{虚拟宠物医院学习系统} 性能测试计划

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | V0.1 |
| 作 者： | 李可欣 |
| 完成日期： | 2018/3/15 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V0.1 | 李可欣 |  | 2018/3/15-2018/3/15 |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 项目概要介绍 1](#_Toc508883717)

[1.1 项目简介 1](#_Toc508883718)

[1.2 项目成员 1](#_Toc508883719)

[1.3 测试范围 1](#_Toc508883720)

[1.3.1 范围内 1](#_Toc508883721)

[1.3.2 范围外 1](#_Toc508883722)

[1.4 前提假设 1](#_Toc508883723)

[1.5 测试目标 2](#_Toc508883724)

[2 性能测试策略 2](#_Toc508883725)

[2.1 性能测试模型 2](#_Toc508883726)

[2.2 性能测试场景 3](#_Toc508883727)

[2.3 重点测试策略 4](#_Toc508883728)

[2.3.1 重点测试原则 4](#_Toc508883729)

[2.3.2 重点测试交易 4](#_Toc508883730)

[3 测试案例设计 4](#_Toc508883731)

[3.1 生产压力分析 4](#_Toc508883732)

[3.2 场景通过标准 4](#_Toc508883733)

[3.3 测试场景设计 5](#_Toc508883734)

[3.3.1 独立场景 5](#_Toc508883735)

[3.3.2 混合场景 5](#_Toc508883736)

[3.3.3 峰值场景 5](#_Toc508883737)

[3.3.4 容量场景 6](#_Toc508883738)

[3.3.5 疲劳场景 7](#_Toc508883739)

[3.3.6 批处理场景 8](#_Toc508883740)

[3.3.7 极限测试 8](#_Toc508883741)

[3.3.8 异常场景 8](#_Toc508883742)

[4 测试实施安排 8](#_Toc508883743)

[4.1 测试进度 8](#_Toc508883744)

[4.2 测试流程 8](#_Toc508883745)

[4.3 测试报告需求 8](#_Toc508883746)

[4.4 性能缺陷管理 8](#_Toc508883747)

[5 性能测试标准 8](#_Toc508883748)

[5.1 启动标准 8](#_Toc508883749)

[5.2 中止标准 8](#_Toc508883750)

[5.3 通过标准 9](#_Toc508883751)

[6 测试环境规划 9](#_Toc508883752)

[6.1 部署环境 9](#_Toc508883753)

[6.2 执行环境 9](#_Toc508883754)

[7 测试风险分析 9](#_Toc508883755)

[8 角色与职责 9](#_Toc508883756)

# 项目概要介绍

## 项目简介

宠物医院近年来在国内逐步兴起。缺乏符合资质的宠物医生是宠物医疗产业的短板。然而，由于过强的分散性和地域经济相关性使得在各地建造实体宠物医生教学和培训机构并不现实。因此，基于互联网的宠物医疗方案应用而生，催生了宠物医院的信息化建设和相关教学培训的大规模发展。

虚拟宠物医院学习系统是一个虚拟宠物医院教学软件，可以使得宠物工作者不去实体医院就能系统地学习各种宠物诊疗专业知识。该软件主要针对相关专业毕业实习医生，能够使得毕业实习生了解宠物医院结构、科室、进行病例学习等。本软件设置不同岗位角色并配备约200个左右的真实病例。使用者可以通过选择如前台、医助等不同角色进行在线学习及考核等。通过宠物医院虚拟学习系统，宠物工作者能够充分熟悉宠物医院的工作环境、岗位责任及工作流程等，积累临床经验，为成为一名合格的宠物医生和建设符合资质的宠物医院奠定基础。

## 项目成员

李可欣、李俊雅、邵腾、乔智劼

## 测试范围

### 范围内

前台系统包括注册登录、角色扮演、职能学习和在线测试四个模块。初步判断，前台系统出现性能缺陷模块为职能学习和在线测试两个频繁使用，业务处理量较大的模块。为保证测试进度，本次测试的范围为职能学习和在线测试两个模块。

### 范围外

为保证测试进度，注册登录和角色扮演两个模块不在本次性能测试的范围内。

## 前提假设

1、系统功能无缺陷

2、接口确定，接口稳定

## 测试目标

1) 系统在单步操作响应时间方面，能满足用户当前及未来1到3年的发展需求；

2) 系统的批量作业运行稳定，处理能力能满足用户当前及未来1到3年的发展需求；

3) 在业务处理能力方面，能满足当前及未来1到3年的业务增长需求；

4) 发现并解决宕机，内存泄漏等严重问题，使系统具备良好的稳定性、健壮性；

5) 系统在资源使用方面比较合理，各项资源平均利用率在30%左右。

# 性能测试策略

与其它测试类型一样，性能测试周期分为：性能测试需求分析、性能测试设计、性能测试实现和性能测试报告四个存在迭代的阶段。其中，性能测试需求和性能测试设计由测试人员手工完成，而性能测试实现需借助性能测试工具达成，通过对性能测试报告的分析判定系统是否存在性能缺陷。

## 性能测试模型

性能测试模型定义了性能测试涉及的各个待测方面，是性能测试设计的指南。



Figure 性能测试模型

根据Figure 1所示的性能测试模型，可以设计如Table 1所示的性能测试类型

Table 1性能测试类型

|  |  |  |  |
| --- | --- | --- | --- |
| **测试种类** | **测试方法** | **测试内容** | **场景类型** |
| 性能测试 | 以系统最大并发用户数的1-2倍作为上限对关键交易进行梯度加压测试，在压力时间内通过的交易量应接近峰值时段的交易量，甚至超过系统全天的交易量 | 注册/登录、角色扮演、职能学习、在线测试 | 独立场景 |
| 压力测试 | 以系统预期最大并发用户数的1-2倍作为上限对常用/关键交易进行混合场景梯度加压测试，在压力时间内通过的交易量应接近峰值时段的交易量，甚至超过系统全天的交易量 | 角色扮演，职能学习 | 混合场景 |
| 峰值测试 | 按照系统峰值时预期最大并发用户数的2-3倍对峰值时段涉及的核心交易进行的混合场景梯度加压测试，在测试时间内通过的交易数量应接近或超过系统全天的交易量 | 职能学习，在线测试 | 峰值场景 |
| 容量测试 | 通过阶段性不断增加系统并发用户的梯度加压方式，测试出系统对核心交易的最大并发处理能力以及系统的最大业务处理能力，测试时最大并发用户数为系统预期最大并发用户数的3-5倍 | 职能学习，在线测试 | 容量场景 |
| 疲劳测试 | 以系统预期最大并发用户数的1-2倍作为上限对关键交易进行持续加压的稳定性与健壮性测试，在压力时间内通过的交易量应为全天交易量3倍以上 | 职能学习，在线测试 | 疲劳场景 |
| 批量测试 | 采用手工/自动调用的方式对后台作业进行测试 | 职能学习，在线测试 | 批处理场景 |
| 极限测试 | 以超过系统处理能力的压力对系统进行不断加压的极限状态测试，测试系统在极限压力下处理能力是否稳定以及是否出现宕机等严重性能问题 | 职能学习，在线测试 | 极限场景 |
| 异常测试 | 测试在正常生产压力下的各种异常场景，例如功能上有互斥关系或者有锁机制的场景、网络闪断、数据库主机切到备机、数据库执行备份操作等场景，重点检查异常场景发生时是否会导致系统出现响应异常等各类性能问题 | 职能学习，在线测试 | 异常场景 |

## 性能测试场景

根据项目实际，确定本次性能测试场景，例如可包含以下场景的性能测试：

独立场景

混合场景

峰值场景

容量场景

疲劳场景

## 重点测试策略

### 重点测试原则

为了全面评估系统性能，本次重点测试策略安排如下：

1. 对关键交易开展全面的独立场景测试；
2. 对关键交易开展全面的峰值、容量测试；
3. 多渠道发起以关键/常用交易为主的疲劳测试；
4. 在集成性能测试方面，对交易量大、容易出现性能瓶颈的模块，重点进行测试并安排专人在系统发布前一直跟进开发组的需求变更以便进行及时的回归性能测试。
5. 在渠道性能测试方面，对渠道场景按照业务量及重要性来划分优先级，按照优先级先后来执行，保证系统上线后关键业务的稳定性。

### 重点测试交易

1. 对注册/登录、角色扮演、职能学习、在线测试模块进行全面的独立场景测试
2. 对职能学习、在线测试进行全面的峰值、容量测试
3. 多渠道发起职能学习、在线测试为主的疲劳测试

# 测试案例设计

## 生产压力分析

## 场景通过标准

Table 场景通过标准

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **场景类型** | **单步操作响应时间** | **加压时间** | **操作账户数据** | **处理业务笔数** | **事务成功率** | **各后台利用率** | **并发/在线上限** | **在线思考时间设置** |
| 独立场景 | <1秒 | 30分钟 | >1万 | >5000 | 100% | <50% | 100 | 无 |
| 混合场景 | <1秒 | 1小时 | >1万 | >10万 | 100% | <80% | 200 | 无 |
| 峰值场景 | <2秒 | 1小时 | >1万 | >10万 | 100% | <80% | 200 | 无 |
| 容量场景 | <3秒 | 4小时 | >1万 | >50万 | 100% | <80% | 300 | 无 |
| 疲劳场景 | <1秒 | 12小时 | >1万 | >100万 | 100% | <50% | 100 | 无 |

## 测试场景设计

### 独立场景

对于关键场景，将分别对其采用渐进式加压的方式来进行独立场景测试。每个场景测试多组并发、并发数从1逐步增加到200。对于具体的场景，测试几组并发依据案例执行。案例需要按照业务量设计。对于使用频度较低的场景，可以采用独立/混合场景的方式进行测试，即在背景压力下以小并发的方式来进行长时间疲劳测试。

Table 独立场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 1 | <1秒 | N/A | 3分钟 | 直接加压 |
| 10 | <1秒 | N/A | 10分钟 | 1个用户/5秒 |
| 20 | <1秒 | N/A | 10分钟 | 1个用户/5秒 |
| 50 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/5秒 |
| 100 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/5秒 |
| 200 | <1秒 | >100笔/秒 | 10分钟 | 1个用户/1秒 |

### 混合场景

对于相关联的关键交易，对其采用渐进式加压的方式来进行混合场景测试，以测试这些关联交易是否存在性能问题。每个场景测试多组并发，并发数从20逐步增加到200。对于具体的场景，测试几组并发依据案例来执行（案例主要依据业务量来设计）。对于使用频度较低的场景，可以采用独立/混合场景的方式进行测试，即在背景压力下以小并发的方式来进行长时间疲劳测试。

Table 混合场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 20 | <1秒 | N/A | 10分钟 | 1个用户/5秒 |
| 50 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/5秒 |
| 100 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/5秒 |
| 200 | <1秒 | >100笔/秒 | 10分钟 | 1个用户/1秒 |

### 峰值场景

通过分析生成数据，多数系统日间各个峰值共有两个时段，上午9:00-12:00的3个小时；下午13:00-17:00点的4个小时，因此，峰值最长持续时间4-5小时。考虑到系统的扩展性及实际情况，峰值测试共测试三组：第一组测试正常峰值压力的处理能力；第二组测试2倍、4倍峰值压力下的系统处理能力，以满足未来的业务发展需求，测试时各组中的交易按照业务比例同时加载。

第一组：先用生成峰值时压力的70%加压1小时作为基准压力，之后采用当前生产上的实际峰值压力并发进行加压。

Table 峰值场景设计1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力的70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/5秒 |
| 生产压力 | <1秒 | >100笔/秒 | 4小时 | 1个用户/5秒 |

第二、三组：分别采用生产压力的2到4倍来进行加压，第二组达标后再进行后面的测试。

Table 峰值测试设计2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力的2倍 | <1秒 | >100笔/秒 | 5小时 | 1个用户/5秒 |
| 生产压力的4倍 | <1秒 | >100笔/秒 | 5小时 | 1个用户/5秒 |

对于性能较好的系统，可以直接加压到最大并发数，持续加压5小时来测试系统的峰值处理能力。

### 容量场景

容量测试从生产压力作为基准压力来加压，逐步加压到生产压力的5-10倍，在这个过程中得出如下结果：

1. 最佳处理能力/最佳并发数：最佳处理能力是指不断增大压力的过程中，在业务处理能力与交易响应时间等性能指标满足用户性能需求的前提下，系统所能达到的最大处理能力，系统初始达到最佳处理能力时对应的并发数可以称为最佳并发数。可以看出，在满足用户性能需求这一原则下，最佳处理能力往往不是最大处理能力，因此，对于在很小压力下，业务处理能力或响应时间已经不满足用户性能需求的系统，不讨论最佳处理能力与最佳并发数两个性能指标。
2. 最大处理能力/最大并发数：最大处理能力是指系统系统所能支撑的最大业务吞吐量，可以用“万笔/小时“或”笔/秒“等来度量。对于性能较好的系统，最大并发数是指响应时间和业务吞吐量等性能指标满足用户性能需求的前提下，系统所能支持的最大并发数。可以看出在满足用户性能需求这一原则下，系统在加压过程中可能先达到最大处理能力，如果这是各项指标仍然满足性能需求，则需要继续加压才能测出最大并发数，即将后继加压过程中业务处理能力或交易响应时间等性能指标开始不满足用户需求前的并发数作为最大并发数。可以看出，对于性能较好的系统最大处理能力和最大并发数往往不存在对应关系。对于性能较差的系统，最大并发数习惯上定义为业务吞吐量达到最大值时所对应的并发数，还有一种做法是将加压过程中响应时间开始不满足用户需求前的并发数定义为最大并发数，实际上，对于响应时间与业务吞吐量等关键性能指标不能满足用户性能需求的系统，讨论最大处理能力和最大并发数没有实际意义，这一点与性能较差时不讨论最佳处理能力与最佳并发数这两个性能指标的出发点是一致的。对于此类性能较差的系统，首先需要做的是对系统进行全面优化，然后再进一步测试系统的最大处理能力/最大并发数。
3. 系统容量上限：系统交易成功率<99%时的并发数
4. 系统上限：系统变得极其缓慢、交易大量失败甚至停止服务时的并发数。对于各个渠道，定义为交易成功率<90%时的并发数。

Table 容量测试场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 50 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/10秒 |
| 100 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/10秒 |
| 150 | <1秒 | >100笔/秒 | 20分钟 | 1个用户/10秒 |
| 200+ | <1秒 | >100笔/秒 | 每增加100个用户后续持续加压10到20分钟 | 1个用户/5秒 |
| 300+ | <1秒 | >100笔/秒 | 1个用户/5秒 |
| 500+ | <3秒 | >100笔/秒 | 1个用户/2秒 |
| 1000+ | <5秒 | >100笔/秒 | 1个用户/1秒 |
| 2000+ | <10秒 | >100笔/秒 | 1个用户/1秒 |

注：以上“响应时间要求”各个渠道根据实际情况进行要求

### 疲劳场景

联机与批处理每天先后轮流执行，连续测试一周。9:00-21:00执行联机交易，21:00-次日9:00执行EOD。测试过程可以根据实际情况进行调换。

联机交易场景采用正常压力持续加压，加压分为3个时段进行。这个过程的目的是为模拟全行一天24小时的业务。

Table 疲劳测试场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |
| 生产压力 | <1秒 | >100笔/秒 | 9小时 | 1个用户/10秒 |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |

### 批处理场景

采用手工/自动调用的方式对后台作业进行测试

Table 批处理测试场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |
| 生产压力 | <1秒 | >100笔/秒 | 9小时 | 1个用户/10秒 |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |

### 极限测试

以超过系统处理能力的压力对系统进行不断加压的极限状态测试，测试系统在极限压力下处理能力是否稳定以及是否出现宕机等严重性能问题

Table 10 极限测试场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |
| 生产压力 | <1秒 | >100笔/秒 | 9小时 | 1个用户/10秒 |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |

### 异常场景

测试在正常生产压力下的各种异常场景，例如功能上有互斥关系或者有锁机制的场景、网络闪断、数据库主机切到备机、数据库执行备份操作等场景，重点检查异常场景发生时是否会导致系统出现响应异常等各类性能问题

Table 11 异常测试场景设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **并发数** | **响应时间要求** | **TPS要求** | **加压时间** | **加压方式** |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |
| 生产压力 | <1秒 | >100笔/秒 | 9小时 | 1个用户/10秒 |
| 生产压力\*70% | <1秒 | >100笔/秒 | 1小时 | 1个用户/10秒 |

# 测试实施安排

## 测试进度

Table 12 测试进度表

|  |  |  |
| --- | --- | --- |
| **名称** | **负责人** | **完成时间** |
| 性能测试计划文档编写 | 李可欣 | 2018/3/15 |
| 测试环境准备 | 李可欣 | 2018/3/15-2018/3/25 |
| 测试数据准备 | 李可欣 | 2018/3/25-2018/3/30 |
| 测试用例编写 | 李可欣 | 2018/3/30-2018/4/5 |
| 性能测试执行 | 李可欣 | 2018/4/5-2018/4/10 |
| 执行报告编写 | 李可欣 | 2018/4/10-2018/4/15 |

## 测试流程



## 测试报告需求

Table 13 测试报告需求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **报告类型** | **响应时间要求** | **报告者** | **接受者** | **报告内容** |
| 邮件 | 每日 | 各渠道测试组负责人 | 测试中心，项目经理 | 每日结果总结报告 |
| 邮件 | 每日 | 各渠道测试组负责人 | 测试中心，项目经理 | 每日测试结果记录 |
| 邮件 | 每日 | 性能测试负责人 | 测试中心，项目经理 | 周报 |
| 文档 | 各测试阶段结束 | 各渠道测试组负责人 | 测试中心，项目经理 | 性能测试报告 |

## 性能缺陷管理

测试过程采用Quality Center进行缺陷管理

# 性能测试标准

## 启动标准

1. 测试环境满足计划需求
2. 基准参数配置完成校验
3. 关键交易通过冒烟测试

## 中止标准

1. 测试环境或关键系统不可用
2. 测试环境距生产标准差距太大
3. 缺陷周转周期不符合规定的时间
4. 出现宕机、不响应等严重的性能问题
5. 系统的交易成功率低于95%

## 通过标准

系统上线至少满足下面标准：

1. 系统无宕机、不响应类的严重性能问题
2. 系统响应时间80%达到系统的期望值
3. 系统的业务吞吐量达到预期目标，即当前生产需求的3倍以上
4. 柜台类系统要求通过7\*12小时以上的疲劳强度测试
5. 电子渠道类系统要求通过7\*24小时以上的疲劳强度测试

# 测试环境规划

## 部署环境

## 执行环境

# 测试风险分析

1. 需求风险

产品需求的不明确，对产品需求理解不准确，导致测试范围存在误差，遗漏部分需求或者执行了错误的测试方式；另外需求变更导致测试用例变更，测试用例维护成本增加，实时更新时存在误差。

1. 测试用例风险

测试用例设计不完整，忽视了边界条件、异常输入等情况，用例覆盖率没有做到足够覆盖，测试用例没有得到全部执行，有些用例被有意或者无意的漏测，需求变更导致的测试时间被压缩等情况。

1. 缺陷风险

某些缺陷偶发，难以重现，容易被遗漏；缺陷跟踪不够积极主动，没做好缺陷记录和及时更新，同样的缺陷，导致的原因可能不同，对这点没意识到导致的线上生产问题等。

1. 代码质量风险

代码质量差，可读性差，重构性差，没做好注释等原因导致缺陷较多，修改难度增大；还有系统架构设计的不足，导致的扩展性不足，性能兼容差等问题。

1. 测试环境风险

测试环境和生产环境配置不同，测试环境交叉影响较大，测试环境数据量不足导致的测试结果误差等问题。

1. 测试技术风险

某些项目存在技术难度，测试能力和经验所限，技术水平相对较差导致测试进展缓慢，测试结果准确性不够，项目发布日期延期等问题。

1. 回归测试风险

回归测试，一般时间相对来说较少，且大多只回归主要的功能点用例，可能造成漏测；另外还有回归验证缺陷时业务流走不通导致的打回修复再验证造成的时间延后问题。

1. 沟通协调风险

项目进行过程中需要多方沟通协调，不同部门，岗位之间的沟通、协作，难免存在误解、沟通不畅的情况，比如需求变更没有及时沟通，开发代码提交没有及时告知，测试结果的反馈不及时等问题。

1. 研发流程风险

其中包括从产品需求评审、研发设计、代码提交、测试发布等一些列流程，流程的不规范不协调很可能导致很多问题；比如开发在不告知其他成员的情况下提交代码，发布没有预生产环境，生产出现问题无法及时回滚等问题。

1. 其他不可预估风险

一些突发状况、不可抗力等也构成风险因素，且难以预估和避免。

# 角色与职责

性能测试工程师、IT运维人员、开发代表、需求、架构设计人员、DBA、项目经理和测试经理的支持。