

软件性能测试能力提升解决方案

课程试用

- 测试经理 TM
- 测试主测 TC
- 测试架构师 TAE
- 测试设计 TE
- 项目质量 QA
- 项目经理 PM
- 开发经理 DM
- 及相关 PMO/IPMT成员

问题

- 测试需要技术么？
- 我们需要什么技术？
- 你懂的！

TOP10

- 能力提升1——性能测试模型
- 能力提升2——性能测试方法
- 能力提升3——软件架构分析技术
- 能力提升4——模块性能瓶颈点分析技术
- 能力提升5——测试场景萃取技术
- 能力提升6——测试场景构建技术
- 能力提升7——性能常用监控工具
- 能力提升8——性能指标解析
- 能力提升9——数据建模技术
- 能力提升10——性能 数据分析技术

什么OO

分析模型 + 设计模式 = OO

学习本课程的意义：

- 1、掌握性能测试分析模型；
- 2、掌握性能测试场景、用例设计模式；

性能测试能力不足表现



系统分析	萃取场景	场景设计	数据分析
分析能力表现在：分解复杂的事物，抽丝剥茧和庖丁解牛的能力。确保后面的设计所需对象的完整性和正确性。	萃取能力表现在：从不同的视角准确评估被测对象商用标准。识别关键路径，提取有效测试场景。	设计能力表现在：从已分析出的测试场景中提取关键测试点，有效检验设计目标是否达到商业应用要求。	分析能力表现在：建立有效的测试数据分析模型，通过数学方法及分解技术，找到瓶颈点及方案设计缺陷。
能力不足表现在： 1、架构对象分析遗漏； 2、模块对象分析遗漏； 3、对象依赖关系错误；	能力不足表现在： 1、评估点空洞； 2、提取的测试场景无法反映；	能力不足表现在： 1、迭代场景设计； 2、集合点设置； 3、事物不合理；	能力不足表现在： 1、数据建模方法； 2、模型分析方法； 3、瓶颈点识别技术；

“功能测试”Vs.“性能测试”区别

	功能测试	性能测试
测试目的	<ol style="list-style-type: none"> 验证系统是否满足需求 验证系统设计是否存在缺陷 验证系统文档及安装过程是否合理有效 	<ol style="list-style-type: none"> 验证系统是否满足实际业务需求 查找系统中可能存在的性能缺陷或性能瓶颈
测试依据	系统的功能需求（主要来自于系统需求规格说明、游戏功能策划案）	系统的性能需求（主要来自于对系统自身估量，包括系统背景、运行的环境、系统应用的服务对象等）
测试过程	<ol style="list-style-type: none"> 分析测试需求 赛选测试策略 设计测试用例 模拟场景数据 构建测试环境 执行测试套件 提交系统缺陷 验证缺陷存在 	<ol style="list-style-type: none"> 分析测试需求 系统技术调研 选择测试模型 测试数据建模 构建度量环境 执行测试套件 分析测试数据 提交性能缺陷
行业背景	要求对系统业务非常熟悉	要求对系统业务和用户场景非常熟悉

如何评价一个系统的性能

- 用户视角论：
 - 应用者视角 (Customer)
 - 运营商视角 (Manager)
 - 开发商视角 (Developer)

应用者视角 (Customer)

- 对于产品的最终使用者，他们则更加关注产品的可用性，我们是否按照实际的业务需求进行合理的产品设计；系统可否按照预先设计承受G量级的并发访问；系统是否能支撑长时间的稳定运行；是否为我们所涉及的工作节省了不必要的开支。
- 参考指标：
 1. 响应时间 (Response Time)
 2. 游戏可玩性 (Game Quality)

运营商视角 (Manager)

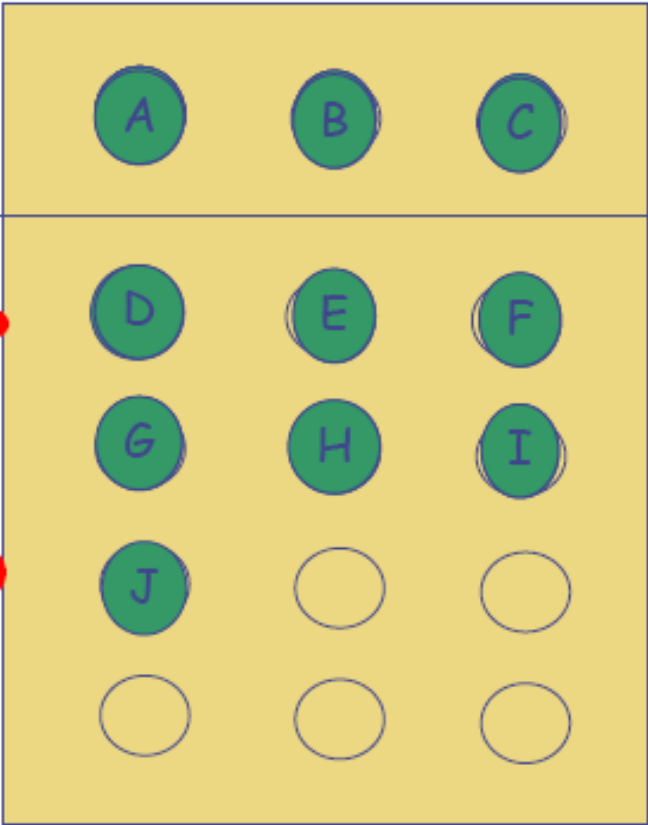
- 对于产品的经营者，他们则更加关注于产品的卖点即一个产品的商业特点；关注于同类产品的优劣性价；是否提供良好的兼容性与可扩展性；是否存在较好的可移植和操作性；是否存在可靠的容错性设计；
- 参考指标：
 1. 响应时间 (Response Time)
 2. 并发用户数 (The Number Of Concurrent Users)
 3. 吞吐量 (Throughput) 近似等于每秒交易次数
 4. 系统资源利用率 (Hardware/Software Resource Utilization)
 5. 可靠性和稳定性 (Reliability or Stability)
 6. 可伸缩性 (Scalability)
 7. 可恢复性 (Recoverability)

开发商视角 (Developer)

- 对于产品开发商，他们则更加关注于产品的需求；用户关心的响应时间；管理员关心的系统可扩展性等；架构设计是否合理；数据库设计是否合理；代码是否存在性能方面的问题；系统中是否有不合理的内存使用方式。
- 性能指标：
 1. 内存消耗 (Memory Expend)
 2. 算法消耗 (Arithmetic Expend)
 3. 数据路径 (Data Route)

理发店内场景的模拟

顾客数	理发师状态	顾客状态
1	有两名理发师空闲	顾客不需要等待
2	有一名理发师空闲	顾客不需要等待
3	没有理发师空闲	没有顾客等待
4	没有理发师空闲	有1名顾客需要等待
9	没有理发师空闲	接近顾客所能忍受最大限度，顾客满意度最高
10	没有理发师空间	必然有一位顾客选择离开

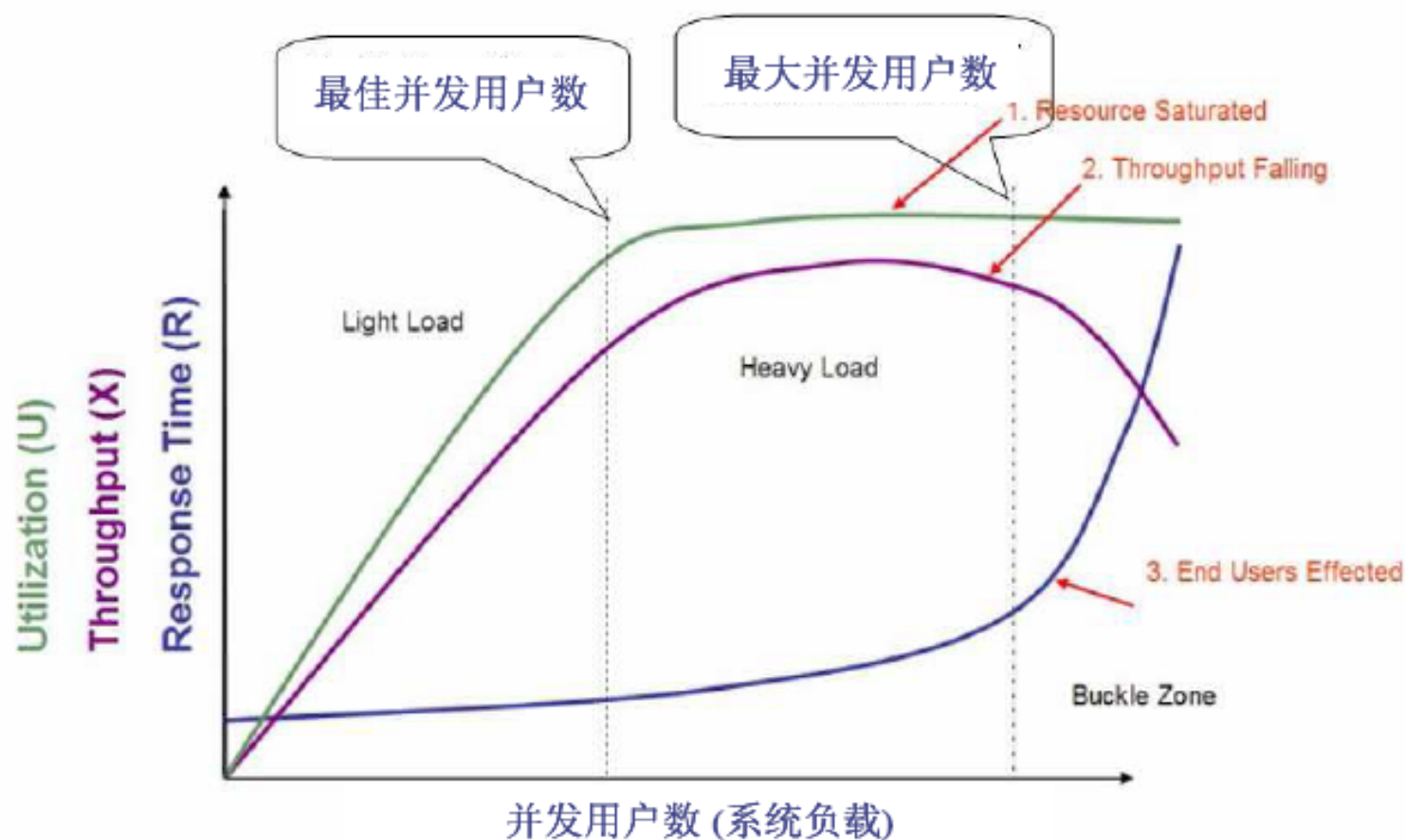


如果持续 每个小时 >3个顾客呢？

“理发店”的性能模型



Sifang Technology
四方科技
让沟通更简单和快捷



能力提升1——性能测试模型

Sifang Technology
四方科技
让沟通更简单和快捷

- 测试的活动应该与软件开发同步进行。
- 测试对象不仅仅是程序，还包括需求和设计。
- 及早发现软件的缺陷可降低软件的开发成本。

LoadRunner性能测试模型



Sifang Technology
四方科技
让沟通更方便和快捷

第一步 规划测试模型

第二步 创建虚拟用户

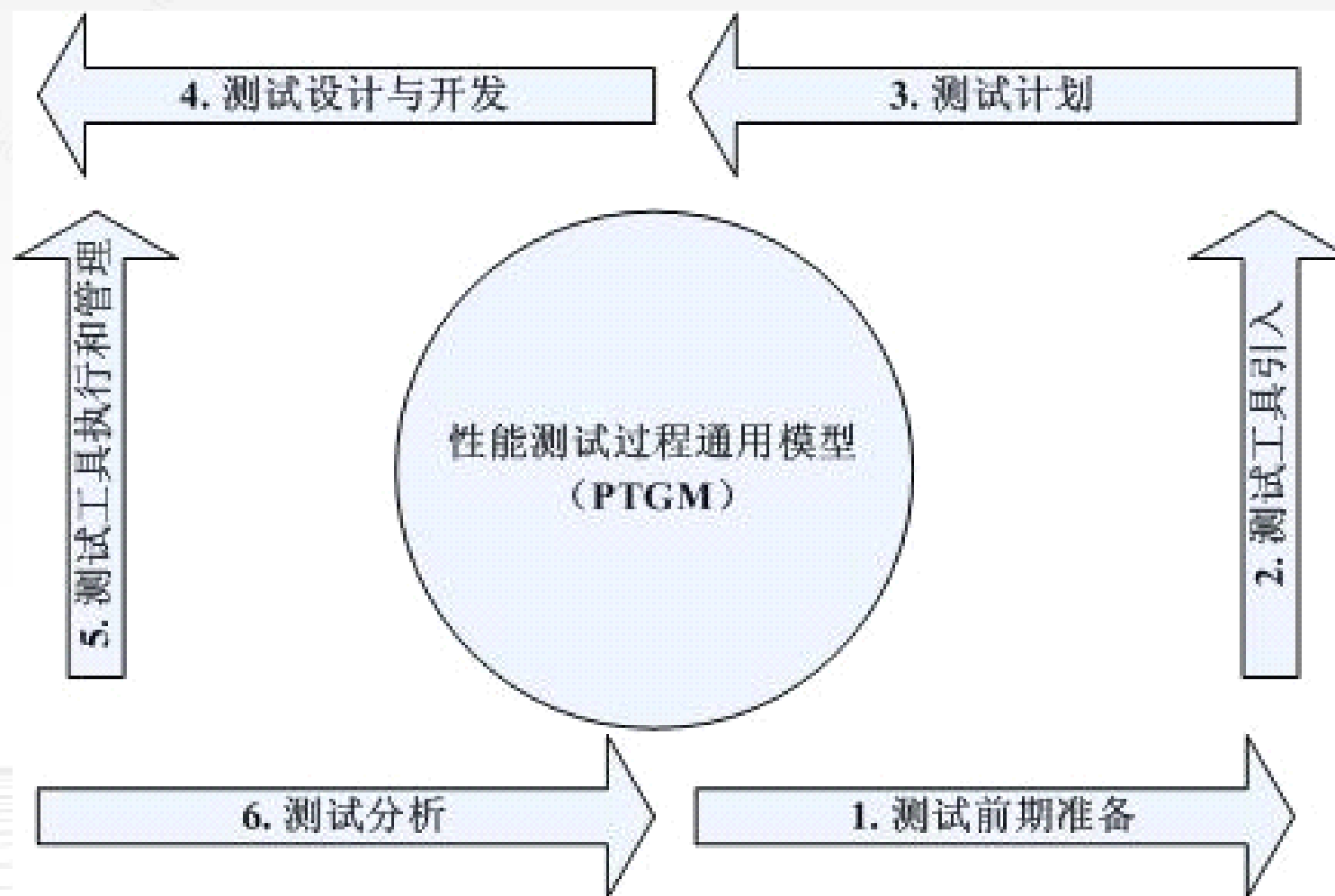
第三步 创建测试方案

第四步 运行测试方案

第五步 监视系统数据

第六步 分析测试结果

PGMT通用性能测试模型



PGMT通用性能测试模型的优势

Sifang Technology
四方科技
通用性能测试

- 为什么要引入PTGM模型？考虑到性能测试工作中自动化的引入和使用，在PTGM模型中我们明确了“测试工具的引入”阶段，用以处理和测试工具引入的相关过程；由于性能测试的需求获取和分析、测试团队组建等工作与功能测试存在不同的侧重点，因此在过程模型中增加了一个独立“测试前期准备”阶段；另外，与功能测试相比，性能测试的测试设计和开发明显会有区别，因此将“测试设计与开发”阶段作为一个过程中的重要阶段。

（有关详细PGMT模型应用，可参考《春秋Q传性能测试规划》）

Let's have a rest
(10 minute)

能力提升2——性能测试方法

Sifang Technology
四方科技
让沟通更方便和快捷

- 性能测试
- 负载测试
- 压力测试
- 可靠性测试
- 并发测试
- 配置测试
- 失效恢复性测试

性能测试 (Performance Testing)



Sifang Technology
四方科技
让沟通更方便和快捷

- Performance Testing 这是一个容易混淆的概念，通常泛指所有的性能测试。本文特指在特定条件下验证性能是否达到预期指标的测试为性能测试。

负载测试 (Load Testing)

- Load Testing 是指模拟真实的用户行为，通过不断加压直到性能出现瓶颈或资源达到饱和。负载测试是我们最经常进行的性能测试，用于测量系统的容量，发现系统瓶颈并配合性能调优。有时候也称为可量性测试 Scalability Testing.
- 测试策略：主要验证服务器在持续加压过程中，各加压点服务器健康状况。目的是找出最佳并发点、最大并发点以及系统所能承受的最大性能瓶颈点，为之后展开全面的压力测试提供数据依据。

压力测试 (Stress Testing)

- Stress Testing 是指测试系统在一定的饱和状态下系统的处理能力。负载测试的不断加压到一定阶段即是压力测试，两者没有明确的界限。压力测试通常设定到CPU使用率达到75%以上，内存使用率达到 70%以上，用于测试系统在压力环境下的稳定性。此处是指过载情况下的稳定性，略微不同于7*24长时间运行的稳定性。
- 测试策略：在负载测试数据基础上，针对系统在特定数据量下负荷压力进行设计。主要验证处于性能转折点时，系统服务器的性能状况。

可靠性测试 (Reliability Testing)



Sifang Technology
四方科技
让沟通更简单和快捷

- Reliability Testing 是指加载一定的业务压力，同时让此压力持续运行一段时间，测试系统是否可以稳定运行. 可以理解为压力测试关注的是过载压力，可靠性测试关注的是持续时间。
- 测试策略：主要验证系统在一定压力环境下，服务器所能承受的最大响应时间是否满足实际项目需要。

并发测试 (Concurrency Testing)

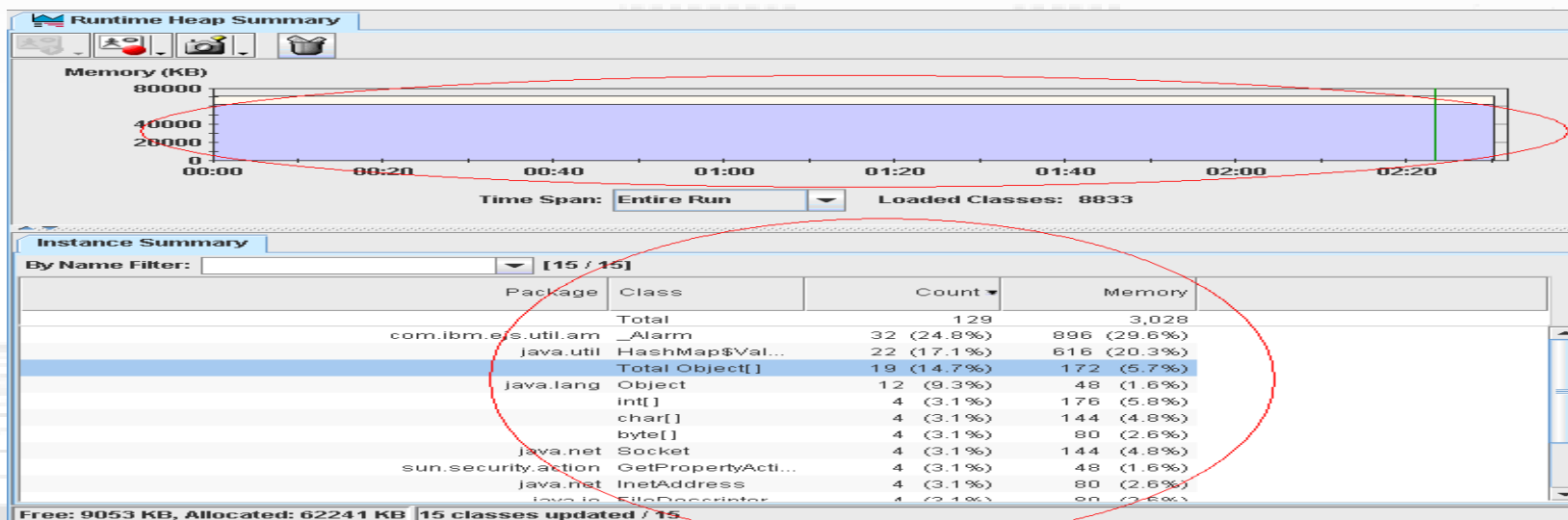


Sifang Technology
四方科技
让沟通更简单和快捷

- Concurrency Testing 是模拟用户并发访问同一应用的测试，用于发现并发问题诸如内存泄漏，线程锁，资源争用，数据库死锁。
- 测试策略：基于负载测试的基础上，针对测试曲线中较为平滑的部分设计测试案例。如：400—800用户区间内，被测试服务器性能相对稳定。设计测试案例，使用第三方软件确定性能问题。

并发测试性能案例

并发人数	性能指标	加压方式	数据量大小
100用户	Cpu<75% Mem<70%	一次性加载400用户，服务器趋于稳定后，每40分钟加载100用户至800用户，服务器持续运行1小时，观测服务器性能健康状况	数据库为2年数据



配置测试 (Configuration Testing)

- Configuration Testing 验证各种配置对系统性能的影响，用于性能调优和规划能力。
- 测试策略：验证系统在不同环境、不同数据量下的服务器性能状况，一般用于系统发生重大需求变更或对底层文件进行大量修改后进行。

失效恢复性测试 (Failover Testing)



Sifang Technology
四方科技
让沟通更方便和快捷

- Failover Testing 针对有冗余备份和负载均衡的系统，检验系统局部故障时用户所受到的影响。
- 测试策略：主要验证系统是否存在故障处理机制，已开发的功能是否满足客户实际需求。

性能测试分析方法

- SEI (Load Testing Planning Process)
- RBI (Rapid Bottleneck Identify)
- Load Runner 性能测试过程
- Segue 性能测试过程
- 性能下降曲线分析法

SEI Load Testing Planning Process



Sifang Technology
四方科技
让沟通更简单和快捷

- SEI Load Testing Planning Process: 是一个关注于负载测试计划的方法，目标是产生“清晰，易理解，可验证的负载测试计划”。区别生产环境和测试环境的不同，分析用户的行为以产生用户和用户场景。
 - SEI方法的特点：
 1. 测试重点更倾向于实际的商业应用
 2. 是一个关注于负载测试的性能模型
 3. 更适用于企业级系统使用该模型
- (有关SEI详细资料，请查阅我的Blog)

RBI (Rapid Bottleneck Identify)

- RBI (Rapid Bottleneck Identify) : 是 Empirix 公司提出的快速识别系统性能瓶颈的方法。
- RBI方法的特点:
 1. 80%的性能问题决定于系统单位时间处理文件的效率（即系统的吞吐量）
 2. 通过并发用户数与吞吐量的关系，确定问题范围
 3. 从网络，数据库，应用服务器和代码本身4个环节确定系统性能的瓶颈。

（有关RBI详细资料，请查阅我的Blog）

Load Runner 性能测试过程

- Load Runner 性能测试过程：计划测试-->测试设计-->创建VU脚本-->创建测试场景-->运行测试场景-->分析结果
 - LoadRunner性能测试的特点：
 1. 存在较的为完善测试流程
 2. 可以较好的对测试数据进行分析
 3. 基于场景的测试案例设计思想
 4. 测试过程可以由设计人员自行控制
- （有关LoadRunner 详细资料，请查阅我的Blog）

Segue 性能测试过程

- Segue 性能测试过程: 基于Web企业级应用状况监控, 提供使用者管理Web应用的性能和可靠性。支持网Web, 无线wireless, 中间件middleware, 数据库database的应用。
- Segue性能测试特点:
 1. 引入了性能基线的概念
 2. 通过取值基线, 设定可接受的性能目标
 3. 使用不同的并发用户数进行 Try-Check的重复测试.

(有关LoadRunner 详细资料, 请查阅我的Blog)

Let's have a rest
(10 minute)

能力提升3——软件架构分析技术



- 软件架构（[software architecture](#)）是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。[软件架构](#)是一个系统的草图。软件架构描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通讯。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在[面向对象](#)领域中，组件之间的连接通常用接口_([计算机科学](#))来实现。[软件体系结构](#)是构建[计算机软件](#)实践的基础。与建筑师设定建筑项目的设计原则和目标，作为绘图员画图的基础一样，一个[软件架构师](#)或者系统[架构师](#)陈述[软件构架](#)以作为满足不同客户[需求](#)的实际系统设计方案的基础。
- 是一般而言，软件系统的架构（Architecture）有两个要素：
 - 它是一个软件系统从整体到部分的最高层次的划分。
一个系统通常是由元件组成的，而这些元件如何形成、相互之间如何发生作用，则是关于这个系统本身结构的重要信息。
详细地说，就是要包括架构元件（Architecture [Component](#)）、联结器（Connector）、任务流（Task-flow）。所谓架构元素，也就是组成系统的核心“砖瓦”，而联结器则描述这些元件之间通讯的路径、通讯的机制、通讯的预期结果，任务流则描述系统如何使用这些元件和联结器完成某一项需求。
·建造一个系统所作出的最高层次的、以后难以更改的，商业的和技术的决定。
在建造一个系统之前会有很多的重要决定需要事先作出，而一旦系统开始进行详细设计甚至建造，这些决定就很难更改甚至无法更改。显然，这样的决定必定是有关系统设计成败的最重要决定，必须经过非常慎重的研究和考察。

AntiSpam架构分析

- 技术架构分析:

- 框架技术: MVC (Struts架构)、Hibernate 技术、Ajax 技术
- Web技术: 静态页面技术、XHTML 技术、HTML 5 技术、Javascript 技术;
- 组件技术: ACE、OCCI、OTL、JDK
- Web容器: Tomcat、Apache;
- 操作系统: linux Suse 64位
- 数据库: Oracle 11G、RAC;
- 中间件技术: WebServices

- 分层架构分析

- 应用层;
- 表示层;
- 会话层;
- 传输层;
- 网络层;
- 数据链路层;
- 物理设备层;

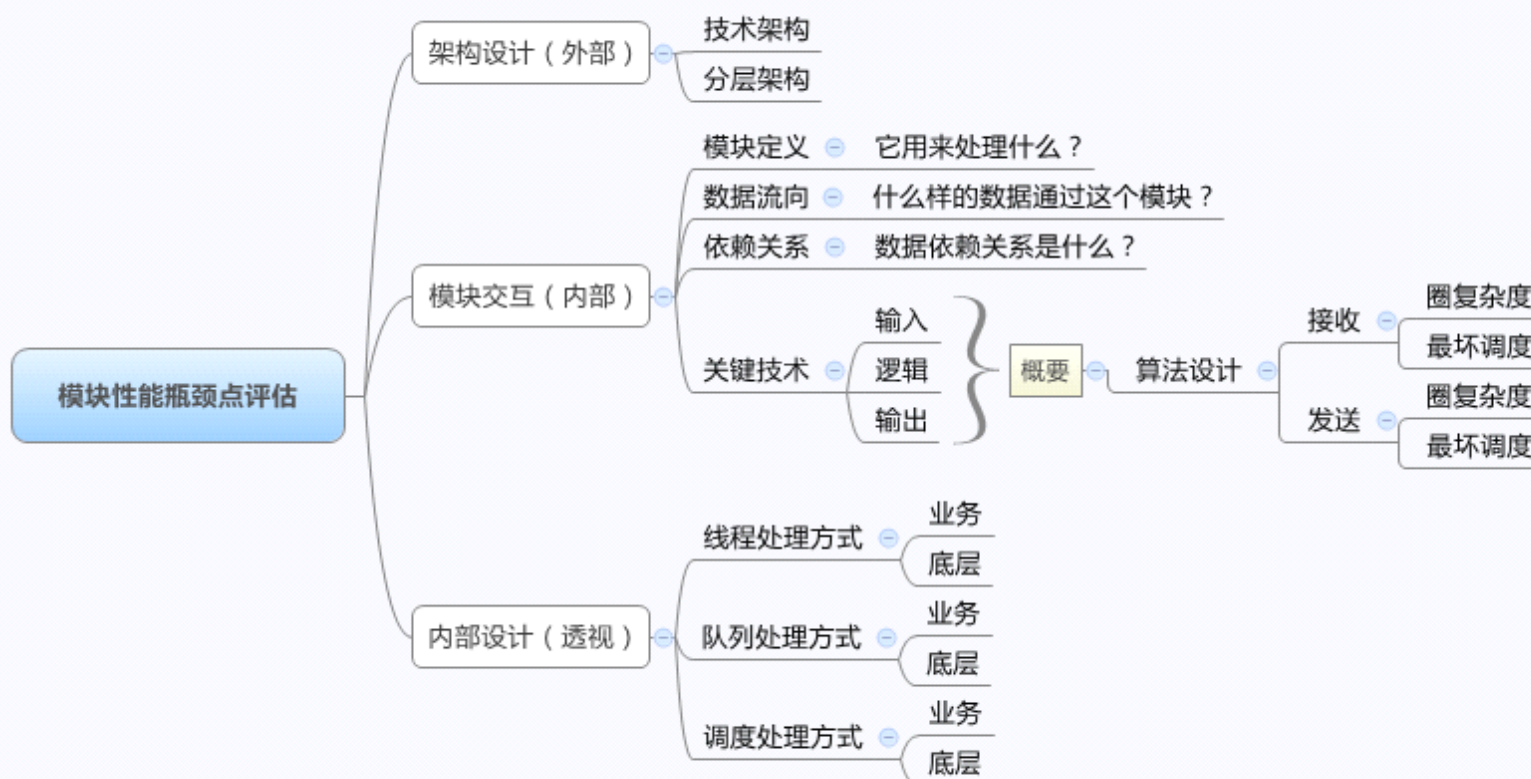
为什么要做架构分析

- 知己知彼百战不殆！！！！
 - 充分了解你的对象，它你能得到什么？
 - 学历、背景、知识、技能、习惯、素养？
-
- OCCI 是什么版本？有什么Bug？如何使用？我们是怎么做的？
 - WebService在架构中的作用是什么？如何传递数据？
 - Hibernate 技术的优势是什么？劣势是什么？哪里是瓶颈点？
 - Oracle的版本？
 - 存储技术？
 - 磁盘划分方式？

Let's have a rest
(10 minute)



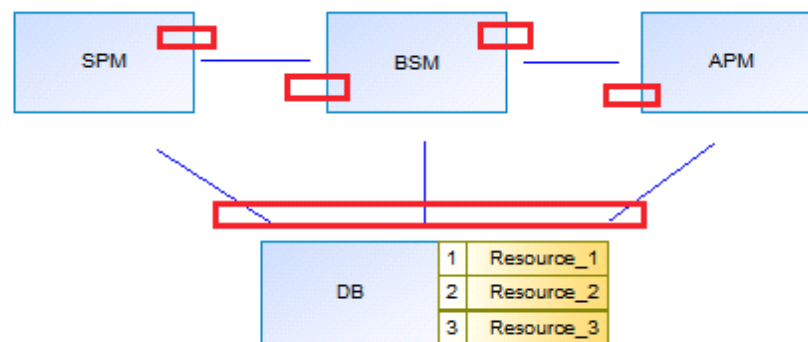
能力提升4——模块性能瓶颈点分析技术



模块交互——挖掘测试点



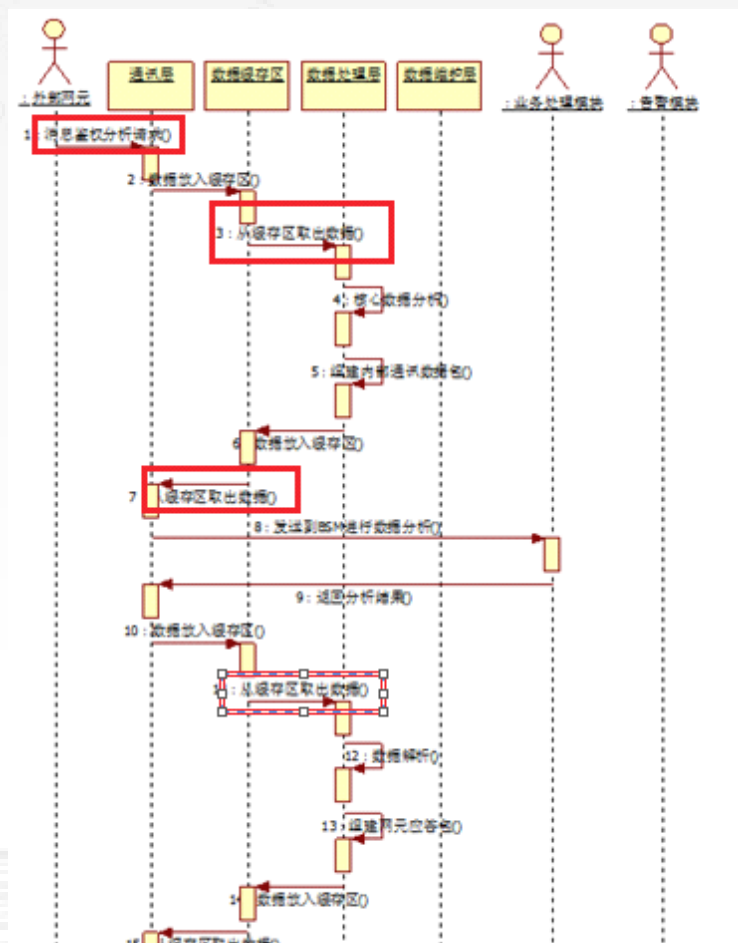
Sifang Technology
四方科技
让沟通更方便和快捷



内部设计——挖掘测试点



Sifang Technology
四方科技
让沟通更方便和快捷



挖掘性能测试点

- 识别哪些是可能存在的瓶颈点？
- 识别线程调度关系？并行 or 串行？哪些业务？
- 为什么队列会满？队列大小是多少？
- 接入算法？逻辑算法？输出算法？

Let's have a rest
(10 minute)

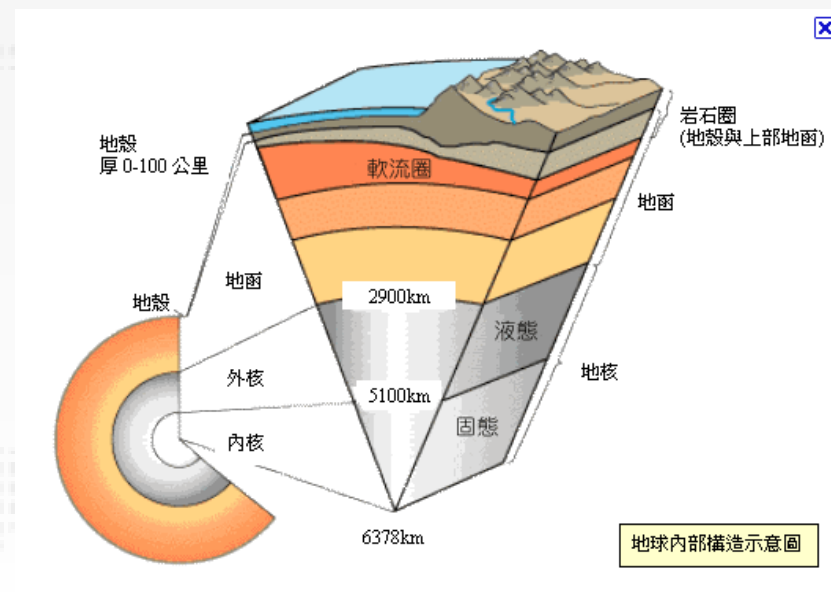
能力提升5——测试场景萃取技术

- 性能设计中的关键章节，本章节描述的内容为测试场景萃取技术。
- 这也是业内争议较大的部分，关于场景萃取技术，我们也能理解为测试目标。
- 定义清晰的测试目标，业内有四种认识：
 - 第一种：根据软件需求规格说明，提取测试场景。
 - 第二种：根据商业客户使用模式，提取测试场景。
 - 第三种：根据软件实际设计能力，提取测试场景。
 - 第四种：根据局点交付实际能力，提取测试场景。
- 应该说这几种方式都有一定的说服力。
- 所以从实际使用的角度来说，能否定义清晰的是决定性能测试是否成功的关键因数。



球形模型切面技术

- 根据右图让我们了解一下什么是球形模型。以及如何应用球形模型划定测试目标。球形模型是一种对象分解技术。将事物比作球体，如何了解事物的核心需要一种从外到内的剥离技术，这种技术被称为球型分解。
- 如何将球形模型应用于测试场景萃取技术呢？让我们先了解，被测试对象分解的步骤：
 - 第一步：需求规格（岩石圈）
 - 第二步：用户场景（地函）
 - 第三步：设计能力（地核）
 - 第四步：交付能力（内核）
- 一个有效的性能目标，需要通过多个维度进行切片。不同层切入代表我们从不同的角度审视产品。



第一步：根据软件需求规格说明，提取测试场景



Sifang Technology
四方科技
让沟通更简单和快捷

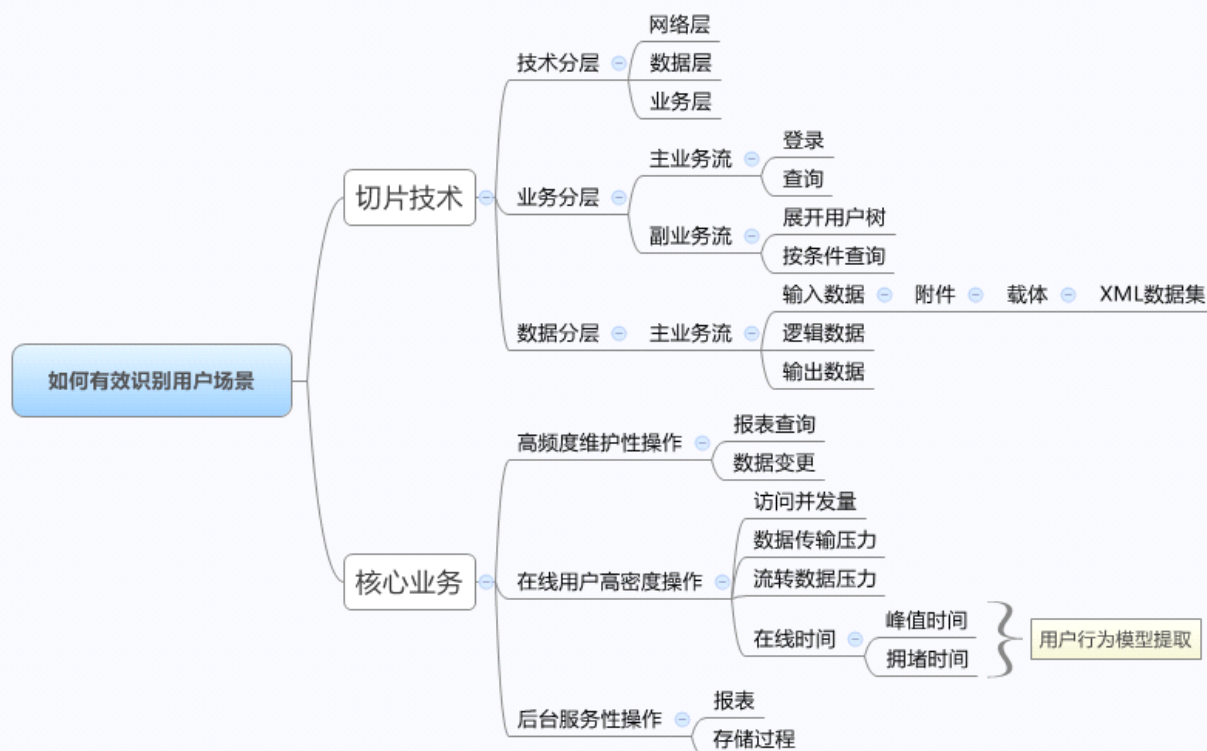
- 根据球形模型分层式剥离技术，这是最基础的也是最重要的一个步骤。识别需求规格说明书中的场景，首先需要了解常用的性能测试方法：
 - 1、负载性测试；
 - 2、可靠性测试；
 - 3、压力性测试；
- 负载性测试：是指模拟真实的用户行为，通过不断加压直到性能出现瓶颈或资源达到饱和。负载测试是我们最经常进行的性能测试，用于测量系统的容量，发现系统瓶颈并配合性能调优。有时候也称为可量性测试 Scalability Testing.
- 可靠性测试：是指加载一定的业务压力，同时让此压力持续运行一段时间，测试系统是否可以稳定运行. 可以理解为压力测试关注的是过载压力，可靠性测试关注的是持续时间。
- 压力性测试：是指测试系统在一定的饱和状态下系统的处理能力。负载测试的不断加压到一定阶段即是压力测试，两者没有明确的界限。压力测试通常设定到CPU使用率达到75%以上，内存使用率达到 70%以上，用于测试系统在压力环境下的稳定性。此处是指过载情况下的稳定性，略微不同于7*24长时间运行的稳定性。

需求规格识别到的性能目标

V100R002C12						
类型	指标项	单位	标配单机设计目标	集群设计目标	设计原则	备注
基本性能指标	支持的主叫用户数	个	500w	4000w	24小时内的活跃用户数	
	支持的被叫用户数	个	500w	4000w	24小时内的活跃用户数	
	消息处理能力	条/秒	6000	60000	集群10台BSM和IPM组网	
	黑名单、嫌疑名单总里	条	100w			
	有效白名单总容量	条	100w			
	关键字分隔符	个	50			
	关键字容量	个	2000			
	嫌疑消息里（非告警消息里）	条/秒	900	9000	垃圾短信按系统消息总数25%*60%（需监控）计算	参考广东版本设计原则：垃圾短信按系统消息总数25%*60%（需监控）计算
	黑/嫌疑告警消息总里	条/秒	60	600	告警入库按流量的1%	广西移动：0.4%；上海联通：0.3%
	拦截消息里	条/秒	60	600	告警入库按流量的1%	
	黑/嫌疑名单动态入库	条/秒	18	100	告警里的30%	广西移动：8%；上海联通：14%
	黑/嫌疑名单动态释放	条/秒	18	100	与入库一致	
	WEB在线客户端数	个	50			
	WEB并发访问客户端数	个	20			
特殊性能指标	缓存消息流量	条/秒	48	480	缓存消息流量为告警消息里的80%	定制开发功能
	缓存用户数里	个/小时	36000	360000	一个小时内缓存用户数：告警数里/10=告警号码数，再乘以3600s	定制开发功能
	用户群个数	个	10	10		
	用户群号码里	个/群	100w			
	SP接入号	个	10w			定制开发功能
	提交审核平台速度	条/秒	18	90	人工仲裁最高处理速度为90条/s、BOSS系统最高处理速度 60条/s	
	审核平台回传处理速度	个	18	90		包含审核加黑和释放
	H2营帐接口速度	条/秒	20	20		定制开发功能（湖南联通）
系统资源占用	CPU	%	<=70%	<=70%		系统稳定阶段的平均取值
	内存	%	<=70%	<=70%		
	磁盘IO	%	<=70%	<=70%		

第二步 如何识别用户场景——提取测试场景

- 用户场景是基于产品交付客户的一种行为识别技术。该技术用于在性能测试规划中，提取有效的测试模型，对被测目标整体能力进行评估。



用户场景提取原则

- 第一，影响终端用户体验；
- 第二，影响产品可维护性；
- 第三，关注80%的系统用户核心使用习惯；

第三步：产品设计能力——提取测试场景

- 什么是产品的设计能力？

- 产品设计能力是指一个产品的原始设计能力。如高铁的设计时速为300公里，而实际运营考虑到各方因数实际运营能力可能只是设计的能力的70%。因为设计能力是按照最大峰值和最大时速考虑，实际运营时一般很难达到设计能力？设计能力的意义在于了解产品的基线，而这个基线并非是产品的最低交付基线。

- 产品发布时，我们到底参考哪个目标？

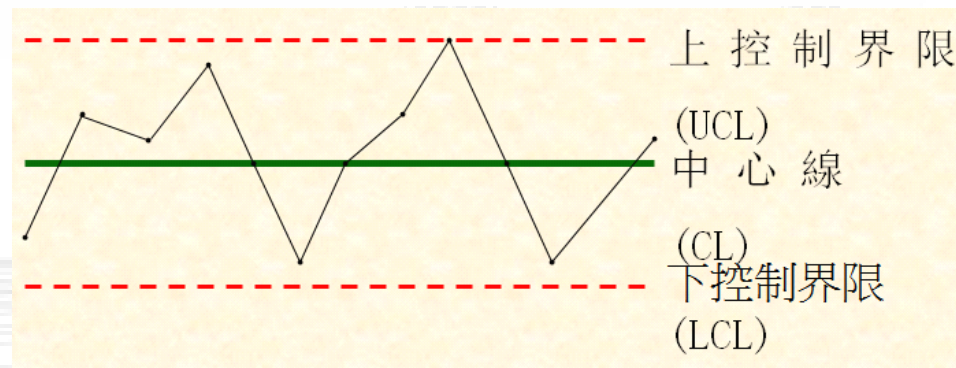
- 产品设计能力表达了设计者对产品整体性能考量。根据木桶原理设计能力是以最低短板为基线的一种衡量方式。也就是说，设计能力不一定代表发布标准，而实际的发布标准与商业目标息息相关。如：国家要求的星火计划、投入运营的京沪高铁等等。

- 什么是性能测试基线？

- 测试基线是一个抽象概念，体现了数据建模思想。性能测试基线决定了产品设计能力，一个产品加入了哪些新的特性？这些特性对原有基线会造成什么样的影响，是需要通过性能基线进行深度分析的。性能基线的意义在于找到一条衡量基准（BaseLine）。

- 性能基线我们可以做哪些事？

- 控制图法
- 散点图法
- 关联图法
- 鱼骨图法



第四步：局点交付能力—提取测试场景

- 收集局点信息、历史版本能力信息、产品能力基线，定义产品最低交付标准。
- BASELINE的意义在于 **了解当下，预测未来**。
- 如何识别和提取局点交付能力？一般有两种情况：
 - 第一种情况：依据产品历史能力定义
 - 第二种情况：依据交付局点接入产品的最大峰值定义

设计原则

- 目标设定不合理，没有抓住产品的特点和交付需求，搞不当前要验证什么？
- 无法确定核心评价指标或评价指标无法反映产品特点？
- 设计过程缺少指导原则或没有指导原则，设计随意，想到哪分解到哪？缺少分析建模。
- 场景被设计出来，却不知道到底验证了什么？场景设计时偏离了测试目标。
- 重要验证点遗漏，架构、业务、接口设计、逻辑算法理解不清。
- 设计时不考虑设计基线，产品缺少牵引目标？

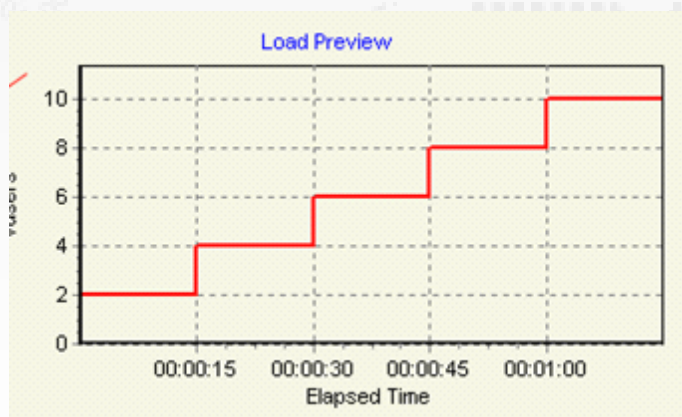
案例分析

- AntiSpam实例分析
- MessagePlus实例分析
- 彩信产品实例分析

Let's have a rest
(10 minute)

能力提升6——测试场景构建技术

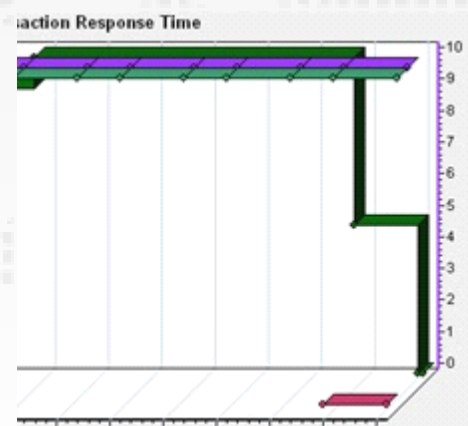
- 在分析Action 构建技术之前，让我们先了解一些常用的技术加压手段：
 - 分布式加压



分布式加压：常用加压方法之一，一般有两种目标方式。一种是加压，一种是减压。如：每 15 分钟增加一些消息量或并发用户。在设计 Action 场景时，我必须考虑每个事物的什么时候聚合用户，什么时候释放用户。这种聚合与释放用于真实再现用户行为模型？由此，你知道提取用户行为模型是多么重要了！

稳定式加压：常用加压方式之一，用于稳定性测试或压力测试。是指系统在短期内加压到一个数量级。在数量级下进行长时间测试。这种测试方法，用于对产品稳定性进行考量。电业行业的标准为：7*24 小时，最小时长为 3*24 小时。

- 稳定式加压：



事物与场景

- **工具的思想？** 这里引用了LoadRunner在场景建模中的一些理念。
 - **什么是场景？**
场景是你的一个测试的工厂，就是要把你的一些事务放到这个里面，通过在这个工厂里进行假设、模拟、分解、实验、总结结果，然后便于分析。
 - **什么是事物？**
事物是场景最小单元。事务是你的一个操作，比如注册、登陆.....就是把操作分解了，这样可以更好的测试每个模块的情况。识别事物就是在一个大的测试场景中抓住测试目标。
 - **事物与场景有何区别？**
事物是场景的一个单元？多个事物构造一个场景。场景=主流+备选流，多个不同的备选流组成不同的场景。也构成了不同的验证目标。
 - **什么是集合点？**
比如订票，每个人都在订票，但是都不提交，你可以在提交之前设置一个集合点，让大家一块提交，这样压力就加大了，可以测试系统的承受能力有多大（能够承受多少人同时干这件事）。

Let's have a rest
(10 minute)

能力提升7——性能常用监控工具

- 系统监控:

- Performer Tools 华为性能模拟桩
- Nmon IBM通用测试套件
- Spliot For XX (Oracle、DB2、System等) Quest公司产品
- Weblogic JVM Console

- 进程监控

- GetCpu.sh 监控cpu、memory、IO、连接数、线程数、句柄
- Monitor.sh 监控iostat、vmstat等

- **C++**内存监控:

- vargrind

- **JVM**内存监控:

- Jconsole
- Jmap
- Jprofile
- Jprobe

- 线程安全检查

- checkfun

- 数据库**SQL**监控

- DBCheck

性能测试工具选择与比较

- CompuWare 公司的QALoad;
- Mercury Interactive公司的LoadRunner;
- Microsoft公司的WAS ;
- RadView公司的WebLoad ;
- IBM公司的RPT / Tivoli;
- 开源工具: OPENSTA、JMeter、Watir;
- 单元性能: Junit、Cunit、BoundsChecker等。

QALoad :

- 1、简介

QALoad模拟成百上千的用户并发执行关键业务而完成对应用程序的测试，并针对所发现问题对系统性能进行优化，确保应用的成功部署。QALoad不单单测试WEB应用，还可以测试一些后台的东西，比如SQL Server等，只要是它支持的协议，都可以测试。

- 2、特征

- 1) 自动捕获实际执行过程，自动生成测试脚本；
- 2) 通过控制台控制各个Agent，进行脚本分配；
- 3) 应用广泛，支持多种协议，如DB2、ODBC、WWW、Winsock等等。

LoadRunner :

- 1、简介

LoadRunner 是一种预测系统行为和性能的负载测试工具。通过模拟上千万用户实施并发负载及实时性能检测来确认和查找问题，能够对整个企业架构进行测试。通过使用Loadrunner，企业能够最大限度的缩短测试时间，优化性能和加速应用系统的发布周期。LoadRunner 能支持广范的协议和技术，功能比较强大，可以为特殊环境提供特殊的解决方案。

- 2、特征

- 1) 轻松创建虚拟用户;
- 2) 创建真实的负载;
- 3) 定位性能问题;
- 4) 分析结果以精确定位问题所在;
- 5) 重复测试保证系统发布的高性能;
- 6) 支持Enterprise Java Beans的测试;
- 7) 支持无线应用协议;
- 8) 支持Media Stream应用。

WAS

- 1、简介:

WAS--Web Application Stress是由微软的网站测试人员所开发，专门用来进行实际网站压力测试的一套工具。透过这套功能强大的压力测试工具，您可以使用少量的Client端计算机仿真大量用户上线对网站服务所可能造成的影响。

- 2、特征:

1) 可以数种不同的方式建立测试指令：包含以手动、录制浏览器操作步骤、或直接录入IIS的记录文件、录入网站的内容及录入其它测试程序的指令等方式；

2) 支持多种客户端接口；

3) 支持多用户：利用多种不同的认证方式仿真实际的情况。

WebLoad :

- 1、简介:

WebLoad是RadView公司推出的一个性能测试和分析工具,它让web应用程序开发者自动执行压力测试;WebLoad通过模拟真实用户的操作,生成压力负载来测试web的性能。

- 2、特征:

1) 用户创建的是基于javascript的测试脚本,称为议程agenda,用它来模拟客户的行为,通过执行该脚本来衡量web应用程序在真实环境下的性能。

2) 如有需要可以在做负载测试的同时,使用服务器监控工具对服务器端的内容进行记录那样使负载测试更加全面。

RPT/ Tivoli

- 1、简介

RPT--Rational Performance Tester是IBM基于Eclipse平台开发出来的最新性能测试解决方案。它可以有效地帮助测试人员和性能工程师验证系统的性能，识别和解决各种性能问题。

- 2、特征

在为性能测试员和性能优化人员提供了前面所提到的各种性能测试能力以外，它还提供了可视化编辑器，一方面可以使新的测试人员可以在无需培训和编程的情况下，即可快速上手完成性能测试；另一方面，也为需要高级分析和自定义选项的专家级测试人员，提供了对丰富的测试信息的访问和定制能力、自定义 Java 代码插入执行能力、自动检测和处理可变数据的能力。

OPENSTA

- 1、简介

OpenSTA-- Open System Testing Architecture是专用于B/S结构的、免费的性能测试工具，可以模拟很多用户来访问需要测试的网站，它是一个功能强大、自定义设置功能完备的软件。

- 2、特征

- 1) 免费、源代码开放；

- 2) 自定义功能设置大部分需要通过Script来完成，因此，在真正的使用这个软件之前，必须学习好它的Script编写。

选择还是开发一种性能测试工具

成都四方信息科技
成都四方信息

需要注意的几点：

- 工具是否支持被测系统运行的平台（软硬件环境、数据库环境）；
- 工具能否支持被测系统使用的协议；
- 工具是否能够支持我们的特殊要求，如防火墙、负载均衡等；
- 工具是否能够提供对我们关心的服务器类型计数器的监控；
- 工具的价格。