性能测试面试题

一、性能测试开展过程：

答：第一步：找产品沟通哪些接口需要压测，需要达到什么样的预期值(TPS 和响应时间)
第二步：编写测试计划，人员、时间周期、工具
第三步：环境搭建
第四步：造数据
第五步：场景测试（单接口基准测试、单接口压力测试、混合接口测试、稳定性测试）
第六步：结果分析，提交测试报告
第七步：等待开发性能调优，复测

二、交付一个性能测试项目，请阐述你的性能测试流程 (偏高级的回答)

答：1：明确测试需求
2：基于需求设计测试用例，测试方案，测试计划
3：准备测试数据，测试账号（预估并发量），设计测试脚本（参数化，表达式，断言，控制器）
4：运行测试脚本，数据监听（响应时间，tps，活动线程），结果分析（判断性能瓶颈）
5：基本性能瓶颈做调优（tomcat 线程池，jvm 内存，swap 内存，带宽）
6：调优之后做性能回归，和前期结果做对比，是否有明显的优化。
7：代码问题优化（自己定位或者交给开发定位）
8：性能测试报告。整理性能测试数据（包括调优之前和调优之后）
9：构建持久化的性能监听平台，监听线上的服务性能
性能测试贯穿项目始终，从需求分析到上线之后，都需要持续跟踪分析发现问题，响应解决问题

三、什么是性能测试？

答：测试系统有没有性能问题
考虑时间，空间
服务端资源是否足够？
响应时间是否超时？
系统是否足够稳定？

四、性能测试的核心原则是什么？

答：基于协议，多线程，场景设计
协议：所有的请求都是基于协议发出去 http，https，udp，tcp，mqtt
多线程：压力测试是基于 java 多线程原理，通过线程去模拟用户的行为
基于场景：控制器+定时器设计各种场景满足压测要求
并发场景、负载场景、稳定性、压力测试
… ….

五、什么是负载？有哪些负载模式？

答：负载就是对服务器迭代式加压，从而寻求性能测试拐点
1：用户模式。不断增加的用户数带来的压力
1 个用户 1s 内发起 20 个请求，rps=20/s
2：请求模式，不断增加的请求数带来的压力
10 个用户，1s 用户 1s 内发起 1 个请求，rps=10/s
不能单纯的通用用户去衡量压力，直接通过每秒请求数去衡量压力。直接从服务端考虑

六、性能测试的应用领域有哪些？

答：1、能力验证：乙方向甲方交付项目时，声明项目的性能数据。

2、瓶颈分析：在能力验证的过程中可能会发现一些瓶颈，通过技术手段分析瓶颈，得到分析数据，为后续调优做理论依据。

3、响应超时：什么负载量的时候出现超时现象？

4、tps 达到瓶颈，波动剧烈：tps 瓶颈点在哪里？，在什么地方出现性能衰减？

5、性能调优：在得到瓶颈分析数据之后，做性能调优。

6、降低超时，提高 tps，减少抖动。。

7、容量规划：基于未来。为将来的用户激增提前做准备

8、数据库扩容

9、服务端硬件优化（增加 cpu，扩充磁盘，提升带宽，分布式，负载均衡。。。）

七、压力工具的工作原理是什么？

答：jmeter 工作原理：基于协议，通过多线程的方式模拟用户行为，设计各种场景压测服务端，得到性能数据，分析性能瓶颈

八、性能测试基本思路是什么？

答：1、测什么：明确测试目标（明确需求）

2、怎么测：怎么设计场景？

测试计划，测试用例，测试方案、数据准备、参数化，表达式，断言、场景设计（并发，负载，压测）

得到性能测试结果、测试结果验证

验证结果数据是否符合预期

如果预期响应时间是 3s，但是实际结果响应时间达到了 5s 不合格

预期最大 tps 需要达到 500，但是实际最大的 tps 只有 300 不合格

八、测试哪些关键场景？

答：1、浏览器层面：

web 端和 app 端（H5 页面）

关注首屏时间（页面打开到完全呈现）

脚本加载时间，cpu 占用，fps 频率（帧频率越高，流畅度越高）

2、接口层面

权限划分

普通用户权限（非常多的的用户）

大并发场景：包括查询，表单提交

数据量也需要考虑（电商平台，门户网站）

负载场景：用户长期在线访问，对资源的要求会很高

3、管理员权限（几个用户）

大数据量的场景（管理几十万用户数据）

列表查询时间，分页时间

数据下载（excel,数据导出）是否会内存溢出

mysql 数据库是否会死锁，sql 查询是否异常

4、超管（1 个）

大数据量的场景（管理几十万用户数据）

九、前端性能测试关注哪些点？了解哪些前端性能优化方法？

答：首屏时间:页面完全展现需要的时间

白屏时间：页面第一帧画面出现之前的时间

脚本加载时间，fps，cpu，network

前端性能优化：使用缓存，压缩图片，压缩 js，css，gc 回收优化，js 前置

十、解释常用的性能指标的名称与具体含义

答：1、用户角度

响应时间（rt）从发起请求，到接口响应，到页面渲染

错误率(error)

2、 服务端角度

rps（request persecond）每秒请求数，用户发起的

tps（transaction persecond）每秒完成事物数，服务端决定的

通过 rps 指标，来测试 tps，从而衡量服务端性能。

3、浏览器层面

qps（query perecond）每秒查询接口数（uv pv）

刷新一次页面，调用了三个接口

hps（hit persecond）每秒点击率

十一、性能测试类型有哪些？按顺序描述

答：并发数先确定

基准测试（得到性能数据，为后续的回归测试做理论依据）

单接口基准测试

容量基准测试

负载测试

不断增加负载量（压力），一直到瓶颈点出现，可以停止

压力测试

1:稳定性压测

假设瓶颈点在 300tps，用对应的负载量的 80%-90%做持续性（几小时或者几天）的压测。目的是发现稳定性问题（内存溢出等等）

2:破坏性压测

用对应负载量的 100%或者 150%做压测，直接让服务器出现异常。目的是及早的暴露问题

失效恢复测试

服务端出现异常之后能不能及时恢复

十二、什么是集合点？设置集合点有什么意义？jmeter 中如何设置集合点？

答：集合点更多的运用在并发测试

为了让压力尽可能的落在同一个时间点

十三、什么是固定等待和隐式等待？

答：固定等待：超时时间=0

线程数一定要>=集合数。一定要集合完毕才发起请求

隐式等待：超时时间>0

达到超时时间范围，无论集合多少线程都会发起请求

十四、你在测试中遇到过哪些性能问题？

答：h5 页面响应时间过长

h5 的分页经常卡死，sql 查询过多，数据量过大

导出 excel 时间过长，页面 503，后台报内存溢出

功能涉及到算法的时候，一定要在测试环境用大量数据去模拟

只要点击，后台 cpu 立刻就是 300%

十五、你在性能场景设计中用到哪些方法？

答：参数化，关联，断言，jdbc 连接

十六、什么是关联，如何动态关联？有哪几种关联的方法？

答：正则关联，json 关联，xpath 关联

保证接口上下游是衔接的

十七、jmeter 负载测试中怎么保持 session 会话？

答：${__setProperty(cookie${counter},${COOKIE_beegosessionID},)} 存储 session

${__P(cookie,)} 从属性表提取 session

十八、什么是 Ramp up？你如何设置？

答：线程启动的时间

ramp 越大，单位时间内的压力越小。ramp=0 表示单位时间压力无穷大，线程启动时间无穷小。ramp=0 不代表时间为 0

十九、如何识别性能瓶颈？

答：随着负载不断升高，tps 也是不断升高的，正常逻辑

随着负载不断增加，tps 不再增加，甚至下降。表示单位线程的 tps 实际在衰减。tps 的瓶颈点

二十、简述堆区的空间分配和 gc 原理

答：年轻代

1 个 eden

2 个存活区（S1 和 S2）

老年代

GC（垃圾回收）

内存溢出：OOM(OUT OF memory)

1：运行内存>当前空间剩余内存

2：垃圾不能及时回收

年轻代 GC:

1：最初的对象是存活在 eden；伊甸园空间满了之后，会进行第一次 GC；

2：第一次 GC 之后，依然存活的对象，会被丢到 S1（第一个存活区）；

3：S1 初次满了之后，会进行第二次 GC（年轻代 GC）

4：第二次 GC 之后，依然存活的对象，会被丢到 S2（第二个存活区）,同时清空伊甸园和 S1;

5：S2 满了之后，会进行第三次 GC，依然存活的对象，会再次被丢到 S1，同时清空伊甸园和 S2;

年轻代里面的垃圾碎片都是比较小；老年代的碎片比较大；

让垃圾尽可能的在年轻代里面进行回收；否则会影响老年代空间的整理；

老年代 GC:

1：年轻代的 GC 年龄超出阈值（默认 16 次），会把年轻代依然存活的对象扔到老年代；

2：对象的尺寸超出了阈值；对象尺寸超出了阈值，会直接进入老年代；

3：对象的大小超出了年轻代剩余的空间大小，直接进入老年代；

老年代 GC=fullGC(一般默认)

1：老年代剩余空间不足以对象进入；老年代会直接进行一次 fullGC;
2：老年代的对象无法进行 GC；老年代会进行一次 fullGC;

老年代的对象尺寸都比较大，所以 gc 时间会很长，同时所有线程会出现暂停；
jvm 调优是为了规避 fullgc 的频繁出现；会影响到 tps；
jstat -gcutil pid 1000 监听 gc 情况
jmap -heap pid 查看内存空间分配情况
jvm 参数调优
调堆内存空间，调年轻代的 gc 年轻，调空间分配比例（老年代：年轻代 / eden：存活区）;调 gc 回收器，并行回收机制

二十一、什么是内存溢出？

答：1：运行需要的内存大于空间剩余内存；会出现内存溢出
2：垃圾无法进行 GC;会出现内存溢出

二十二、简述 cpu 的工作原理

答：分析 cpu 参数：lscpu
CPU(s): 2 cpu 个数
Thread(s) per core: 1 每核的线程
Core(s) per socket: 1 每个卡槽的核
Socket(s): 2 每个 cpu 的卡槽
双 cpu，4 核多线程；每个 cpu 是双核；

cpu 负载和利用率
可运行的进程（双 r）+不可中断的进程（block）
running（运行中的）+runnable（等待运行的）+block（等待 io）=cpu 的负载

利用率：cpu 双线程如果都在调度 java 进程，表示当前 cpu 利用率是 100%；
cpu 双线程如果只有一个 java 进程在调度，表示当前 cpu 利用率是 50%；

最理想的情况：每个 cpu 线程都调度一个 java 进程，此时的负载=4；
不理想的情况：java 进程数远大于 cpu 线程数，此时负载会远远超出 cpu 线程数；不能及时调度的进程就会排队；
cpu 会给运行中的和等待运行的进程数均匀分配时间片；
cpu 调度是以时间片为基准的；假设调度时间为 1ns，调度超出 1ns 之后，进程会被挂起；切换到下一个进程
队列越长，cpu 时间片就越小，调度时间就会越短，切换的越快；切换的过于频繁，cpu 利用率就会很低，线程也会暂停

进程是最小资源分配单元；
线程最小调度单元；

二十三、什么是上下文切换？哪些场景会存在上下文切换？

答：1：线程的切换
切换的时间加载寄存器和计数器
保存数据和位置信息，然后切换到下一个线程并读取它的数据和位置信息
2：进程切换
3：特权切换

系统调用（切换两次上下文）

用户空间向内核空间发起申请，内核空间返回 api 给用户空间调用；

二十四、什么是 swap 空间？oomkiller 了解吗？怎么开启 swap 空间

答：swap：从磁盘空间开辟的虚拟用户空间。

cat /proc/sys/vm/swappiness 查看 swap 比例

当内存空间使用超出了比例，会启用 swap 空间（内存交换）

so 换出

si 换入

swapon -a 启用 swap

sudo sysctl vm.swappiness=10 临时修改 swappiness 比例

vim /etc/sysctl.conf 永久修改 swappiness 比例

二十五、什么是进程优先级？

答：pr 和 ni

ni 范围 -19---+20，ni 值越小，进程优先级越大；ni 越大，进程优先级越小；

优先级越高的进程，优先调度 cpu，时间片分配的越多；

二十六、吞吐量大幅度波动有哪些原因？

答：上下文切换的过快；

gc 次数过于频繁；

二十七、哪些现象说明了 IO 瓶颈？

答：await=io 等待时间=io 处理时间+io 队列时间

svctm=io 处理时间

await 与 svctm 的差值越大，表示队列时间越长

util 磁盘繁忙度，值越大，磁盘越繁忙；

二十八、了解哪些资源监控命令？

答：top 家族

top，htop，atop，iftop（查看网络），iotop（查看 io）

sysstat 家族

vmstat vmstat 1 10

mpstat mpstat -P ALL 1 10 查看逻辑 cpu

iostat iostat -x -k -d 1 10 查看 io 处理

netstat 查看网络情况

pidstat 查看进程

二十九、如何用命令行生成测试报告？

答：jmter -J{参数名} -r{host} -n -t XX.jmx -l XX.jtl -e -o httpreport

三十、性能遇到的问题：

答：tps 非常低，响应时间长

联合索引失效 --- explain +sql 语句定位

联合索引最左原则，从左到右查数据

使用 EXPLAIN 关键字可以模拟优化器执行 SQL 查询语句，从而知道 MySQL 是如何处理你的 SQL 语句的。分析

你的查询语句或是表结构的性能瓶颈。

三十一、性能定位可能会导致的问题

答：1、CPU 过高：(命令 top)

接口加密，需要计算

压缩解压资源

代码 bug 导致死循环

json 序列化 --- 工具 jprofiler 监控定位

回收机制频繁，大量计算导致 CPU 过高

2、内存：（free）

垃圾回收异常

3、网络 IO（netstat）

4、TPS 抖动，出现毛刺(抖动幅率区间在 5%左右属于正常)

网络问题导致抖动

垃圾回收机制导致抖动

三十二、常用函数

答：CSVRead 读取文件参数化

Random 随机生成数

count 计算总数

time 获取当前时间

machinename 机器名(分布式压测)

machineid 机器名 id(分布式压测)

三十三、性能测试目的

答：1、测试系统最大处理能力

寻找系统最大的 TPS，判断 TPS 和对应响应时间是否满足预期

2、测试系统支持最高并发

寻找系统最高能支持多少并发，当系统出现宕机、进程崩溃、报错率持续上升、响应时间超过可忍受范围，程序无响应等情况，即可认为系统达到了可支持的最高并发

三十四、性能测试场景

答：先进行单接口测试

再按照一定的并发比例，进行多接口混合测试

最后按照混合场景比例，进行长时间稳定性测试

三十五、性能测试执行策略

答：1、加压策略

从小并发开始，逐步增加并发，寻找性能拐点

2、执行策略

试压阶段：寻找拐点，记录拐点数据

收集数据：选择拐点前后 5 组数据，按照固定时间(3-5 分钟)重新跑一次，记录详细数据

三十六、APP 端性能测试

答：1、先区分哪些接口是与服务端交付

2、APP 性能测试，需要考虑站手机的 CPU、内存、流量、电量、流畅度、稳定性等指标

3、工具：Appium、monkey 等

4、监测：adb 命令、腾讯 PerfDog、滴滴 DoraemonKit 等

三十七、 Redis

答：1、数据存储在内存，数据不易过大

2、支持每秒十几万次的读/写操作，速度很快

3、支持集群、分布式、主从同步等，还支持一定事务能力

4、应用场景：

缓存 --用户登陆 token

消息队列 --支付

活动排行榜或计数

发布、订阅消息、消息通知

商品列表、评论列表

有效期控制

三十八、性能监控 -- CPU/内存/网络 IO/磁盘 IO

答：1、Linux 监控命令：top、free、iostat、df、wmstat

2、redis 监控：redis-stat

三十九、内存溢出

答：1、堆内存溢出

堆内存中存在大量对象，这些对象都有被引用，当所有对象占用空间达到堆内存的最大值，就会出现内存溢出

四十、内存泄露现象

答：1，tps 出现大幅波动，并慢慢降低，甚至降为 0，响应时间随之波动，慢慢升高

2，通过 jstat 命令看到，Jvm 中 Old 区不断增加，FullGC 非常频繁，对应的 FullGC 消耗的时间也不断增加

3，通过 jconsole/jvisualvm 可以看到，堆内存曲线不断上升，接近上限时，变成一条直线

4，日志报错 java.lang.OutOfMemoryError: Java heap space

四十一：在什么样的场景下监控内存泄露问题？

答：1，在试压阶段，或任意场景都可以考虑通过 jvisualvm 和 jstat 监控 jvm 的情况

2，在稳定性场景中，一定要关注 Jvm 内存使用的情况，在长时间的压测下，最容易看出内存泄露的问题

四十二、线程死锁问题

答：线程死锁就是有两个线程，一个线程锁住了资源 A，又想去锁定资源 B，另外一个线程锁定了资源 B，又想去锁定资源 A，两个线程都想去得到对方的资源，而又不愿释放自己的资源

从而造成一种互相等待，无法执行的情况

四十三、线程阻塞问题

答：在多线程情况下，如果一个线程对拥有某个资源的锁，那么这个线程就可以运行资源相关的代码。而其他线程就只能等待其执行完毕后，才能继续争夺资源锁，从而运行相关代码。

1、减少代码中没有必要的日志输出

2、如果可以，提升日志级别，以减少日志

3、如果可以，更换其他的日志组件，如 Log4j2、Logback 等

四十四、CPU 消耗过高问题

答：压测过程中，发现应用服务器的 CPU 使用率比较高（>80%）

两种情况：

1、接口的性能非常好，比如响应时间<10ms，tps 很高，此时 CPU 使用率高是正常的，不需要优化

2、接口性能不好，比如响应时间>200ms，tps 很低，此时需要考虑优化

CPU 消耗高可能的原因

1、使用了复杂的算法，比如加密、解密

2、压缩、解压、序列化等操作

3、代码 bug，比如死循环

四十五、数据库性能问题 -- 索引

答：现象：tps 很低，响应时间很长，数据库服务器 cpu 很高（接近 100%），应用服务器负载比较低

数据库服务器 CPU 高，一般都是因为 SQL 执行效率低导致的，可能有三方面原因

1、数据库表缺少必要的索引；

2、索引不生效

3、SQL 不够优化

四十六、代码优化

答：1、使用对象池减少对重复对象的创建；

2、调整对后端的连接

3、增加本地缓存

4、如果不涉及事务的情况下，考虑使用 Nosql 进行存储

5、一次请求合并多次操作

6、由串行修改为并行操作

7、同步修改为异步

四十七、性能瓶颈定位思路（面试必问）

答：整体思路：从前到后，从表象到内部

1、首先排除压力机自身的问题，如 CPU、内存，网络，脚本编写等

2、监控中间件的访问日志，观察响应时间，大体确定耗时处于哪一段

3、排查网络问题，监控压力机到后端服务器的网络，以及各服务器间的网络，是否达到网络上限

4、监控服务端所有机器的操作系统负载，如 CPU、内存、磁盘、网络是否达到瓶颈

5、监控应用服务器的日志，查看是否存在 ERROR 日志，比如 TimeOut 或其他类型报错

6、监控各中间件的连接数，如 nginx、tomcat、mysql 等，是否达到上限

7、监控应用程序线程状态，使用 jstack 或 jvisualvm 查看是否有死锁、阻塞等情况

8、监控应用程序的 jvm，使用 jstat 或者 jmap 查看 GC 情况，是否内存泄漏等

9、使用 jprofiler 监控应用程序，可以查看耗时比较长的代码方法

10、监控数据库，是否存在慢查询，一般数据库 CPU 高都是因为 SQL 语句效率低造成的

11、检查数据库执行计划，是否有全表扫描，以及索引不生效的情况

12、检查系统外部依赖情况，如果外部依赖系统性能差，也会造成本系统性能低

13、对于不好定位的问题，可以考虑采用模块隔离法来确定问题