1、如何对一个服务进行性能优化

准备阶段：了解应用的概况：总体架构、服务器信息等；通过性能测试，对性能瓶颈的进行粗略评估，明确优化目标；

分析阶段：通过各种工具或手段，初步定位性能瓶颈点；

调优阶段：根据定位到的瓶颈点，进行应用性能调优；

测试阶段：让调优过的应用进行性能测试，与准备阶段的各项指标进行对比，观测其是否符合预期，如果瓶颈点没有消除或者性能指标不符合预期，则重复步骤2和3。

2、如何进行一个完整的压测：

1.梳理高峰业务场景：可以按照一些问题描述模板比如5W2H来梳理业务场景，例如：

场景1：2018年11月11日00:00~01:00，共计xx万个用户同时操作xx。

场景2：每天xx时间段，xx系统会轮询调用xx服务xxxx次。

2.明确压测目标，根据业务场景评估或者历史流量峰值评估，并且制定目标时需要留一些Buffer。

3.准备压测数据&链路改造&检查压测环境（写库问题）

1、直接DB建数据 2、通过业务接口建数据 3、线上流量COPY

4.接口单元测试：需要验证压测链路是否通，如果有影子/隔离环境，看流量是否打到正确的环境中去。

5.压测前通知到相关运营、业务、开发同学，进行小流量试压

6.正式压测：

1.逐步加压到目标QPS，记录系统指标：CPU使用率、内存使用率、Load等，服务指标：QPS、RT、

2.在加压过程中，观测并记录发现的问题，如果出现阻塞性问题，停止压测。

7.压测后发出压测报告，压测报告对于目标和结论尽量一句话表述出来。

3、压测过程中遇到的一些问题

一、 压测环境压测服务时出现链接数过多

1、时间：2020/6/11

2、请求链路：施压机-- 负载均衡--nginx--服务器。

机器：4c8g

3、遇到问题：

1、从施压机给服务器每秒发送50个请求，使用netstat -a ，施压机本地链接未超过200个，服务器链接快速增加，服务器链接跟负载均衡10.4.11.160的链接在6000+。

4、定位方法：

1、修改负载均衡的/etc/sysctl.conf 文件，将net.ipv4.tcp\_fin\_timeout 由60变成10，具体操作： 在/etc/sysctl.conf 中添加 tcp\_fin\_timeout = 10， 并执行sysctl -p 生效。再次压测，情况无大变化。

2、在施压机通过绑定hosts直达机器，情况无大变化。

3、修改服务所在机器的/etc/sysctl.conf， 修改此三项配置如果，服务器链接正常。但此配置与线上配置不一致。

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_fin\_timeout = 10

4、三个参数的解释，一般不建议开启：https://www.cnblogs.com/lulu/p/4149312.html

net.ipv4.tcp\_tw\_reuse = 0 表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认为0，表示关闭

net.ipv4.tcp\_tw\_recycle = 0 表示开启TCP连接中TIME-WAIT sockets的快速回收，默认为0，表示关闭

net.ipv4.tcp\_fin\_timeout = 60 表示如果套接字由本端要求关闭，这个参数决定了它保持在FIN-WAIT-2状态的时间

二、不同机器之间性能差距较大，性能差机器可能会拖慢整体服务性能。

1、时间：2020/5/17

2、请求链路：施压机-- 外网服务。（绑到hosts到某台外网负载均衡）

3、遇到问题：

1、扩容后进行复测，服务性能没有上升。

2、对不同型号机器进行绑定hosts到指定机器，服务性能差距比较大。

4、解决方案：

1、服务均衡对机器分流进行调整；

2、下掉性能较差机器（采用此方法）

3、在服务部署在不同型号的机器上时，建议先进行框架压测。

4、参考数据：裸金属 200左右 ；4c8g的云主机 500 ；10c32g云主机 1800（php7）

三、压测过程中，出现压测图形周期性出现断崖式下跌现象。

1、时间：2019/5/18

2、施压机-- 外网服务。（绑到hosts到某台外网负载均衡）

3、遇到问题：

在压测过程中观察压测流量曲线 发现在一瞬间请求出现断崖式下跌 然后迅速爬升 出现一个明显的抖动点。

4、定位原因：

服务机器资源正常，观察下流调用，发现mysql数据库连接数被打满，导致无法新建连接。 分析导致连接未被释放原因：发现sql没走到索引上，所以部分查询过慢，过慢链接累加导致连接数被打满。

5、修改mysql查询语句。

四、gatling ssl握手超时问题

1、时间：2019/5/18

2、施压机-- 外网服务。（绑到hosts到某台外网负载均衡）

3、遇到问题：

压测https服务过程中，gatling出现大量ssl握手超时报错。

4、定位原因：

gatling的rps每秒都新建socket连接，因为https是双向认证的，nginx会去解析ssl证书和密文 ，当nginx内存不够大的时候 连接数增加导致，导致nginx划分的32m解密内存迅速被打满，后续解密握手在等待最后出现握手超时。

4、如何搭建一个压测平台

背景：小猪近一年来业务量大幅上涨，多次发生因流量过大导致的服务不可用问题，故服务性能优化迫在眉睫。而小猪还没有性能测试相关的平台。大量的压测需要手工进行，存在重复劳动，效率低下的问题。在此背景下，我们设计并开发了业务压测平台。

设计方案：因为是从0到1进行设计开发，我们参考学习了公司一些现有的平台框架，并结合自身的代码情况，最终选定了压测平台前端使用vuejs+element，后端使用微服务框架spring cloud、spring boot、网关zuul等框架进行压测任务的接收、下发和数据收集等。由于开源的gatling方便模块化，搭建环境和编码都相对简单，所以压测工具选择了gatling，数据库使用redis、mysql进行数据存储。

实现过程：在我们实施设计方案的过程中，也遇到了一些与预期不符合的情况，比如说：设计时计划在压测平台实时展示压测过程数据，在编码过程中发现会发生因数据量过大导致web页面卡死的问题。通过对gatling工具的学习了解，发现gatling封装了接口往时序数据库influxdb中存储数据，grafana读取influxdb展示数据也是目前比较成熟的方案。所以我们调整了设计方案，嵌套grafana进行压测过程数据展示的方案。在开发过程中，也遇到了排期的问题，所以我在完成前端的基础上，也开发了一些后端功能。

目前已实现压测数据自动产生、压测过程监控和报警、压测报告自动生成功能。正在开发压测环境自动化检测功能。

收益：压测系统上线后，针对业务系统做了大量压测，摸清了系统的性能状况。业务同学据此优化核心接口，性能平均提高了5倍，并经受了线上大促活动的检验，

长期规划：通过更多的内置场景例如：线上首页压测，小程序首页压测等，结合公司现有的工具平台，实现一键式自动化压测平台。