中的master配置为salt-broker所在的主机, 并

将/etc/salt/pki/minion/minion_master.pub删掉, 然后重启minion服务

2. 在Master接收minions的key(如果之前已accept, 无需操作本步骤)

其他注意事项

默认配置中, 使用的是pub(4505)及ret(4506)端口, 如果master端口并非该端口, 需要在/etc/salt/broker配置文件中增加:

```
ret_port: 'new_ret_port'
publish_port: 'new_publish_port'
```

更改后并重启salt-broker服务.

Posted on: 2014-09-07