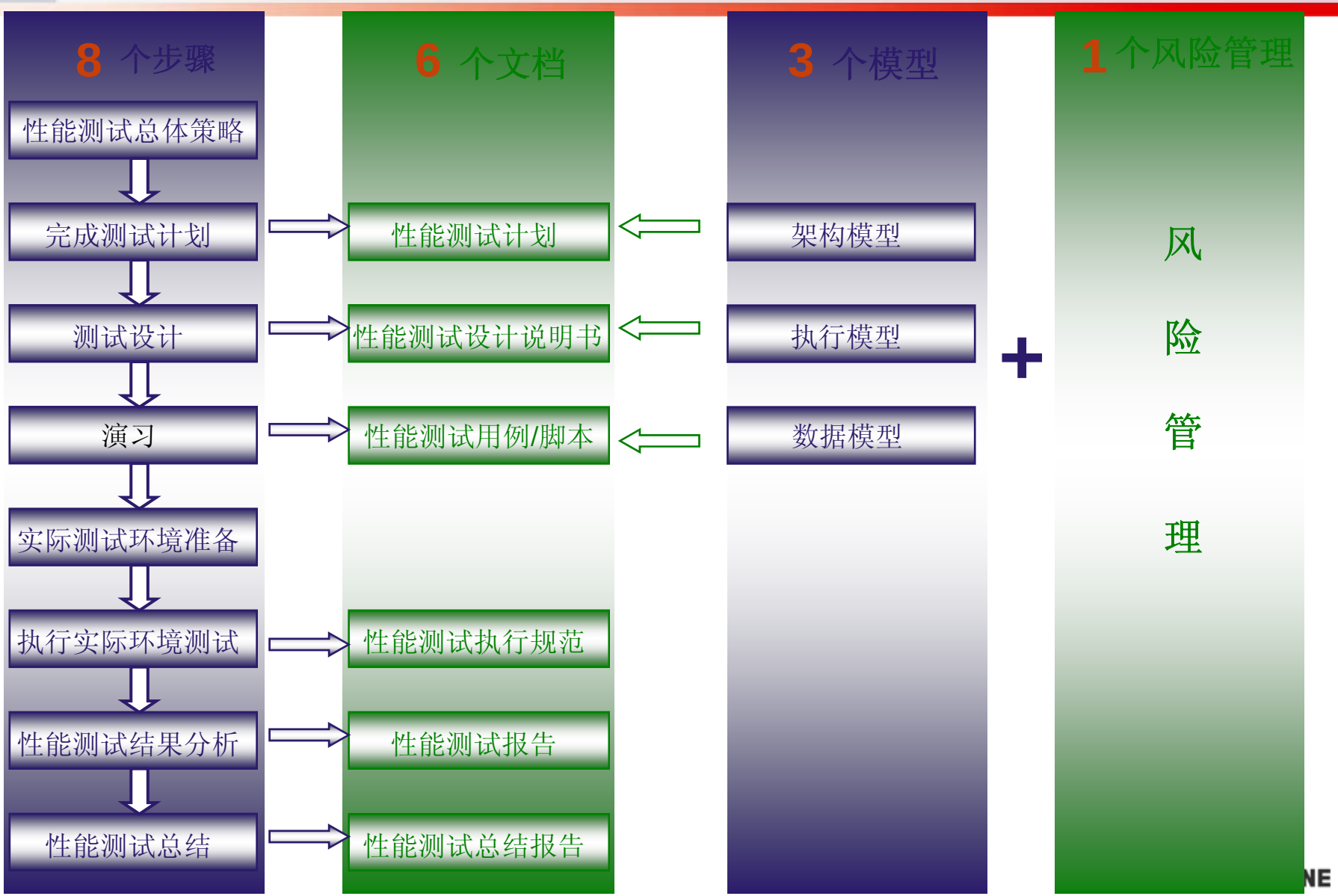
性能测试过程

2009年8月6日

性能测试流程——863+1


8 个步骤
性能测试总体策略
完成测试计划
测试设计
演习
实际测试环境准备
执行实际环境测试
性能测试结果分析
性能测试总结
6 个文档
性能测试计划
性能测试设计说明书
性能测试用例/脚本
性能测试执行规范
性能测试报告
性能测试总结报告
3 个模型
架构模型
执行模型
数据模型
+
1 个风险管理
风险管理

第一步：性能总体分析策划

1. 确定测试目标

- 预期性能指标测试；
- 并发性能测试
- 疲劳性/大数据量测试
- 服务器性能测试
- *网络性能测试*

制定性能测试目标原则：

- ☑ “以需求为本”
- ☑ 测试目标确定的经济性考虑
- ☑ 基于风险的测试目标确定

思考：如何确定目标？

第一步：分析性能测试总体策划

2. 初步确定性能指标

- 并发用户数
- 交易响应时间
- 每分钟交易数

第一步：分析性能测试总体策划

- 具体的指标可划分为：
- 总体指标
  - 整个系统反映出来的整体性能情况
- 网络指标
  - 网络因素对系统的影响情况
- 服务器指标
  - 服务器需要监控的资源
  - 服务器操作系统资源监控指标
  - 数据库资源监控指标
  - Web服务器监控指标
  - 应用服务器监控指标

性能指标选择原则：

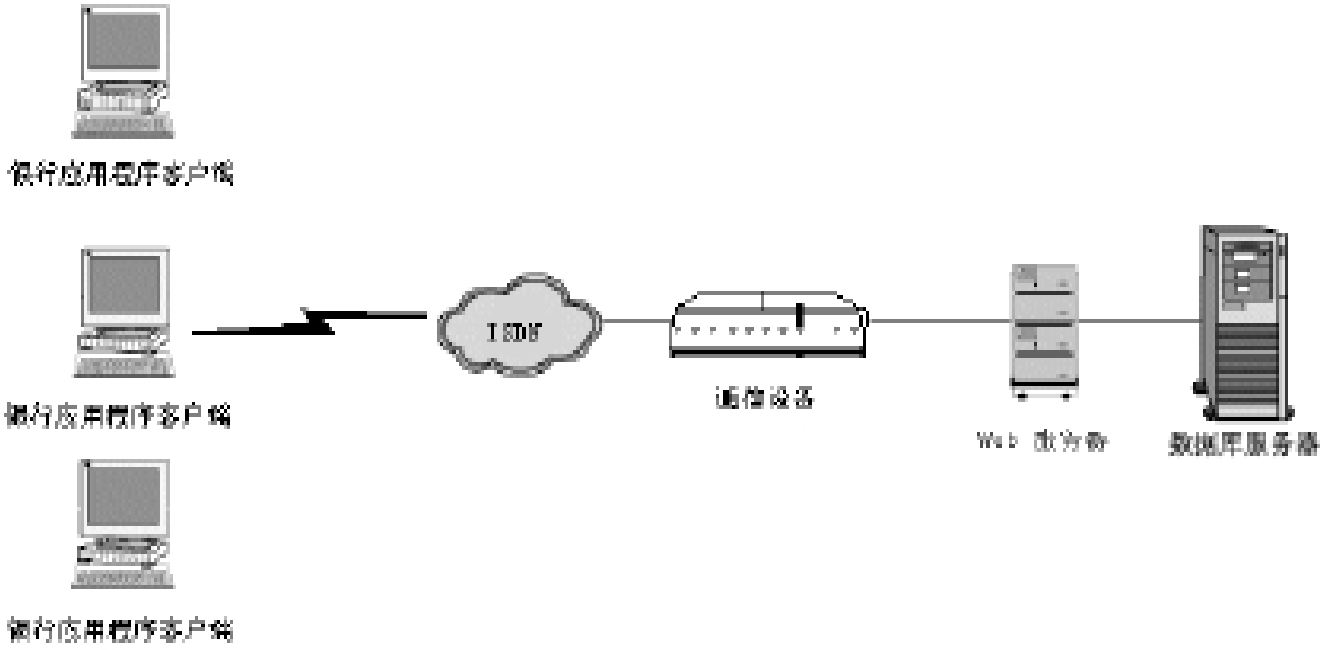
•1、根据系统结构、性能测试目标选择

•2、选择有针对性的指标

第一步：分析性能测试总体策划

3. 架构模型

- 网络拓扑结构：



第一步：分析性能测试总体策划

- 常见协议：
  - **Client/Server: MS SQL, ODBC, Oracle (2-tier), DB2 CLI, Sybase Ctlib, Sybase Dblib,infomix，Windows Sockets及DNS**
  - **定制: C templates, Visual Basic templates, Java templates, Javascript 及 VBscript.**
  - **分布式组件: COM/DCOM, Corba-Java, 及Rmi -Java.**
  - **E-business: FTP, LDAP, Palm, SOAP，Web (HTTP/HTML)**
  - **Enterprise Java Beans: EJB Testing及Rmi-Java.**
  - **Legacy: Terminal Emulation (RTE).**
  - **Mailing Services: Internet Messaging (IMAP), MS Exchange (MAPI), POP3, 及SMTP.**
  - **Middleware: Jacada及Tuxedo (6, 7).**
  - **Streaming: MediaPlayer及RealPlayer.**
  - **Wireless: i-Mode, VoiceXML, 及WAP**

第一步：分析性能测试总体策划

- 软件环境：
  - **OS**
  - **DB**
  - **Middleware**
  - **Others**

第一步：分析性能测试总体策划

4. 风险管理

- 风险识别：识别出本次性能测试的风险（如技术风险、性能测试重点、难点以及其他方面的风险）；
- 定性风险分析：根据风险发生的概率，风险一旦发生对项目或本次性能测试的影响程度，对已识别的风险进行优先排序；
- 定量风险分析：对定性风险分析中排序在先的风险进行分析，并就风险分配一个值；
- 风险应对：增加积极风险，降低消极风险而制定的应对策率；
- 风险监控：在测试执行过程中，保持对项目进行监督以寻找新风险以及变化的风险；

第一步：分析性能测试总体策划

5. 资源规划：

- 估算本次测试所需的人力资源；
- 估算本次测试所需的设备资源、软件、工具资源及支持资源；
- 确定人员沟通管理办法（何时，由谁负责与谁进行沟通，沟通目标及效果），设备资源在何时到位，设备资源管理办法；

6. 进度规划：

- 列出本次性能测试所需的步骤（共**8**步）以及步骤执行顺序；
- 规划每个步骤所需要的资源；
- 根据每个步骤地持续时间，制定进度表；

第二步：确定人员，完成计划编制

1. 确定服务目标

- 制定性能测试开始标准以及测试结束标准，并得到项目组的认可；
- 确定性能测试范围，并得到项目组的认可；

2. 确定参与人员及职责

- 确定测试角色：根据本次性能测试的范围，确定测试角色（如，测试负责人、测试设计工程师、测试执行工程师等等）；
- 确定职责：为角色分配人员，并制定人员职责分配表；
- 确定人员到位时间，规划如何建设、管理测试团队；

3. 完成计划

- 根据“863+1”原则，整合第一步、第二步内容，完成本次性能测试计划的编制----《性能测试计划模版》见附件；

第三步：测试设计

1. 测试环境设计

- 网络环境
- 软硬件环境
- 环境的维护方案
- 时间同步问题
- “镜像”环境

“真实环境”与“测试环境”
对于性能测试有何区别？

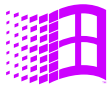
第三步：测试设计

2. 基础数据设计
   - 数据库数据
   - 方便识别和观察的数据
   - 有特殊含义的数据
   - 数据维护方案
3. 业务/交易数据设计
   - 用户数据设计
   - 业务数据设计
   - 负载数据设计
   - 测试过程中的数据验证

第三步：测试设计

4. 执行模型：用例和场景设计

- 对业务的分析和分解
- 根据业务确定用例
- 不同用例按照不同发生比例组成场景
- 了解每个场景的实际意义

 用例与场景示例：

- 背景：一个进销存系统，包括登录、货物入库、订单处理、货物出库、查询五个模块
  - 用例设计：针对模块设计用例
- 场景设计：
  - 场景1：10％登录，10％入库，30％订单，20％出库，30％查询（1000用户）——日常
  - 场景2：10％登录，90％查询（400用户）——周末盘点

第三步：测试设计

5. 监控资源设计：

- **CPU** 利用率；
- **Mem** 使用情况；
- **Disk I/O** 状况；
- 数据库监控；
- **JVM** 使用状况监控；

6. 完成《性能测试设计》----《性能测试设计模版》见附件；
7. 修改执行策略、进度调整，更新测试计划；

第四步：演习

1. 测试工具准备：
   - 检查工具，确保工具能够正常运行；
2. 脚本开发 ：
   - 使用计划中确定的测试工具生成脚本（本人建议使用**loadrunner**） 脚本数据准备
3. 脚本调试 ：
   - 确保涉及的交易能够正常跑通，相关的参数能够收集到。
4. 数据模型 ：
   - 包括基础数据、交易数据（日志分析，业务预测；时间分布，业务高峰，交易类型表）
5. 编制《性能测试用例》（或性能测试脚本）

此块可以在模拟环境进行，但要注意环境架构要
和真实环境尽量一致，保证脚本可以在真实环境下运行等。

第五步：实际测试环境准备

1. 真实环境搭建；
2. 数据录入；
3. 冒烟测试；
   - 执行系统最基本的功能，使性能测试满足测试开始标准，确保性能测试能够顺利地执行
4. 测试场景配置、测试；
5. 性能数据采集；
6. 分析准备；
7. 制定《性能测试执行规范》--- 《性能测试执行规范模版》见附件
   - 规定脚本 执行的规范，以及脚本、基础数据、交易数据的存放位置及版本控制

真实环境测试前的准备，如果在模拟环境下的测试准备顺利，这里的工作就比较轻松了。

第六步：执行实际环境测试

1. 按照测试计划、测试设计执行脚本过程/记录；
2. 执行过程要符合《性能测试执行规范 》；
3. 问题报告；
4. 在执行中发现测试设计缺陷，则要更新测试设计及测试计划；

第七步：性能测试结果分析

1. 测试工具生成的结果：

- 平均事务响应时间图
- 用户图
- 网络监视器图
  - 如果服务器耗时过长，请使用相应的服务器图确定有问题的服务器度量 并查明服务器性能下降的原因。如果网络耗时过长，请使用“网络监视器”图确定导致性能瓶颈的网络问题
- 事务摘要图
  - 细分事务并分析每个页面组件的性能。查看过长的事务响应时间是由哪些页面组件引起的？问题是否与网络或服务器有关？

第七步：性能测试结果分析

- **Web请求指标**
  - 每秒点击次数图
  - 点击次数摘要图
  - 吞吐量图
  - 吞吐量摘要图
  - **HTTP** 状态代码摘要图
  - 每秒**HTTP** 响应数图
  - 每秒下载页面数图
  - 每秒重试次数图
  - 重试次数摘要图
  - 连接数图
  - 每秒连接数图

第七步：性能测试结果分析

- **Web页面组件指标**
  - 激活网页细分图
  - 页面组件细分图
  - 页面组件细分（随时间变化）图
  - 页面下载时间细分图
  - 页面下载时间细分（随时间变化）图
  - 第一次缓冲细分时间图
  - 第一次缓冲时间细分（随时间变化）图
  - 已下载组件大小图

第七步：性能测试结果分析

2. 分析问题产生的原因，性能问题通常来自于：

- 内存资源成为系统性能的瓶颈；
- CPU资源成为系统性能的瓶颈；
- I/O资源成为系统性能的瓶颈；

3. 提交《性能测试报告》

第七步：性能测试结果分析

♣性能测试经验之谈：

- 如果交易的响应时间如果很长，远远超过系统性能需求，且数据库服务器的CPU占用率或磁盘交换率很高，表明数据库设计存在缺陷或SQL语句需要优化，可考虑是否有索引以及索引建立的是否合理；尽量使用简单的表联接；水平分割大表格等方法来降低该值。
- 在百分之二十的负载测试中，您会发现 Web 和 Web 应用程序服务器是出现性能瓶颈的原因。瓶颈通常是由服务器配置不当和资源不足造成的。例如，有问题的代码和 DLL 可能会使用几乎所有的计算机处理器时间 (CPU) 并且会在服务器上造成瓶颈。同样，物理内存容量限制和服务器内存管理不当很容易导致服务器瓶颈。因此，建议您在调查 Web 或 Web 应用程序服务器性能较低的其他可能原因前，先检查服务器的 CPU 和物理内存。

第八步：性能测试总结

1. 本次性能测试活动的总结；
2. 执行与计划的偏差，以及偏差原因；
3. 缺陷统计及分析；
4. 发布前软件版本的质量综合评价；
5. 优化建议；
6. 编写《性能测试总结》----- 《性能测试总结模版》见附件；

讨论

性能测试培训下节课：性能测试工具的使用