单机&分布式&可伸缩容

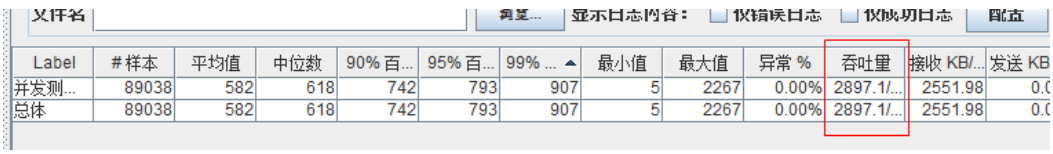
一、 单机版部署

1. 部署架构

数据库和应用程序在同一个物理机，数据库和应用程序将会存在资源争用情况。

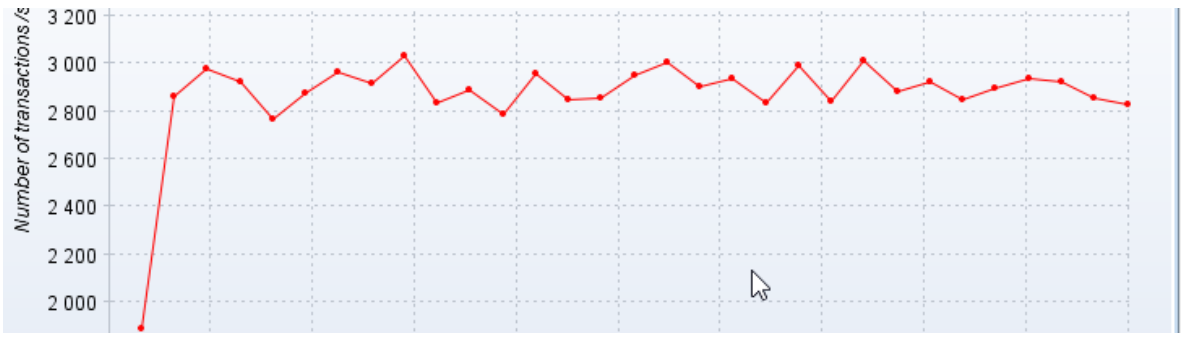
2. 压测情况

聚合报告： 平均响应时间，吞吐量（每秒的请求数量）

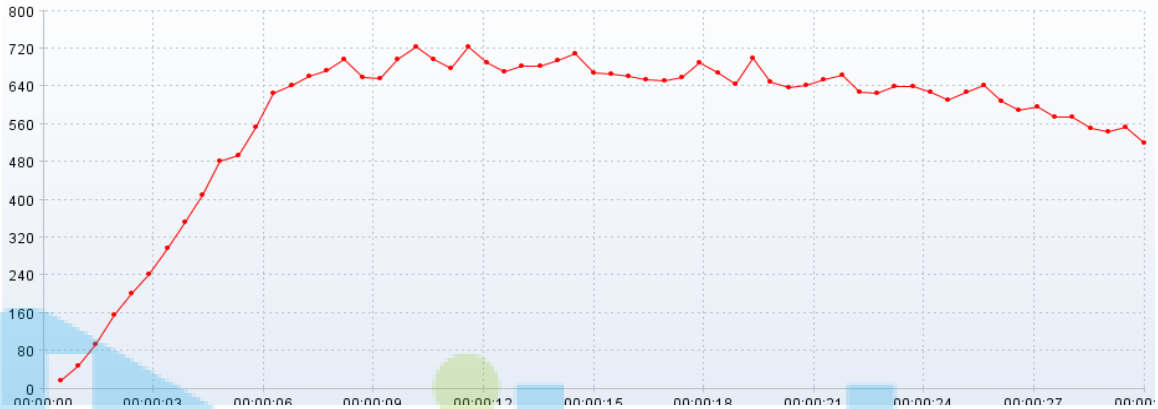


Label	# 样本	平均值	中位数	90% 百...	95% 百...	99% ...	最小值	最大值	异常 %	吞吐里	接收 KB/...	发送 KB
并发测...	89038	582	618	742	793	907	5	2267	0.00%	2897.1/...	2551.98	0.0
总体	89038	582	618	742	793	907	5	2267	0.00%	2897.1/...	2551.98	0.0

TPS 曲线图：



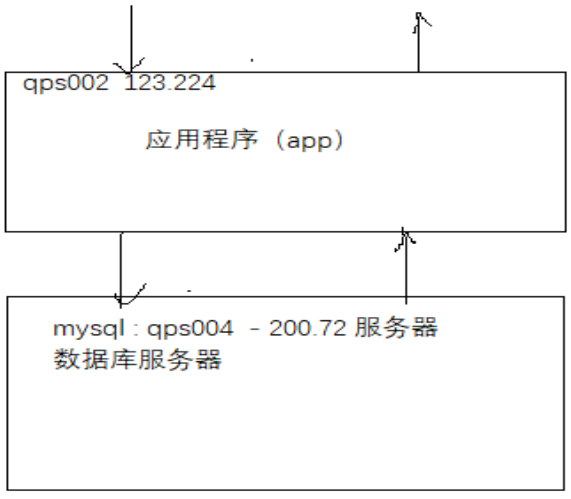
RT 曲线图：



极端苛刻要求： 每一个接口都是几十 ms, 100ms 以内

二、 分离模式

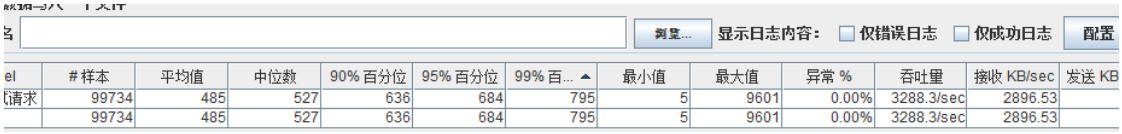
1. 分离部署架构



服务器通信使用内网通信，因此网络的影响可以忽略不计。（局域网内进行通信）

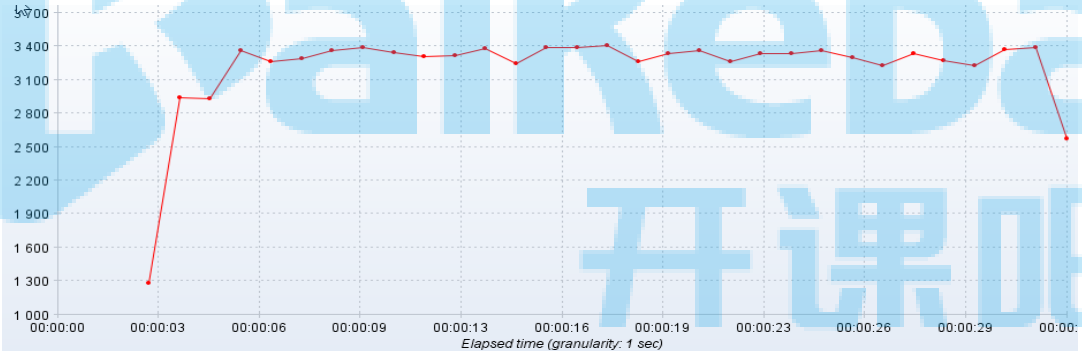
2. 压测情况

聚合报告： 提高 400QPS

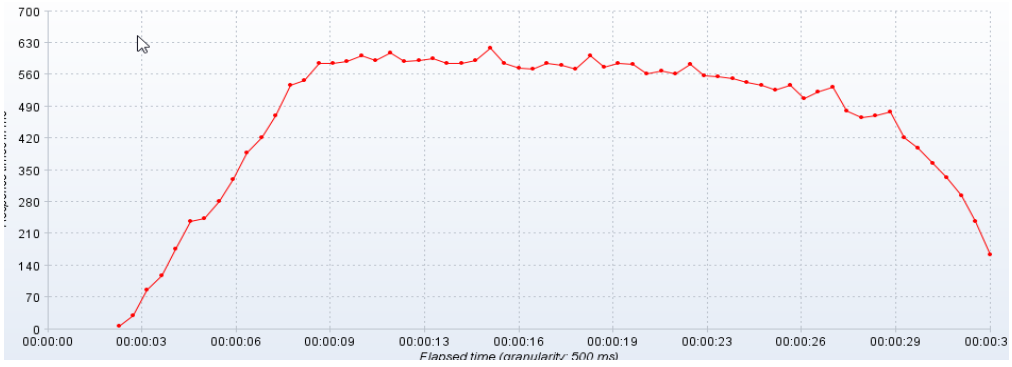


el	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百...	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB
请求	99734	485	527	636	684	795	5	9601	0.00%	3288.3/sec	2896.53	
	99734	485	527	636	684	795	5	9601	0.00%	3288.3/sec	2896.53	

TPS 曲线图：

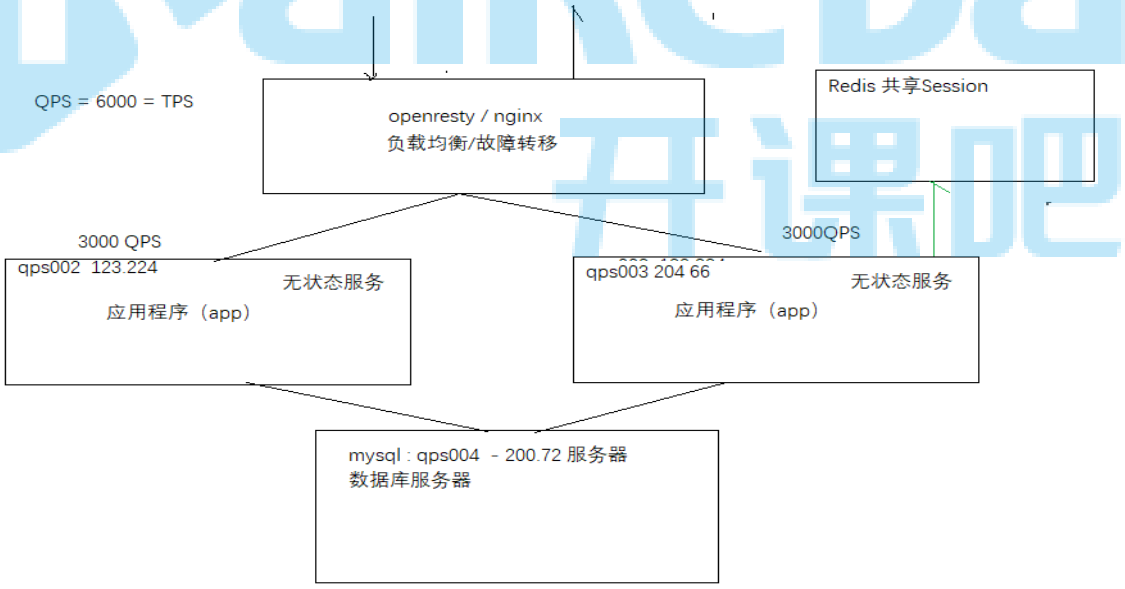


RT:



三、 分布式部署

1. 部署架构



流量估算：

单机：3288

2 台机器： 3288 * 2 = 6500 +

2. Openresty

下载 http://openresty.org/cn/download.html

.configure

make && make install

默认安装：/usr/local/openresty

配置：

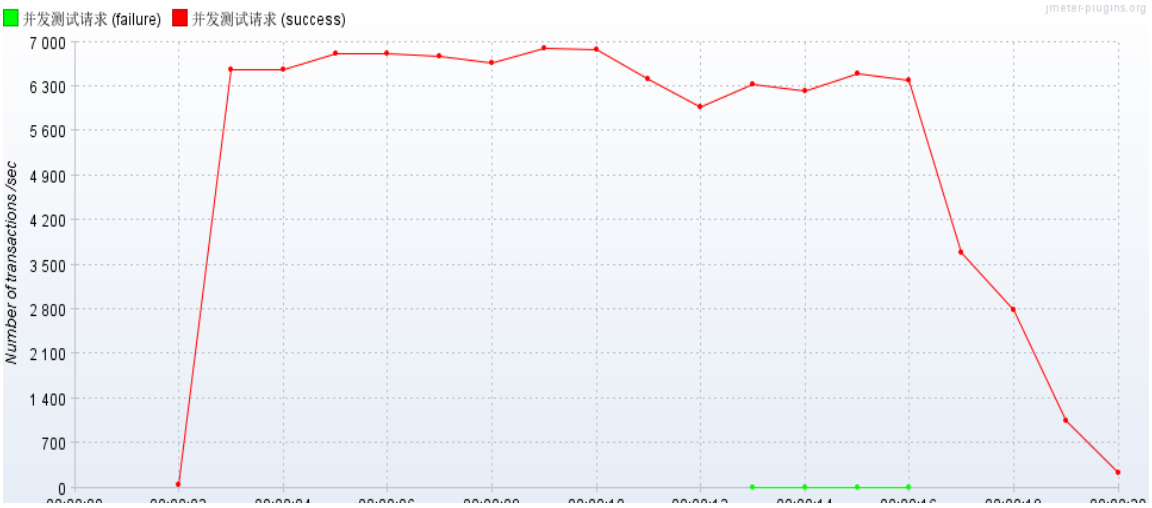
```
upstream BACKEND {
    server 10.0.3.176:8888 weight=1 max_fails=2 fail_timeout=30s;
    server 10.0.3.179:8888 weight=1 max_fails=2 fail_timeout=30s;
}

server {
    listen       80;
    server_name  qps001;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        #root   /usr/local/src/nginx-svc/static_vipshop;
        #root   html;
        #index  index.html index.htm;
        proxy_pass http://BACKEND;
    }
```

3. 压测情况

聚合报告：

Label	#样本	平均值	中位数	90% 百分位	95% 百分位	99% 百...	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/
并发测试请求	100000	159	98	282	386	1487	4	7031	0.02%	5624.6/sec	5352.51	
总体	100000	159	98	282	386	1487	4	7031	0.02%	5624.6/sec	5352.51	

TPS 曲线图：



四、 K8s 可伸缩容

1. 云原生

阿里巴巴： 去年项目全部上云 （下单：50w+/s）

云：

海量的服务器所组成一个环境，而这个环境可以看成一个整体，就是一台计算机（cpu:和 ，内存：和）

问题： 难以管理

Openstack

– 构建企业级私有云 （虚拟机）

容器（docker 容器）

- Docker 容器中部署很多服务，海量服务需要部署，意味着有海量的容器（google：2 亿容器/day）

问题： 难以管理

Kubernetes

云原生：

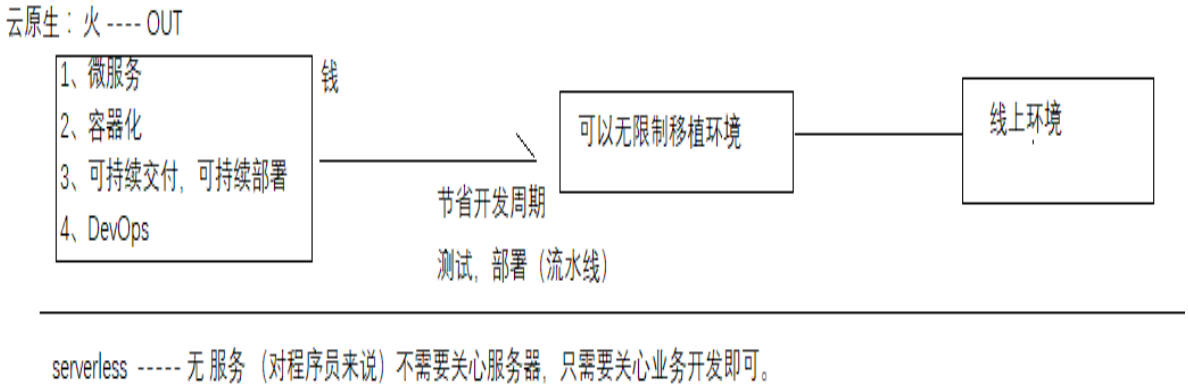
1、容器化 （所有的项目都必须跑在容器中）

2、微服务

3、DevOps

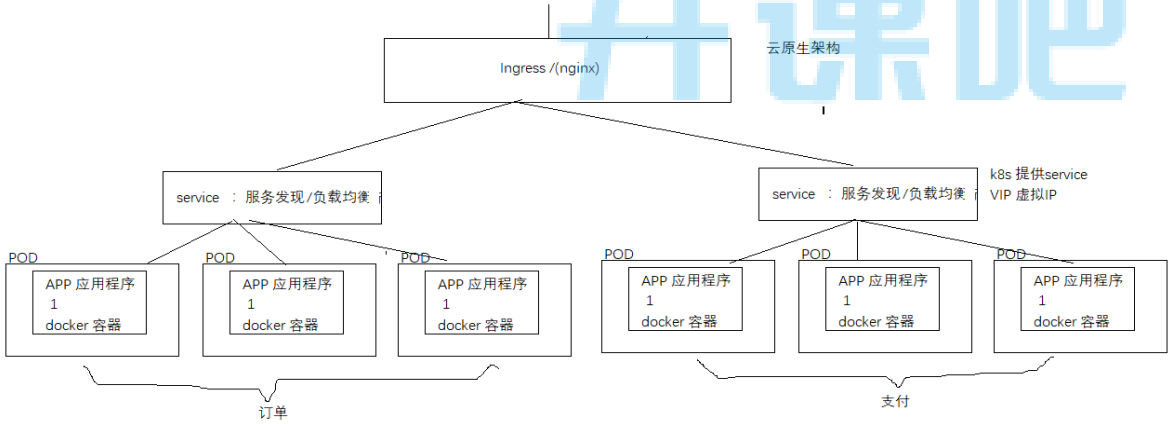
4、ci/cd

CNCF : service mesh 服务网格架构加入云原生



serverless ----- 无服务（对程序员来说）不需要关心服务器，只需要关心业务开发即可。

2. 部署架构



POD 服务如果发现 cpu，内存利用率过高，立马进行 pod 扩容，以方便应对大流量的来袭。因此还需要部署 HPA(自动触发扩容，缩容)

3. 部署

1、deployment 部署资源对象

2、service 资源对象

3、Ingress 资源对象

4、HPA 资源对象

4. 扩容实现

kaikeba
开课吧