HSVS 内网版部署指南

定义

> hcap: hsvs HTTP流还原模块，使用Docker方式安装部署
>
> hsvs-mbridge: hsvs 预处理模块，使用Dockr方式安装部署

> ElasticSearch: 开源文档存储引擎https://www.elastic.co
>
> kafka: 开源分布式消息队列 http://kafka.apache.org
>
> redis: 开源内存k/v数据库 https://redis.io/
>
> nsq: golang开发的开源分布式消息队列 https://nsq.io

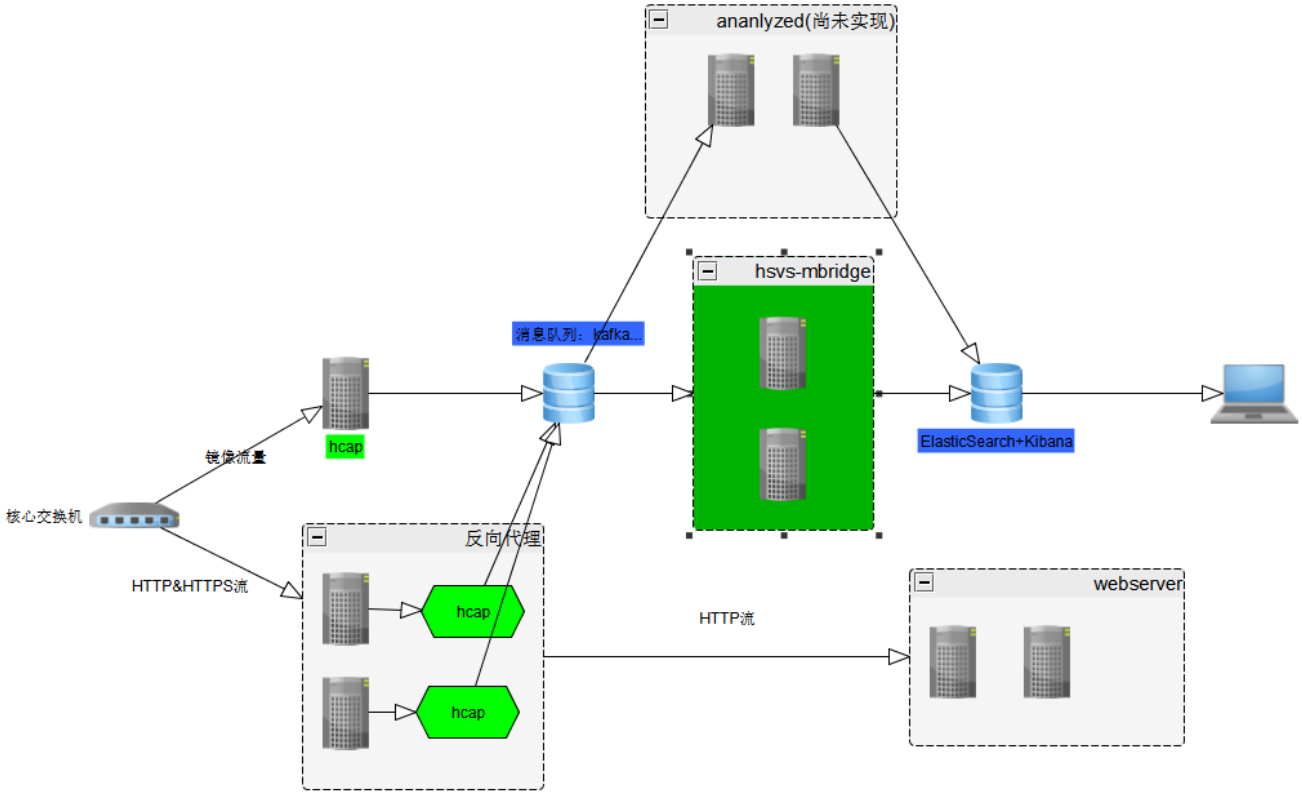
部署方案

为了方便适配各种规模企业的使用场景，HSVS支持三种消息中间件redis/nsq/kafka。

kafka方案

优点: kafka吞吐率，稳定性，持久化很优秀，经过长期实践检验，kafka可以多group消费者同时消费一topic数据，方便自定义开发分析程序

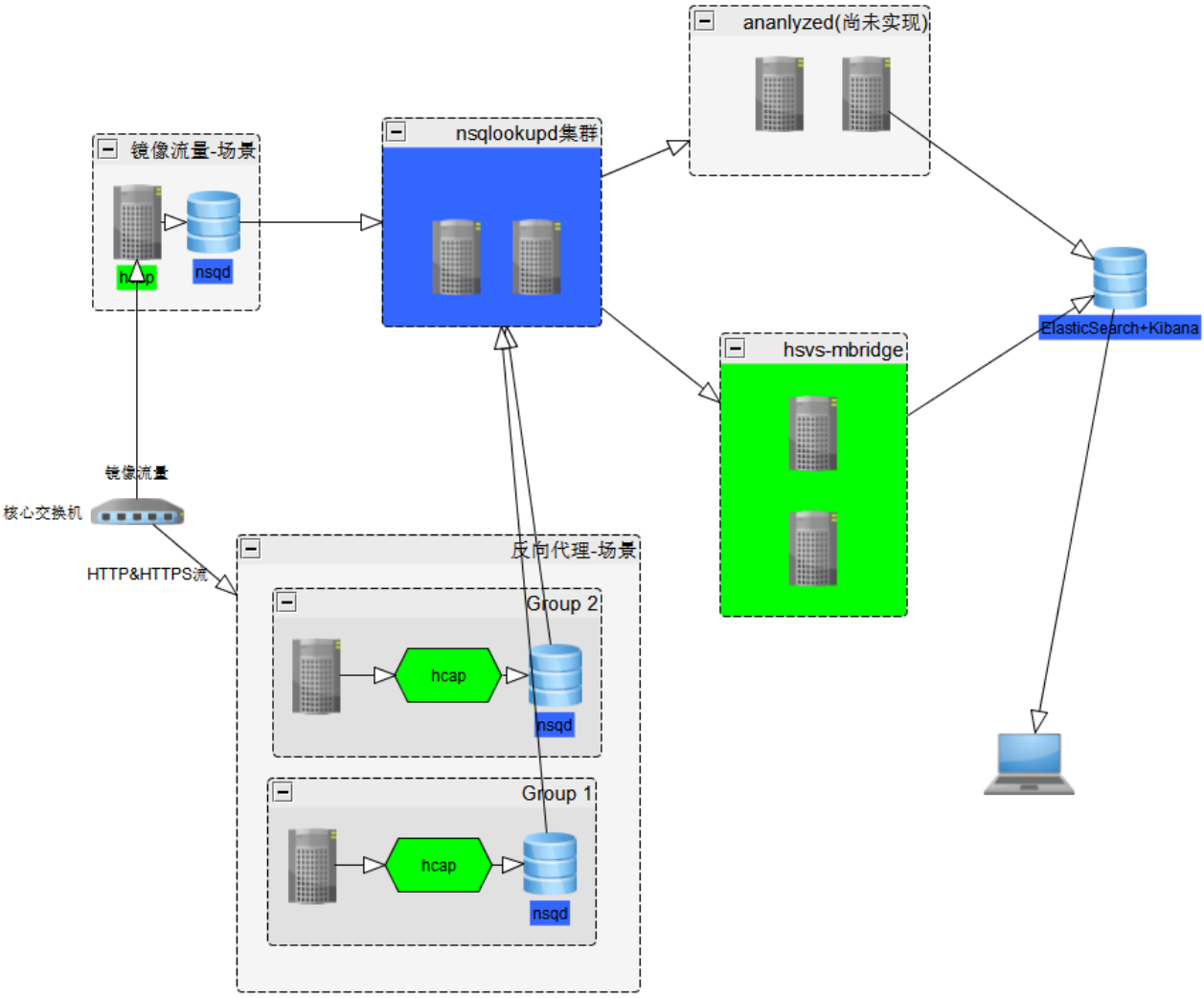
缺点: kafka部署较为麻烦，kafka要配合zookeeper来使用； 优先选择企业内部已经搭建好的kafka，最新版的kafka已经集成了zookeeper，如果需要自己搭建请参考http://kafka.apache.org/quickstart



nsq方案

优点: nsq是一个golang开发的消息中间件，部署简单，使用也简单，支持持久化，支持多group消费者消费同一topic数据。

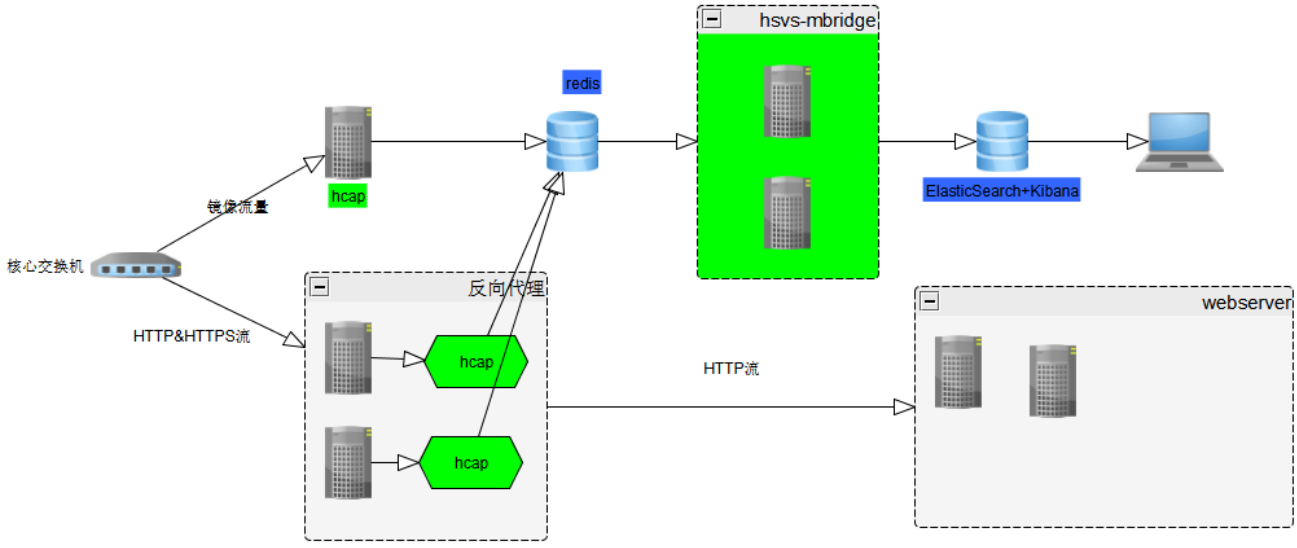
缺点: nsq设计理念是本地agent，所以如果在镜像流量的节点nsq消息两会大幅大于其他端点，没有实现想要的负载均衡。



redis方案

优点: 部署简单，可以使用企业已有redis(目前不支持redis集群)

缺点: 目前hcap不支持写入集群，redis不支持多group订阅消费，无法自定义分析程序，且不支持持久化。



抓取过滤器规则

hcap支持按照域名和资源来进行过滤，只保留自己所需要的内容

```
[FILTER]
include_domains = *.baidu.com
exclude_domains = ad.baidu.com
drop_extentions = png,jpg,bmp,css,js,gif,ico,woff,woff2,ttf,eot
drop_urls =
drop_resp_types =
drop_req_types = multipart/form-data

说明
* include_domain: 只有匹配到这里的规则的数据才会被保留
* exclude_domain: 在include_domain之后，会排除掉匹配到这条规则的字段
* drop_extenstions: 如果扩展名是这些，那么响应体(response.body)会被drop掉，其他记录任然会保留
* drop_urls: 丢弃指定URL的记录，只保留记录
* drop_resp_types: 根据响应的content-type来丢弃响应体(response.body)
* drop_req_types: 根据请求体来丢弃响应体(response.body)
```

使用**kafka**时 分区数**hsvs-mbridge**对应关系

kafka对生产者的个数没有限制，HSVS使用轮训随机写入多个分区 kafka对消费者的数量有限制，应当能够被分区书目整除，负责会出现有的分区时钟无法被消费的情况。 kafka分配分区时建议按照2的整数倍进行分配，建议从4开始，如果不够，那么分配8、16，消费者hsvs-mbridge 建议从2开始，如果不够加到4.通常2Gbps下，4个hsvs-mbridge足够应对。

怎么查看处理性能是否足够，是否需要扩容。 系统搭建完毕后在kibana中可以看到没跳抓取的内容。 没条记录里均含有三个时间raw*time*和*@timestamp*，*raw*time表示网卡抓取到数据的时刻，@timestamp表示hsvs-mbridge的处理时间。如果相隔距离越来越大，表明消费者处理能力不够，需要增加。 另外查看每一个hsvs-mbridge的CPU占用率也是重要的参考。

判断kafka分区数目是否需要扩容，通过经验值+带宽估算，kafka主要的性能瓶颈是带宽。 千兆网络环境下，kafka broker性能等于带宽乘以broker所在物理机数量。 通常设计时，分区数量的性能应当在最大流量的两倍。