测试技术五大流程

1. 性能测试（性能测试流程+案例？）
2. 你还负责过性能测试吗？详细描述一下你是怎么做性能测试

的？

Key:回答较为简化的性能测试流程，我们做性能测试分为三个阶段

第一阶段：确定需求🡪分析交易，调研性能指标🡪设计场景和用例🡪搭建性能测试环境🡪准备性能测试数据和脚本

第二阶段：开发脚本，调试脚本，增强脚本

第三阶段：分析结果，定位问题，回归测试，编写测试总结

1. 我看你还做过不少金融类的项目/OA类的项目/证劵类的项目/

视频类的项目等等（结合学生随机抽取），能举一个例子告诉我如何进行测试的吗？

（这个性能测试1个月，确定需求+调研性能指标1-2天，设计场景和用例3-5天，搭建环境2天，写脚本准备数据3-5天、执行用例测试1周，回归测试1周，总结写报告3天）

上家公司（北京金蝶管理软件有限公司+北京市顺义区复兴四街金蝶软件园）民生花园2区步行出发到软件园约10分钟

项目有哪些功能模块（首页，凭证，资金，发票，工资，期末转结，帐簿，报表，一键报税，设置，进销存）

3个人 Java 一个半月）

第一阶段： （结合项目讲解）

1. 确定需求

* 测试的范围

由项目经理或者开发负责人或者产品经理提供接口文档说明书

* 测试的业务及主要功能

财务组（新增凭证，查看凭证，查询总账）首页，凭证，资金，发票，工资，期末转结，帐簿，报表，一键报税，设置，进销存

1. 分析交易，调研指标（需要结合项目，带入交易）

* 交易是什么？（3到4支）为什么针对该交易做性能？

交易：新增凭证，查看凭证、总账查看

原因：用户使用频繁、特定时间会发生多用户同时访问的现象、交易路径较长

* 本次性能测试需要收集哪些指标以及各指标的指数要求？

吞吐量不小于500、新增凭证、查看凭证和总账平均响应时间<3秒，服务器资源使用率（cpu<80%、内存<80%、磁盘<80%）、并发用户数暂定1000 因为当时测试的时候系统没有上线，这些指标是参照其他已上线的同类型的系统，并且和项目经理确认

1. 设计场景和用例

* 设计了哪些场景？
  1. 单交易基准场景：一个用户去执行一个交易，比如说一个用户正在使用视频剪辑，执行5-10min（5-10min怎么控制？一般使用线程组自带的调度器）
     1. 目的：验证单个交易在无压力的情况下是否有性能问题
  2. 单交易负载场景：多个用户去执行一个交易（以阶梯性的形式依次增加并发用户数），比如说多个用户正在使用视频剪辑功能，执行5-10min（分为同步并发和阶梯式并发，同步并发怎么做？使用同步定时器，再问什么是集合点，固定等待和隐式等待的区别？；递增式并发怎么做？使用steping thread group线程组）
     1. 目的：验证单个交易最大处理能力
  3. 混合测试场景：多个用户去执行多个交易（压力比？4:4:2怎么确定的？？如果产品上线可以找运维协助分析后台数据；如果没有上线可以和同类产品对比，或者发行测试版，手机测试数据），比如说现在有100个用户，其中有10个用户正在使用视频剪辑功能，50个用户正在使用视频编码功能，40个用户正在使用视频转换功能（在jmeter中怎么设计的？使用吞吐量控制器）
     1. 目的验证系统的最大处理能力，可以通过tps或者吞吐量（单位时间内完成的请求数）来判断
  4. 稳定性测试场景：往往我们的系统会执行5x8h或者7x24h，这个时候我们就需要验证系统在长时间大压力情况下系统的性能问题，选择混合测试场景下最大压力的80%（80%：系统最大处理能力也就是性能拐点的80%）（）
  5. 基于目标场景这么设计？？使用常量吞吐量定时器；

举例某个交易的吞吐量要达到300，平均响应时间小于2秒，怎么设计？

常量吞吐量定时器中的target throughput（目标吞吐量）=300\*60=18000，因为常量吞吐量定时器中的target throughput是按分钟来计算的，平均响应时间只需要在聚合报告中查看即可。

* 如何设计性能测试用例？

写了多少条用例（200条），用了多长时间（5天）

用例：用例的基本信息（用例标题、用例编写姓名、用例前置条件）、集合点、断言、测试步骤（对应性能测试场景）、预计结果和实际结果

用例给了我们一个用例模板，以新增凭证为例，我们一般会写用例标题、用例编写人、此次交易的url地址、用例前置条件（比如说：做单交易基准时要确保没有其他人使用这个系统，网络，硬件配置等没有问题）、脚本设置（包括参数化设置（要传入的付款交易过多时，就可以做参数化，可以针对付款的交易做参数化），集合点设0置（批量付款时，可以设置一个集合点，比如说有20条交易时就可以进行付款），检查点设置（就是断言，比如返回值中包含200，就是付款成功））、场景设置（测试步骤，仅针对付款的话场景就是单交易基准和单交易负载）、预期结果、实际结果、等等

1. 搭建性能测试环境：（**为什么选择这套环境——因为我们这个系统当时没有上线嘛，这套测试环境是最接近用户使用环境的，客户端使用4 台PC机是因为我们要做分布式负载测试，因为一个虚拟用户就占2M内存，如果只用一台的话不够用）**

* 客户端环境：压力产生机（PC,硬件配置，lr11）

硬件环境：4台PC机，联想，处理器：Intel i7 内存8G 固态硬盘：512G

软件环境：win7，jmeter3.1

* 服务器环境：软件（操作系统，数据库，中间件。。。），硬件（型号，CPU,内存，磁盘）

硬件环境： SR658，处理器：i7，内存：64G，四块，硬盘：4T,

软件环境：操作系统：CentOS6.5，

数据库：MySQL5.6，中间件：Tomcat7.0

* 测试工具以及监控工具？

测试工具：jmeter3.1（JDK1.8）

监控工具：spotlight、以及服务器自带的

1. 准备性能测试数据和脚本

* 如何造数据，以及准备了多少条数据
* 如何准备脚本？

1. 脚本的准备：如果有接口文档的话，按照接口文档写脚本；无接口文档的话：一般用badboy进行简单录制脚本，测试一下可以运行的话继续开发增强。或者用fiddler抓包自己写一份接口文档，然后再按照接口文档写脚本

**Fiddler抓包抓哪些？**

**接口名称，接口英文名称，请求方式，url地址，调用参数，返回值**

1. 测试数据的准备：数据准备10000条，需要单独的压测数据库。

方法：excel导入数据库中或者自写sql语句生成测试数据或者直接在后台数据库表中添加

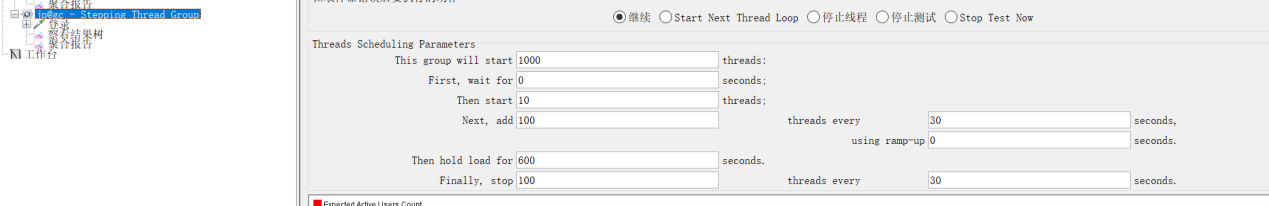
第二阶段：（结合项目讲解）

* 开发脚本，调试脚本，增强脚本

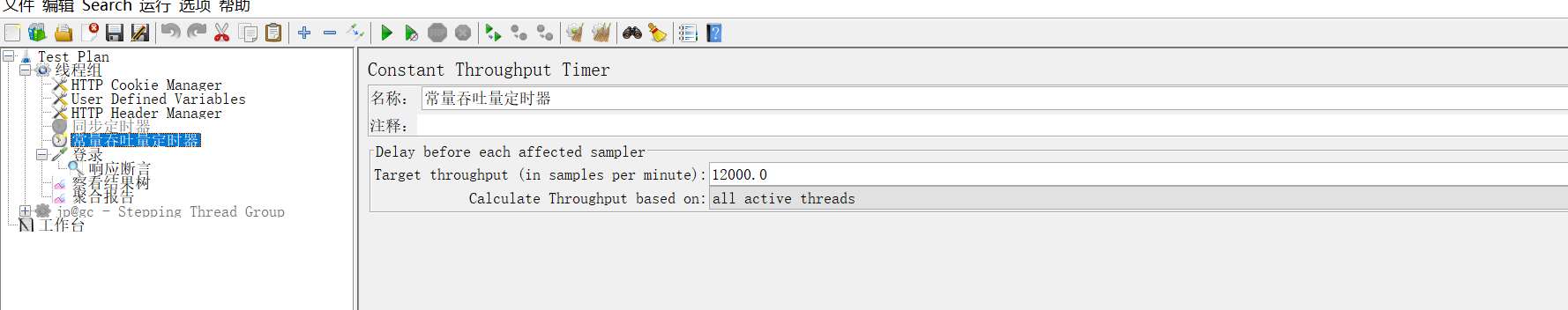
关联、断言、参数化

* 根据前期设计的场景进行执行

1. 单交易基准
2. 单交易负载场景：jp@gc - Stepping Thread Group



1. 混合场景：多个用户跑多个交易，需要用到一个组件：**吞吐量控制器**
   1. 压力比如何确定
      1. 未上线系统：可以和项目经理以及开发运维人员进行确定，查询相关数据和日志分析得到压力比；或者自行预估出一个压力比，进行混合测得到相关指标和项目经理进行确认；或者开展一个小活动实际调研用户的使用行为
      2. 已上线系统：参考实际交易数据，后台统计数据以及数据库实际数据记录
2. 基于目标场景设计：常量吞吐量定时器

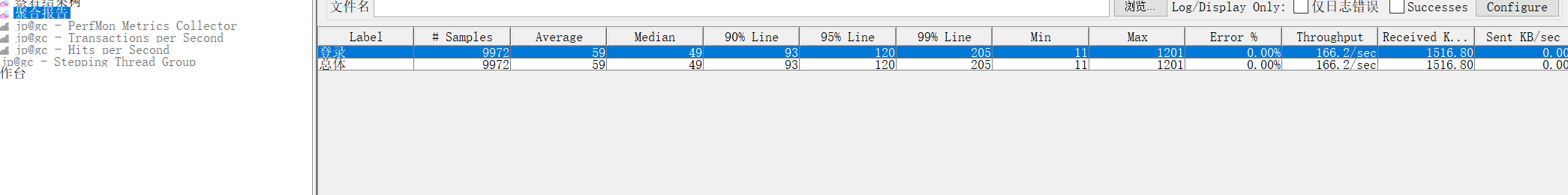


1. 分布式负载测试：
   1. 调度机：
      1. 将所有执行机的IP地址以及端口号添加到
   2. 执行机：
      1. 确定执行机的ip地址以及端口号

* 收集结果，监控各项指标值

执行完场景后，我们使用jmeter聚合报告组件收集结果，查看性能的各项指标

1. 收集指标用到一个监听器🡪聚合报告（聚合报告可以收集哪些指标？）



Label：http请求名称

Samples：完成的总请求数

**Average：平均响应时间（单位：ms）**

Median：50%用户的响应时间小于该值

90%Line：90%用户的响应时间小于该值

95%Line：  
95%Line：

Min：最小的响应时间

Max：最大的响应时间

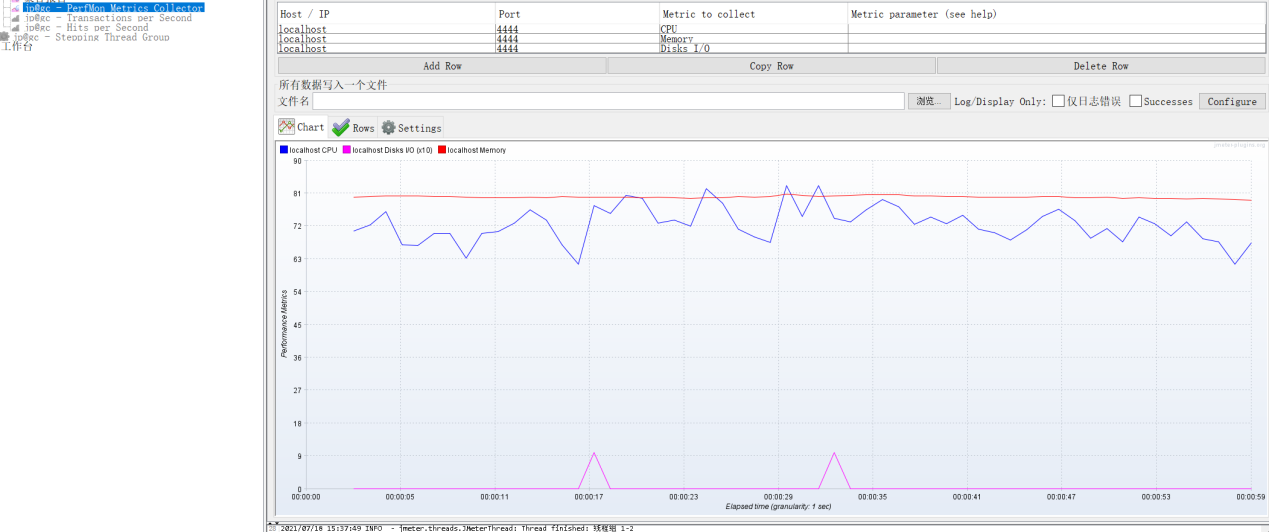
**Error：请求的错误率**

**Throughput：吞吐量，单位时间内完成的请求数**

Received：单位时间内服务端接收到的请求数量

Sent：单位时间内客户端发送的请求数量

1. 收集服务器硬件指标，用到一个监听器🡪jp@gc - PerfMon Metrics Collector（性能指标收集器）（cpu、内存、磁盘使用情况使用该组件来收集）



1. 监控Linux服务器硬件资源使用情况：
   1. sar -u 1 10：每1s收集1次cpu使用情况，共计10次
   2. sar -r 1 10：每1s收集1次内存使用情况，共计10次
   3. sar -d 1 10：每1s收集1次磁盘使用情况，共计10次
   4. sar -o：将收集的数据保存到某个文件中：sar -u -o /home/movie/0718cpu.txt 1 10
   5. sar -f ：读取保存的文件信息：sar -f 0718cpu.txt
2. 第三方检测工具：spotlight

第三阶段：（结合项目讲解）

* 分析结果，定位问题，提交bug。

系统问题：一般通过查看服务日志进行问题排查，或者由开发人员协助查看日志进行问题定位，主要查看日志当中error或者warn信息

环境问题：硬件的配置、数据库配置（数据库连接池、索引？）、中间件配置（tomcat连接池）、测试脚本、测试工具配置

* 回归测试
* 测试总结
* 如果不做性能测试，我们的系统在运行的过程中会遇到哪些性能问题？

1. 有些功能响应时间长？
2. 并发用户多了，系统奔溃，死机等等？
3. 软硬件不协调，有可能造成资源浪费？
4. 无法还原性能问题，支持不了运维人员？
5. 无法评估系统的寿命，做不了合理的性能规范？

* 为什么做性能测试？性能测试作用？

第一、为新系统的上线提供数据依据。例如：响应时间，并发用户数等等。

第二、已发生性能问题的系统，提供故障复现，监测系统的性能瓶颈。

针对上述流程和性能测试案例追问以下问题：

1. 脚本是用什么语言开发的？

Java

1. 几个人做的？各自负责什么？

2个人1.测试用例编写2.测试脚本以及测试数据的准备服务部署

1. 测试了多少个接口，花费了多长时间，什么时候开始的？

4个，一个月

1. 这个项目是什么时候做的性能测试？你怎么可能还记这么清楚指标和环境？

几个月前做的，因为当初这个测试方案是我负责编写的，所以记得比较清晰点

1. OSI有几层，http协议对应哪一层？

7层



1. 进程与线程的区别？

背：进程是系统资源分配的基本单位，而线程是处理器任务执行的基本单位。通常在一个进程中包含若干个线程，线程可以利用进程中的一些资源，而且进程奔掉的话不会影响其它进程，但是线程奔掉的话进程也会受到影响奔溃掉

1. **以线程的方式运用虚拟用户有哪些优点？**

线程相对于进程来说，线程使用的内存更小，开销小，运行速度更快，可以将更多的使用内存分配给虚拟用户。

1. **了解连接池吗？请介绍一下？**

C3P0：开源的JDBC连接池，有自动回收空闲连接能力，提供最大的空闲时间

DBCP：没有自动回收空闲连接能力，提供最大连接数

连接池：连接池是将已经创建好的链接保存在连接池中，当有请求的时候可以直接使用已经创建好的链接对数据库进行访问，省略了创建链接和销毁链接的过程，在性能上得到了提高，

1. 除了Jmeter，你还了解哪些性能测试工具？

Loadrunner：是分布式用户负载测试工具。它能够模拟很多用户对网站进行负载测试，

vug 脚本开发 压力调度分析器controller 结果搜集： analysis

1、虚拟用户脚本生成器（Virtual User Generator）VuGen VUG-----功能：录制、编辑、调试测试脚本

2、压力调度控制台（Controller）---功能：创建场景、运行场景、监控场景、收集测试数据

3、压力结果分析器（Analysis ）----功能：把收集到的测试数据以图表的形式展示出来，保存生成测试报告。

（不太熟悉，但是我觉得loadrunner与jmeter的功能是一样的如果贵公司使用的是loadrunner我觉得我也是很快就能上手的）

1. 性能监测工具了解哪些？nmon、spotlight

spotlight（监控Linux或者win或者数据库）或者nmon（监控linux）

jmeter自带的监控工具：聚合报告（平均响应时间average、吞吐量throughput、error）、PerfMon Metrics Collector、TPS（每秒0处理的事务数）、HPS（每秒点击率）

1. 负载均衡的原理，实现方式（回答上来两个即可）

负载均衡（Load Balance），将负载（访问请求）进行平衡、分摊到多个服务器上进行执行。因为服务器有多台，而我们的网址比如说淘宝只有一个，想要用户的请求分发到集群中不同的机器上就需要使用负载均衡

**（1）http重定向协议实现负载均衡**

原理：根据用户的http请求计算出一个真实的web服务器地址，并将该web服务器地址写入http重定向响应中返回给浏览器，由浏览器重新进行访问。

**（2）dns域名解析负载均衡**

原理：在DNS服务器上配置多个域名对应IP的记录。例如一个域名www.baidu.com对应一组web服务器IP地址，域名解析时经过DNS服务器的算法将一个域名请求分配到合适的真实服务器上。

**（3）反向代理负载均衡（Nginx）**

原理：反向代理处于web服务器这边，反向代理服务器提供负载均衡的功能，同时管理一组web服务器，它根据负载均衡算法将请求的浏览器访问转发到不同的web服务器处理，处理结果经过反向服务器返回给浏览器。

**（4）IP负载均衡**

原理：在网络层通过修改目标地址进行负载均衡。

**（5）数据链路层负载均衡**

原理：在数据链路层修改Mac地址进行负载均衡。

1. 脚本进行了哪些增强（断言、关联、参数化等）？会写脚本吗？

参数化，关联，检查点（脚本的断言），集合点

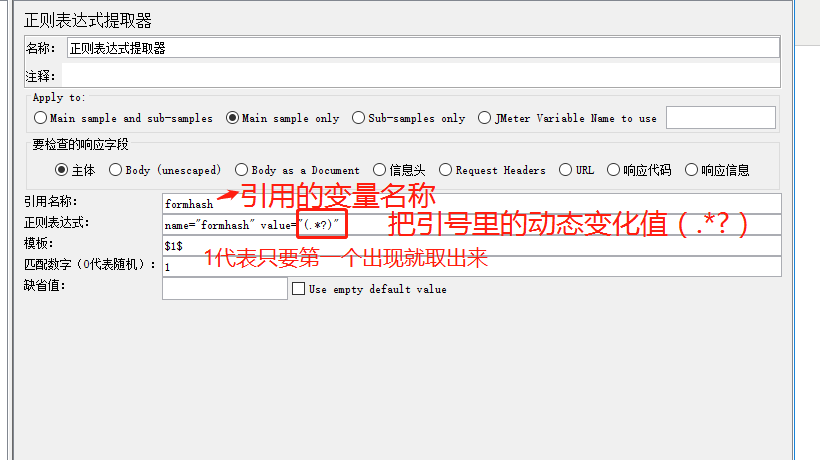
1. 如何进行关联的，交易哪些功能做了关联？如何识别出需要关联？

当前请求依赖上次请求从服务器端返回的数据（第一次访问服务器返回a，第二次访问服务器需要用第一次返回的结果a，这种情况称为关联）

正则表达式提取器（**正则表达式设置的参数都要知道，如何获取第几个值 $1$**）

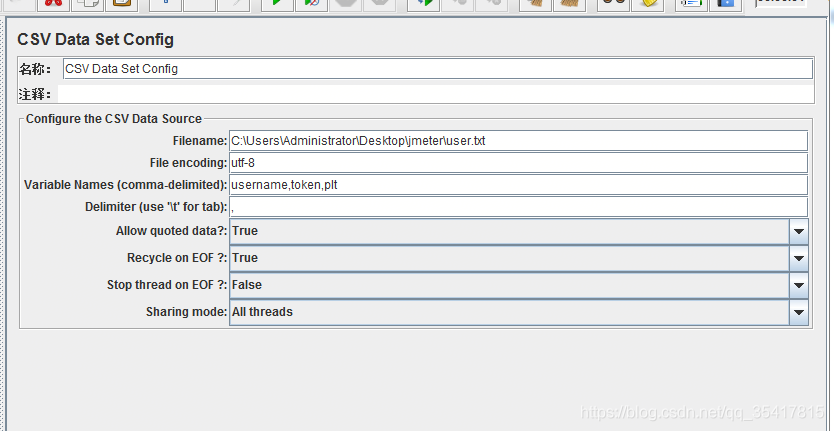
数据用（.+？）代替

* . : 任意字符
* +: 一个或者多个
* ?:匹配一组则结束



我当时测的云账房系统所有的查询交易都需要与用户的token值进行关联，也就是说所有的账套都必须是当前的用户名下的账户

1. 参数化的方式有哪些？用到了哪些组件？
2. CSV Data Set Config-从本地文件取数据（.txt文件）

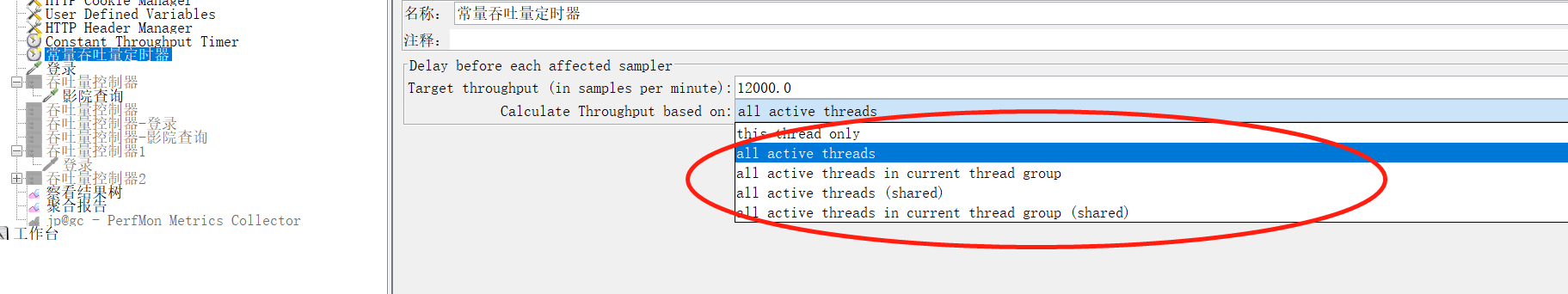


1. JDBC参数化技术

先导入mysql-connector（柯南科特）-java-5.1.7-bin.jar包、使用组件JDBC Connection Configuration（康飞个瑞申）、JDBC Request（瑞款斯特）（最后设置${\_\_V(查询变量名\_${随机数变量名})}）

1. 场景设计的方法有哪些？如何基于目标值设计场景？
   * 单交易基准：一个用户跑一支交易，跑个5-10min
   * 单交易负载：多个用户以阶梯性递增的方式跑一支交易，跑个5-10min
   * 混合测试场景：多个用户跑多支交易，跑10-30min
   * 稳定性测试场景：在混合测试场景最大压力下的80%进行12个小时的长时间测试

基于目标场景：constant throughput timer常量吞吐量定时器，首先要先将吞吐量设置为500（第一行的数=吞吐量\*60），然后看他的响应时间是否能达到要求



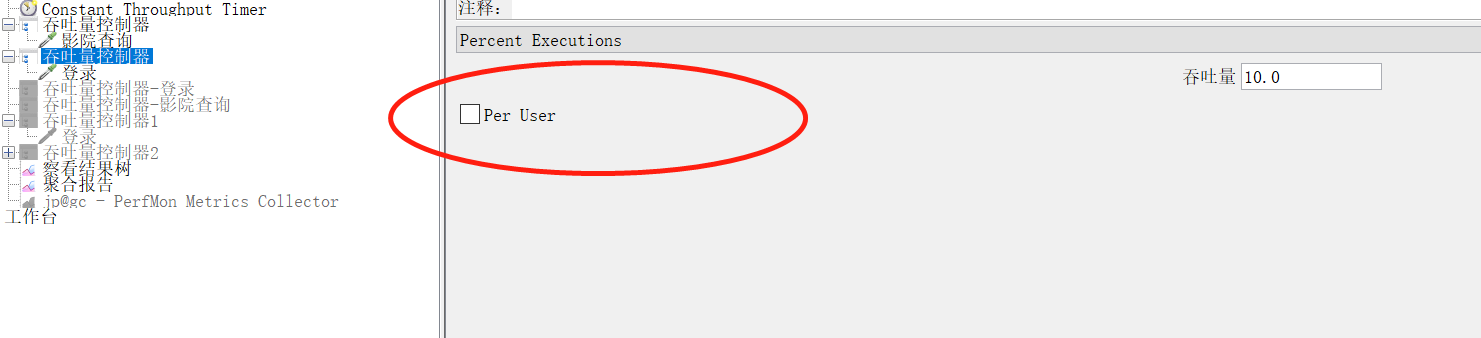
* **This thread only：**控制每个线程的吞吐量，选择这种模式时，总的吞吐量为设置的target Throughput 乘以该线程的数量
* **All active threads**：设置的target Throughput 将分配在每个活跃线程上，每个活跃线程在上一次运行结束后等待合理的时间后再次运行。活跃线程指同一时刻同时运行的线程。
* **All avtive threads(shared)：**与All active threads的选项基本相同。唯一区别是，每个活跃线程都会在所有活跃线程上一次运行结束后等待合理的时间后再次运行。
* **All active threads in current thread group：**设置的target Throughput 将分配在当前线程组的每一个活跃线程上，当测试计划中只有一个线程组时，该选项和All active threads 选项的效果完全相同。
* **All active threads in current thread group(shared)**：与All active threads in current thread group 基本相同，唯一的区别是，每个活跃线程都会在所有活跃线程的上一次运行结束后等待合理的时间后再次运行。

1. 混合测试场景？如何做并发的？

压测比：3：3：4（付款：收款：电子对账的并发用户数的比例）：并发用户数

因为是未上线的系统：压测比是参考同类型的系统并且和项目经理确认的

组件：吞吐量控制器，要在吞吐量控制器下添加交易的http请求



1. 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？

为了让压力尽可能落在同一个时间点，所以我们采用了集合点，

组件名称：Synchronizing Timer同步定时器

a) Number of Simulated Users to Group by：按组分组模拟用户数（需要在集合点等待的线程总数，如果设置为0，等同于线程组(Thread Group)的线程总数）

b) timeout in milliseconds（单位：ms）：timeout意思是等待请求多久后，不管线程数有没有达到设置的并发数量都开始运行测试

i. 固定等待：线程数>集合点数，且等待超时=0

ii. 隐式等待：线程数=<>集合点数，等待超时>0

1. **什么是内存溢出？**

内存溢出：程序在申请内存时，没有足够的内存空间供其使用

内存泄漏：是指程序在申请内存后，无法释放已申请的内存空间，一次内存泄露危害可以忽略，但内存泄露堆积后果很严重，无论多少内存,迟早会被占光。

1. 什么是固定等待和隐式等待？

固定等待就是线程数（10）>集合点数（8），等待超时设置为0，点击执行时，立即执行8个线程，剩下的2个一直在等待，不执行。

隐式等待就是线程数（10）>集合点数（8），等待超时设置为20s，点击执行时，立即执行8个线程，剩下的2个等待20s后执行

注：

* 线程数（8）<集合点数（10），等待超时=0

点击运行时不执行，停止后，执行8个进程

* 线程数（8）<集合点数（10），等待超时=2000ms

等待2s后执行8个进程，执行完后停止

* 线程数（10）=集合点数（10），等待超时=5000ms

立即执行，不等待

* 线程数（10）>集合点数（8），等待超时=0

点击执行后，立即执行8个进程，点击停止后，执行剩下的2个进程

* 线程数（10）>集合点数（8），等待超时=5000ms

点击执行后，立即运行8个进程，等待5s后，执行剩下的2个进程

1. 如何用命令行生成测试报告？

jmeter -n -t 【脚本路径】 -l 【结果文件路径】-j 【执行日志路径】 -e -o 【测试报表生成文件夹】

参数详解：

-n：设置命令行模式

-t：指定JMX脚本路径，若非当前目录需用相对路径

-l：指定结果文件路径

-e：设置测试完成后生成测试报表

-o：指定测试报表生成文件夹。文件夹可指定路径保存，但前提文件夹必须为空或不存在

-j：指定执行日志路径，路径不存在不会自动创建

1. 什么是分布式负载测试？如何实现？

使用一台调度机和多台执行机进行负载测试

a) 调度机：远程操控执行机，本机依然执行

i.将执行机的ip地址和端口号配置到jmeter的bin目录下jmeter.properties文件中remote\_hosts中，多个ip地址以逗号进行隔开，然后启动jmeter-server.bat

ii.执行之前保证调度机和执行机中的jmeter版本和jdk版本保持一致，脚本、参数化文件及路径保持一致，本地服务都要全部启动。

b) 执行机：执行分布式负载测试所用的压力机

i.执行机配置：首先获取本机ip地址和jmeter端口号(jmeter的bin目录下jmeter.properties文件中server\_port=1066)，然后执行jmeter-server.bat

1. 性能测试过程中有发现bug吗？你是如何分析这些bug的？

平均响应时间远远大于3秒，数据库查询语句复杂，大量的 where 子句，order by， group by 排序等，CPU容易出现瓶颈

1. 数据库方面一般会有哪些性能问题？

（1）数据库锁问题，死锁：同一时间只有一个事务能进行修改，以保证数据库数据的一致性

（2）数据库索引问题，没有建索引

1. 影响性能。 如果子表外键没有创建索引，那么当父表查询关联子表时，子表将进行全表扫描。影响表连接方式。

2. 影响并发。 无论是更新父表主键，或者删除一个父记录，都会在子表中加一个表锁（在这条语句完成前，不允许对子表做任何修改）。这就会不必要

地锁定更多的行，而影响并发性

3：在特殊情况下，还有可能造成死锁。

1. 吞吐量大幅度波动有哪些原因？

代码需要优化、网络原因、服务器遇到大批量请求，有延迟处理的迹象、压力高峰时server端短暂休克(页面显示不全，或者不显示需要多次刷新)、服务器运行不稳定、应用服务过程受其他非性能测试干扰

1. 如何识别性能瓶颈？
2. **查看系统日志，如果日志记录的全面，很容易通过日志发现问题。比如，系统宕机时，系统日志打印了某方法执行是抛出out of memory的错误，很快定位到导致内存溢出的问题在哪里。**
3. **利用性能监控工具，比如：linux系统环境下通过nmon来监控系统性能。**
4. **设计合理的性能测试场景，好的测试场景能更加快速的发现瓶颈。**
5. **了解系统参数配置，可以进行后期的性能调优**
6. 测试过程中出现性能问题，你会从哪些方面考虑？

**系统问题**：一般通过查看服务日志进行问题排查，或者由开发人员协助查看日志进行问题定位，主要查看日志当中error或者warn信息

**环境问题**：硬件的配置、数据库配置（数据库连接池、索引？）、中间件配置（tomcat连接池）、测试脚本、测试工具配置

1. 一个接口查询速度过慢，可能有哪些原因导致？

* 应用服务器或数据库服务器的硬件：CPU，内存，磁盘等
* 网络问题：

带宽（bps：单位时间内传输的字节数），交换机（传输数据的），路由器，OSI七层（物理层、数据链路层、网络层、传输层、会话层、表示层、应用层），TCP/IP协议

* 程序问题：就是代码的问题

算法优化（程序代码需要优化），例如：变量使用不释放，连接数据库不关闭资源造成内存泄漏。

* SQL问题：数据库问题

索引（为了快速查找数据），多表查询的笛卡尔积，数据库死锁等等

数据查询使用过多的where条件查询

+**框架设计，模块设计，算法与数据流设计，**界面设计（软件架构设计实际上要求软件架构师找到一个设计的平衡点，在软件的可用性，可靠性，可维护性，高性能等取平衡点。如果追求高性能，必须在设计阶段就规划好，否则设计完成，调整成本很高，有时可能需要重构）

* 中间件问题：

tomcat，webservice，websphere，JVM内存和调优，redis（数据库中间件）

（这个性能测试1个月，确定需求+调研性能指标1-2天，设计场景和用例3-5天，搭建环境2天，写脚本准备数据3-5天、执行用例测试1周，回归测试1周，总结写报告3天）