**性能测试**

**性能测试不仅包括测试，还包括分析和调优。**

**第一个模块是性能测试基础篇。我想在这个模块里澄清一些性能测试的基础概念，讲解一些关键部分。但并不是对概念的简单描述，而是根据实际项目，告诉你真正具有指导价值的性能测试概念是什么，并解析这些概念在实际操作中的指导性作用。**

**在第二个模块中，我将通过性能测试工具的实际操作实例，对应性能测试的前后逻辑关系。在这一部分中，我会重点给你讲解，为什么要使用某些工具的某些功能，以便确保工具的使用及结果是为性能测试需求指标和性能分析报告而服务的，而不是浮于表面的“炫技”。**

**在第三个模块中，我将通过操作系统、应用服务器、数据库、缓存服务器、Java、C++ 等监控工具的使用和分析方法，告诉你它们产生的数据在性能分析过程中该如何判断，为测试报告及性能分析提供有效的历史数据。**

**最后一个模块是对前三个模块的凝练，我会讲解不同实际操作场景中的性能测试分析过程，比如实际的瓶颈判断的过程是怎样的，怎么分析出根本的原因，如何提出具体的解决方案，最后的实施效果又是怎样的。**

性能测试的指标、测试分类、测试设计有很深刻的理解。还要学习系统业务和架构相关知识，这样才能更好的设计性能场景，分析出系统的性能瓶颈。性能测试常用的工具有jmeter和loadrunner。

性能测试的意义，常见性能指标的理解,性能测试的分类.性能测试的流程包括需求分析,场景设计,脚本编写,执行资源监控,性能调优,回归测试. jmeter和loadrunner的使用,性能测试报告输出.

典型的两个业务方向，一个是音频业务，一个是视频业务。尤其视频，已经是目前甚至是未来信息交互的主要形式，而视频业务对于带宽的要求会更高，相同并发下对CPU、内存的消耗也更多。

**线程:**线程是程序中的一个执行流，每个线程都有自己的专有寄存器(栈指针、程序计数器等)，但代码区是共享的，

即不同的线程可以执行同样的函数。一般在性能测试过程中，我们认为一个线程就是一个虚拟用户。

**多线程:** 多线程是指程序中包含多个执行流，即在一个程序中可以同时运行多个不同的线程来执行不同的任务，

也就是说允许单个程序创建多个并行执行的线程来完成各自的任务。一个进程是由一或多个线程组成，进程只负责资源的调度和分配，线程才是程序真正的执行单元，负责代码的执行。

**多线程的好处：**可以提高CPU的利用率。在多线程程序中，一个线程必须等待的时候，CPU可以运行其它的线程而不是等待，这样就大大提高了程序的效率。并发用户数：在同一时刻与服务器进行交互的在线用户数量。

**性能测试:** 性能测试是通过自动化测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能测试。

**性能测试分类:**

**性能指标:**

响应时间：从客户端发起的一个请求开始，到客户端接收到从服务器端返回的响应结束，这个过程所耗费时间。

平均响应时间（average）:每个请求request的平均响应时间。

吞吐量TPS（Transaction Per Second）:每秒钟系统能够处理的交易或事务的数量。

并发数：即并发用户数，指系统可以同时承载的正常使用系统功能的用户数量。

每秒点击次数HPS（Hits Per Second）：指在一秒钟的时间内用系统处理的事务量。它一般和TPS成正比关系，是B/S系统中非常重要的性能指标之一。

**事务（Transaction）:** 通俗理解就是用户某一步或几步操作的集合。一般，事务的定义都需要跟业务相关，比如用户一次登录过程，使用淘宝时的支付过程等，需要理解事务不等于请求。

**事务响应时间:**完成整个事务所用的时间，这个性能指标是我们在业务层面重点关注的一个指标。如跨行转账业务。

**TPS（Transaction Per Second）:**每秒事务数，它是吞吐量的常用量化指标，系统吞吐量是最能直接体现软件系统负载承受能力的指标。

**TPS需要理解的几个点：**(利用jmeter测试的指标为Throughput，在聚合报告中可以查看到该指标。)

QPS（Query Per Second）：每秒查询数。HPS/RPS（Request Per Second）：每秒HTTP请求数。QPS（TPS）= 并发数/平均响应时间.如果是单接口请求，QPS有时也说TPS，指的就是系统的统统量，每秒请求/事务数。

**一个原则，一个系统的性能其实是由TPS决定，跟并发用户数没有多大关系。一般在同样的TPS下，可以用不同并发用户数去压测得到，只是得到响应时间不同。**

**响应时间公式：**响应时间=请求传输时间+Server处理时间+响应传输时间+前端解析渲染时间。

一般测试的是服务器级别的响应时间，不包括前端解析渲染时间。2/5/8 原则：2秒之内用户觉得很快， 5秒之内用户觉得还可以，8秒之外用户觉得系统慢，无法忍受。

**并发用户数:**在同一时刻与服务器进行了交互的在线用户数量。并发用户数是指对Server产生压力的用户数。

**在线用户数:**当一个用户登录到一个网站或应用后，无论是否进行发送请求操作，这些用户都称为在线用户。

**两者的关系：**一般并发用户数是包含在在线用户数中的。并发用户数计算：一般需求采集人员会将线上的并发用户数根据日志或工具分析统计出。

**资源利用率:**指的是对不同系统资源的使用程度，例如服务器的CPU利用率，磁盘利用率等。资源利用率是分析系统性能指标进而改善性能的主要依据。

**并发场景策略设计**:多个用户同时在同一系统进行同一操作，比如同时购买商品。多个用户同时在同一系统进行不同操作，比如有的在浏览查看商品，有的在购买商品。

**Jmeter**

**Jmeter：**apache组织使用纯java开发的一个开源免费测试工具，它可以实现接口和性能测试。用于对软件做压力测试，它最初被设计用于Web应用测试，但后来扩展到其他测试领域。它可以用于测试静态和动态资源，例如静态文件、Java 小服务程序、CGI 脚本、 Java 对象、数据库、FTP 服务器， 等等。Jmeter能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回期望的结果。

**JDK安装与安装配置:**

官网下载解压http://jmeter.apache.org/ ,配置环境变量: 在系统变量框，点击“新建”，建立一个变量：JMETER\_HOME,值为你解压的jmeter安装路径。配置classpath量%JMETER\_HOME% %JMETER\_HOME%\lib\ext\ApacheJMeter\_core.jar;%JMETER\_HOME%\lib\jorphan.jar;%JMETER\_HOME%\lib/logkit-2.0.jar; 确定即可.

**文件目录：**

**bin目录：**主要存放可执行文件，主程序的jar包，配置文件，日志文件等。

**常用文件：**jmeter.bat:windows下的应用启动文件。jmeter.sh:linux下启动文件。jmeter.log：应用运行的日志文件。jmeter.properties：系统配置文件，经常需要根据需要修改。

**说明：**jmeter.bat/jmeter.sh：启动脚本中，可以对Jmeter的启动参数JVM堆内存大小进行配置，默认为512M,建议修改成1024M。\* setHEAP=Xms1024m-Xmx1024m\*

docs目录：存放Jmeter官方API文档，便于进行二次开发。

extras目录：扩展插件目录，常用的是 Jmeter与ant集成的文件。

lib\ext目录:该目录存放的是Jmeter的插件或者扩展组件。Jmeter会自动在lib和ext下寻找需要的类。

一般扩展的依赖包，会打成jar包放入ext目录。

printtable\_docs目录：该目录存放的是官方用户手册。

**组件介绍:**

**线程组:**用来模拟用户，所以也叫Users。一个线程组模块可以包含多个线程，每个线程代表一个用户，这样可以模拟高并发下的请求，并根据网站的响应信息来判断网站的相关性能。

**线程数：**虚拟用户数，即并发数，一个线程表示一个虚拟用户；

**Ramp-Up Period：**所有线程启动的时间，单位s。通过这个参数可以设置每个线程间的启动间隔；

**循环次数：**测试循环的次数，如果勾选了“永远”，那么所有线程会一直发送请求，直到手动停止运行脚本；

**监听器（Listener）：**是用来监听系统资源的元件。它是用来对测试结果数据进行处理和可视化展示的一系列元件。 图形结果、查看结果树、聚合报告、用表格察看结果都是我们经常用到的元件。

**聚合报告(aggregate report):**对于每个请求，它统计响应信息并提供请求数，平均值，最大，最小值，错误率，大约吞吐量(以请求数/秒为单位)和以kb/秒为单位的吞吐量.

**测试计划(Test Plan)：**使用 Jmeter 进行测试的起点，它是其它 Jmeter 测试元件的容器。

**线程组(Thread Group)：**代表一定数量的并发用户，它可以用来模拟并发用户发送请求。

**采样器(sampler)：**定义实际的请求内容，被线程组包含，比如HTTP请求、java请求等。

**监听器(Listener)** ：响应结果的显示，统计等。

**逻辑控制器(Logic Controller)** ：帮助用户控制Jmeter的测试逻辑及执行顺序。

**断言(Assertions) ：**用来检查从服务器获得的响应内容是否符合预期。

**配置元件(Config Element) ：**初始化默认值和变量，以便采样器使用。

**前置处理器(Pre Processors)和后置处理器(Post Processors)** ：请求前后的处理操作。

**定时器(Timer)：**为采样器设置等待时长。

**每个接口请求会显示独立的一行数据，每行有13个字段，含义分别如下：**

Label：请求名或者请求标签，每个JMeter 的组件（例如 HTTP Request）都有一个 “名称” 属性，这里显示的就是“名称”属性的值。

**#Samples：**表示你这次测试中一共发出了多少个请求，如果模拟10个用户，每个用户迭代10次，那么这里显示就是100了。

Average：平均响应时间，默认情况下是单个 Request 的平均响应时间，当使用了 Transaction Controller 时，也可以以Transaction 为单位显示平均响应时间。

Median：中位数，也就是所有请求响应时间中排在中间的那个响应时间点，就是50% Line，参考90% Line解释。

90% Line：90%用户请求响应时间，如果把某次任务的所有请求的响应时间按从小到大排序，它是指排在90％处那个点的请求的响应时间，也就是说有90%的请求的响应时间小于等于这个响应时间。后面的50/60/70/80/90/95/99 % Line 也是类似的含义。

95% Line：95%用户请求响应时间，请参考90% Line解释。 99% Line：99%用户请求响应时间，请参考90% Line解释。Min：最小响应时间，所有请求样本中的最小响应时间。 Max：最大响应时间，所有请求样本中的最大响应时间。

Error%：本次测试中“出现错误的请求数/请求的总数”百分比值。

Throughput：吞吐量，服务器一定时间范围内处理的请求数。在本报告中它的含义其实其实是吞吐率，表示每秒完成的请求数（Request per Second），此指标代表服务器的处理能力，比如支付宝峰值处理量达到12万笔/秒。

Received KB/sec：每秒从服务器端接收到的数据量。 Sent KB/sec：每秒从客户端发送到服务器端的数据量。

**jmeter操作数据库：**

接口测试中对数据进行查询，传入数据作为请求参数，或者断言时查看数据是否正确写入数据库。接口或者性能测试，通过接口并发请求，批量实现数据的构造，数据恢复操作。快速实现对数据库的增删改查。

添加数据库驱动:将对应的数据库驱动jar包放入jmeter的lib目录下，以mysql驱动为例，重启jmeter生效。

添加JDBC Connection Configuration（数据库配置连接池）添加路径：测试计划->添加->配置元件->JDBC Connection Configuration. 添加线程组及JDBC Request. 查看执行sql语句响应数据

**Jmeter接口测试之参数关联:**

**关联:**请求之间通过传递参数建立联系。一般需要将一个请求的响应参数，作为另一个请求的入参。比如登录后的操作，第一步实现登录请求，然后将请求返回的token提取出来保存到一个变量中，后续请求作为入参使用。

**Top命令**

**top命令：**Linux下常用的性能分析工具，能够实时监控系统的运行状态，主要显示系统中各个进程的资源使用情况，并根据cpu及内存等进行排序，类似于windows系统中的任务管理器，可以实时查看系统的各种资源状态。

首先前五行显示的系统资源的总体统计信息：

**第一行：类似于update命令，**他可以快速简洁的获取系统的负载，依次显示现在时间，多少个用户，系统1分钟、5分钟、15分钟的平均负载。up xx days, HH:MM：系统当前时间和系统已运行时间， 当前登录用户数，系统平均负载可以运行任务的平均数量（3个数值分别统计最近1，5，15分钟的系统平均负载）系统平均负载：单核CPU情况下，0.00 表示没有任何负荷，1.00表示刚好满负荷，超过1侧表示超负荷，理想值是0.7。

**第二行：Tasks 任务（进程）**系统现在共有的进程数，运行中进程数，休眠中的进程数，停止状态的进程数，僵尸状态进程数。

**第三行：cpu状态信息，具体属性说明如下：**

0.2% us：用户空间占用CPU的百分比

0.3% sy：内核空间占用CPU的百分比

0.0% ni：改变过优先级的进程占用CPU的百分比

99.6% id：空闲CPU百分比，反映一个系统cpu的闲忙程度，越大越空闲。

0.0% wa：IO等待占用CPU的百分比

0.0% hi：硬中断（Hardware IRQ）占用CPU的百分比

0.0% si：软中断（Software Interrupts）占用CPU的百分比

0.0% st：Steal time虚拟机被hypervisor偷去的CPU时间

**第四行：内存信息：**

物理内存总量：total 空闲的内存总量：free 使用中的内存总量：used 缓存的内存总量：buffers

**第五行（Swap）：交换分区**

总的交换空间大小；空余空间交换空间大小；已经使用交换空间大小；cached缓冲的交换空间大小

**进程列表说明：**

PID: 进程pid USER: 拉起进程的用户

PR: 该列值加100为进程优先级，若优先级小于100，则该进程为实时(real-time)进程，否则为普通(normal)进程，实时进程的优先级更高，更容易获得cpu调度，以上输出结果中，java进程优先级为120，是普通进程，had进程优先级为2，为实时进程，migration 进程的优先级RT对应于0，为最高优先级

NI: 进程的nice优先级值，该列中，实时进程的nice值为0，普通进程的nice值范围为-20~19

VIRT: 进程所占虚拟内存大小（默认单位kB） RES: 进程所占物理内存大小（默认单位kB）

SHR: 进程所占共享内存大小（默认单位kB） S: 进程的运行状态 %CPU: 采样周期内进程所占cpu百分比

%MEM: 采样周期内进程所占内存百分比 TIME+: 进程使用的cpu时间总计 COMMAND: 拉起进程的命令

**常用参数说明：**

M：根据驻留内存大小进行排序；

P：根据CPU使用百分比大小进行排序；

T：根据时间/累计时间进行排序；

d：屏幕刷新间隔时间；

p<进程号>：指定进程

分享一些前端和后端性能测试工具的基本原理，并基于 LoadRunner 去分析一些大型企业性能测试的规划、设计、实现的具体实例.

分享一些跨国软件公司性能测试卓越中心的实际案例.

**对于不同类型的系统，软件性能的关注点各不相同，比如：**Web 类应用和手机端应用，一般以终端用户感受到的端到端的响应时间来描述系统的性能；非交互式的应用，比如典型的电信和银行后台处理系统，响应时间关注更多的是事件处理的速度，以及单位时间的事件吞吐量。

**系统响应时间**反应的是系统能力，又可以进一步细分为应用系统处理时间、数据库处理时间和网络传输时间等；

**前端展现时间**取决于用户端的处理能力。

**不同的对象群体可以分为四大类：终端用户、系统运维人员、软件设计开发人员和性能测试人员。**

**终端用户眼中的软件性能:**

响应时间是终端用户对系统性能的最直观印象，包括系统响应时间和前端展现时间.

**系统运维人员眼中的软件性能:**

单个用户的响应时间外，更要关注大量用户并发访问时的负载，以及可能的更大负载情况下的系统健康状态、并发处理能力、当前部署的系统容量、可能的系统瓶颈、系统配置层面的调优、数据库的调优，以及长时间运行稳定性和可扩展性。最常见的情况就是，系统运维人员必须在最大并发用户数和系统响应时间之间进行权衡取舍。

**软件设计开发人员眼中的软件性能:** 软件性能通常会包含算法设计、架构设计、性能最佳实践、数据库相关、软件性能的可测试性这五大方面。

**第一，算法设计包含的点：**核心算法的设计与实现是否高效；必要时，设计上是否采用 buffer 机制以提高性能，降低 I/O；是否存在潜在的内存泄露；是否存在并发环境下的线程安全问题；是否存在不合理的线程同步方式；是否存在不合理的资源竞争。

**第二，架构设计包含的内容：** 站在整体系统的角度，是否可以方便地进行系统容量和性能扩展；应用集群的可扩展性是否经过测试和验证；缓存集群的可扩展性是否经过测试和验证；数据库的可扩展性是否经过测试和验证。

**第三，性能最佳实践包含的点：**代码实现是否遵守开发语言的性能最佳实践；关键代码是否在白盒级别进行性能测试；是否考虑前端性能的优化；必要的时候是否采用数据压缩传输；对于既要压缩又要加密的场景，是否采用先压缩后加密的顺序。

**第四，数据库相关的点：**数据库表设计是否高效；是否引入必要的索引；SQL 语句的执行计划是否合理；SQL 语句除了功能是否要考虑性能要求；数据库是否需要引入读写分离机制；系统冷启动后，缓存大量不命中的时候，数据库承载的压力是否超负荷。

**软件性能的可测试性包含的点：**是否为性能分析（Profiler）提供必要的接口支持；是否支持高并发场景下的性能打点；是否支持全链路的性能分析。

**性能测试人员眼中的软件性能:**

关注的是算法设计、架构设计、性能最佳实践、数据库相关、软件性能的可测试性这五大方面。

性能需求的总结和抽象能力；根据性能测试目标，精准的性能测试场景设计和计算能力；性能测试场景和性能测试脚本的开发和执行能力；测试性能报告的分析解读能力；性能瓶颈的快速排查和定位能力；性能测试数据的设计和实现能力；面对互联网产品，全链路压测的设计与执行能力，能够和系统架构师一起处理流量标记、影子数据库等的技术设计能力；深入理解性能测试工具的内部实现原理，当性能测试工具有限制时，可以进行扩展二次开发；极其宽广的知识面，既要有“面”的知识，比如系统架构、存储架构、网络架构等全局的知识，还要有大量“点”的知识积累，比如数据库 SQL 语句的执行计划调优、JVM 垃圾回收（GC）机制、多线程常见问题等等。

**软件性能的三个最常用的指标：并发用户数、响应时间，以及系统吞吐量**。

**并发用户数:**性能需求与测试最常用，也是最重要的指标之一。它包含了业务层面和后端服务器层面的两层含义。

**业务层面的并发用户数:**指的是实际使用系统的用户总数。但是，单靠这个指标并不能反映系统实际承载的压力，我们还要结合用户行为模型才能得到系统实际承载的压力。

**后端服务器层面的并发用户数:**指的是“同时向服务器发送请求的数量”，直接反映了系统实际承载的压力。

分析得到准确的用户行为模式，是性能测试中的关键一环。但，获得精准的用户行为模式，是除了获取性能需求外，最困难的工作.—从用户行为模式来获取真正对服务发起请求的人=服务器并发请求数.

获取用户行为模式的方法，主要分为两种：

对于已经上线的系统来说，往往采用系统日志分析法获取用户行为统计和峰值并发量等重要信息；而对于未上线的全新系统来说，通常的做法是参考行业中类似系统的统计信息来建模，然后分析。

**响应时间**

反映了完成某个操作所需要的时间, 分为前端展现时间和系统响应时间两部分。其中，前端时间，又称呈现时间，取决于客户端收到服务器返回的数据后渲染页面所消耗的时间；而系统响应时间，又可以进一步划分为 Web 服务器时间、应用服务器时间、数据库时间，以及各服务器间通信的网络时间。

注意, 前端性能评估，则应该采用用户主观感受时间的定义。后端性能工具推荐loadrunner和jmeter，前端性能工具推荐webpagetest和Yslow

**系统吞吐量**

体现软件系统承受负载能力的指标，但也必须和其他指标一起使用才能更好地说明问题。

对性能测试而言，通常用“Requests/Second”“Pages/Second”“Bytes/Second”来衡量吞吐量。当然，从业务的角度来讲，吞吐量也可以用单位时间的业务处理数量来衡量。

“Bytes/Second”和“Pages/Second”表示的吞吐量，主要受网络设置、服务器架构、应用服务器制约；

“Requests/Second”表示的吞吐量，主要受应用服务器和应用本身实现的制约。

如果考察HTTP或者业务层面，可以选择“Requests/Second”“Pages/Second”如果考察系统层面或网络层面，可以选择“Bytes/Second”，即网卡每秒接收/发送到字节数.

**并发用户数、响应时间、系统吞吐量之间的关系:**可分为空闲时间,线性增长区间,拐点,过饱和区间.

**常用的七种性能测试方法: 后端性能测试、前端性能测试、代码级性能测试,压力测试,配置测试、并发测试，以及可靠性测试.**

**后端性能测试:** (平时听到的性能测试，大多数情况下指的是后端性能测试，也就是服务器端性能测试。)

**通过性能测试工具模拟大量的并发用户请求，然后获取系统性能的各项指标，并且验证各项指标是否符合预期的性能需求的测试手段。**指标除了包括并发用户数、响应时间和系统吞吐量外，还应该包括各类资源的使用率，比如系统级别的 CPU 占用率、内存使用率、磁盘 I/O 和网络 I/O 等，再比如应用级别以及 JVM 级别的各类资源使用率指标等。

**根据应用领域的不同:** 基于性能需求目标的测试验证；探索系统的容量，并验证系统容量的可扩展性.

**前端性能测试:** 浏览器端的页面渲染时间、资源加载顺序、请求数量、前端缓存使用情况、资源压缩等内容，希望借此找到页面加载过程中比较耗时的操作和资源，然后进行有针对性的优化，最终达到优化终端用户在浏览器端使用体验的目的。减少 http 请求次数, 减少 DNS 查询次数, 避免页面跳转, 使用内容分发网络（CDN）, Gzip 压缩传输文件.

**代码级性能测试:** 是指在单元测试阶段就对代码的时间性能和空间性能进行必要的测试和评估，以防止底层代码的效率问题在项目后期才被发现的尴尬。

**改造现有的单元测试框架。**将原本只会执行一次的单元测试用例连续执行 n 次，这个 n 的取值范围通常是 2000~5000；统计执行 n 次的平均时间。如果这个平均时间比较长（也就是单次函数调用时间比较长）的话，比如已经达到了秒级，那么通常情况下这个被测函数的实现逻辑一定需要优化。

**压力测试:** 一般采用后端性能测试的方法，不断对系统施加压力，并验证系统化处于或长期处于临界饱和阶段的稳定性以及性能指标，并试图找到系统处于临界状态时的主要瓶颈点。

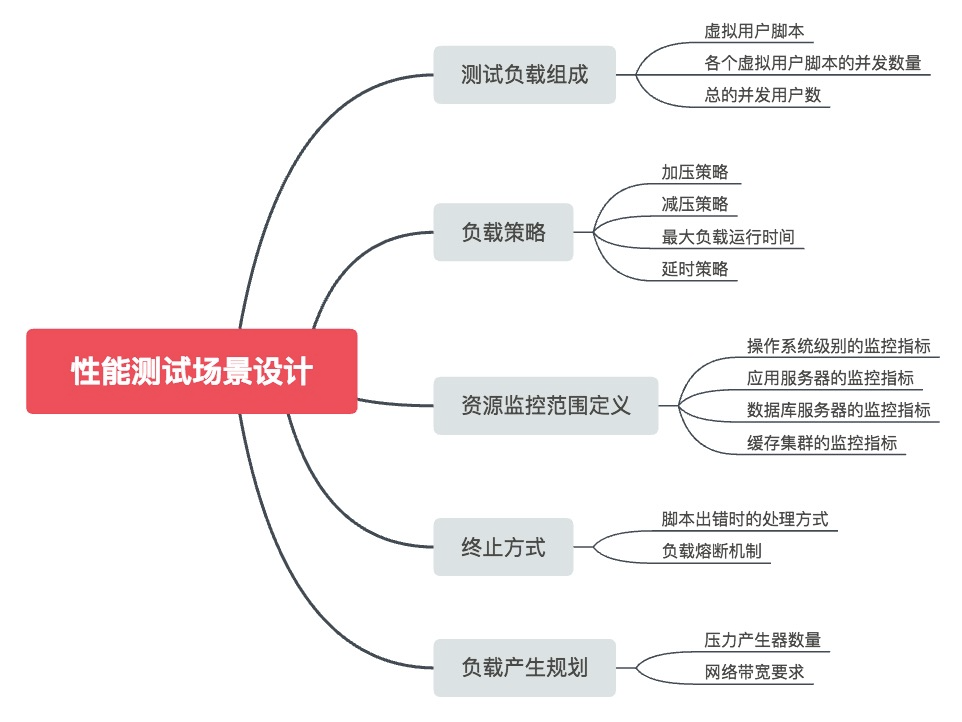
**配置测试:** 主要用于观察系统在不同配置下的性能表现，通常使用后端性能测试的方法.

**并发测试:** 指的是在同一时间，同时调用后端服务，期间观察被调用服务在并发情况下的行为表现，旨在发现诸如资源竞争、资源死锁之类的问题。

**可靠性测试:** 是验证系统在常规负载模式下长期运行的稳定性。

**性能测试的四大应用领域:** **能力验证、能力规划、性能调优、缺陷发现这四大方面**

|  |
| --- |
|  |
| **能力验证** | **能力规划** | **性能调优** | **缺陷发现** |
| **后端性能测试** | **√** | **√** | **√** | **√** |
| **前端性能测试** |  |  | **√** |  |
| **代码级性能测试** |  |  | **√** | **√** |
| **压力测试** | **√** | **√** | **√** | **√** |
| **配置测试** |  | **√** | **√** |  |
| **并发测试** |  |  | **√** | **√** |
| **可靠性测试** | **√** | **√** | **√** |  |

完整的后端性能测试应该包括性能需求获取、性能场景设计、性能测试脚本开发、性能场景实现、性能测试执行、性能结果报告分析、性能优化和再验证。后端性能测试工具, 是后端性能测试中的一个必要步骤而已.

后端性能测试工具和 GUI 自动化测试工具最大的区别，即它们模拟用户行为的方式以及测试的执行方式不同。

后端性能测试工具的基本原理。它首先通过虚拟用户脚本生成器生成虚拟用户脚本；然后根据性能测试场景设计的要求，通过压力控制器控制协调各个压力产生器以并发的方式执行虚拟用户脚本；同时，在测试执行过程中，通过系统监控器收集各种性能指标以及系统资源占用率；最后，通过测试结果分析器展示测试结果数据。