**Bugfree性能测试方案**

目录

[1.BugFree系统性能测试概述 2](#_Toc8657177)

[1.1被测系统定义 2](#_Toc8657178)

[1.2性能测试指标 2](#_Toc8657179)

[1.3系统结构及流程 3](#_Toc8657180)

[1.4性能测试环境 4](#_Toc8657181)

[2性能测试 4](#_Toc8657182)

[2.1预期性能概述 4](#_Toc8657183)

[2.2用户并发测试 5](#_Toc8657184)

[2.3大数据量测试 5](#_Toc8657185)

[2.4负载能力测试 5](#_Toc8657186)

[2.5测试指标及期望 5](#_Toc8657187)

[2.6应用软件级别的测试指标 5](#_Toc8657188)

[2.7网络级别的测试指标： 6](#_Toc8657189)

[3测试过程及结果描述 6](#_Toc8657190)

[3.1测试描述 6](#_Toc8657191)

[3.2测试场景 6](#_Toc8657192)

[3..3测试结果标准 6](#_Toc8657193)

前言

BugFree是借鉴微软的研发流程和Bug管理理念，使用PHP+MySQL独立写出的一个Bug管理 系统。简单实用、免费并且开放源代码(遵循GNU GPL)。随着软件行业蓬勃发展，各类软件如雨后春笋，bug也随之而来，未来势必会出现业务系统中信息量大量增加的态势。随着业务系统在生产状态下日趋稳定、成熟，系统能稳定在什么样的西能水平，面临行业公司业务增加时，系统能否经受住“考验”，这些问题需要通过一个完整的性能测试来给出答案。

本《性能测试计划书》即是基于上述考虑，参考科学的性能测试方法撰写的，用以指导即将进行的性能测试。

## 1.BugFree系统性能测试概述

### 1.1被测系统定义

BugFree作为本次测试的系统，是一款bug管理系统，bugfree基于浏览器，一个Bug从创建到关闭这个“生命周期”的处理过程，BugFree 全面借鉴Raid的处理流程，处理方法甚至词汇都和Raid一样 (所以我用BugFree处理Bug的感觉和在微软时候基本一样)；

BugFree 还有一个独创的功能：当一个Bug被指派给你的时候，系统会自动给你发一封邮件，告诉你有个Bug需要你处理，这样结合 Email，BugFree被完美使用起来，成为我们[网站开发](https://baike.baidu.com/item/%E7%BD%91%E7%AB%99%E5%BC%80%E5%8F%91" \t "_blank)、运行、维护必备的工具。我们还增加了两个Bug统计功能：一是每天早上8点钟每个同事都会收到一封Email，告诉他/她头上还有多少 Bug等待处理；二是每周一中午给所有人发一封邮件，告知上周Bug的处理情况和到目前为止所有Bug的统计数据。

### 1.2性能测试指标

本次测试是针对bugfree系统进行全面的性能测试，主要需要获得如下测试指标。

1. 应用系统的负载能力：即系统所能容忍的最大用户数量，也就是在正常的响应时间中，系统能够支持的最多的客户端的数量。
2. 、应用系统的吞吐量：即在一次事务中网络内完成的数据量的总和，吞吐量指标

反映的是服务器承受的压力。事务是用户某一步或几步操作的集合。

1. 应用系统的吞吐率：即应用系统在单位时间内完成的数据量，也就是在单位时间内，应用系统针对不同的负载压力，所能完成的数据量。
2. TPS：每秒钟系统能够处理事务或交易的数量，它是衡量系统处理能力的重要指标。
3. 点击率：每秒钟用户向服务器提交的HTTP请求数。
4. 系统的响应能力：即在各种负载压力情况下，系统的响应时间，也就是从客户端请求发起，到服务器端应答返回所需要的时间，包括网络传输时间和服务器处理时间。
5. 应用系统的可靠性：即在连续工作时间状态下，系统能够正常运行的时间，即在连续工作时间段内没有出错信息。

### 1.3系统结构及流程

BugFree系统在实际应用中的体系结构跟本次性能测试所采用的体系结构是一样的，作为一个学测试的学生，可以应用bugfree来管理基本的bug。实际应用中可能bug会更加的丰富。

#### 1.3.1系统总体结构

描述本系统的总体结构，包括：硬件组织体系结构、网络组织体系结构、软件组织体系结构和功能模块的组织体系结构。

#### 1.3.2功能模块

1.本次性能测试中各类操作都是由若干功能模块组成的，每个功能都根据其执行特点分成了若干操作步骤，每个步骤都是一个功能点（即功能模块），本次测试主要涉及的模块及操作如下

|  |  |  |
| --- | --- | --- |
| 步骤 | 说明 | 备注：Action；平均响应时间 |
| 1 | 打开主界面 | Action：访问首页；5 |
| 2 | 输入用户名密码（参数化），登录系统，进入首页 | Action：登录；5 |
| 3 | 查询bug（模块，路径，创建者，等于否等，提交查询），进入要查询的bug页面 | Action：提交查询；5 |
| 4 | 点击bug进入bug页面 | Action：bug；5 |
| 5 | 点击case进入case页面 | Action：case；5 |
| 6 | 点击result进入result界面 | Action：result；5 |
| 7 | 新建bug（点击新建bug进入页面，填入必填项，保存） | Action：新建bug；5 |

#### 1.3.3关键点描述（KP）

本次性能测试的关键点，就是查看XXX系统在不同用户数量（并发）压力下的表现和大数据量操作时系统的性能状态，本次性能测试的关键点，就是查看XXX系统在不同用户数量（并发）压力下的表现和大数据量操作时系统的性能状态，

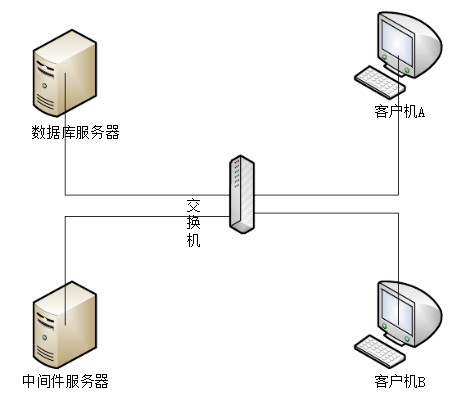
### 1.4性能测试环境

本次性能测试环境与真实运行环境硬件和网络环境有所不同，是真实环境的缩小，数据库是真实环境数据库的一个复制（或缩小），本系统采用标准的cs结构，客户端通过前台安装访问应用系统。

其中具体的硬件和网络环境如下：

* 操作系统：windows10/linux
* 网络环境：LAN（100M）
* 数据库：MySQL
* 客户端: PC

网络拓扑和结构图如下：



# 2性能测试

本次包括：预期性能测试，用户并发测试、大数据量测试、疲劳强度测试、负载能力测试等。

将使用测试工具LoadRunner11.0对被测试项目的各模块进行监控，并判断该系统各模块的性能表现。

### 2.1预期性能概述

通过模拟bug的不断建立的压力量和使用场景组合，测试系统的性能是否满足生产性能要求。.

特点：1.主要是验证系统是否有系统宣称具有的能力。

2.要事先了解被测系统经典场景，并具有确定的性能目标。

3.要在已确定的环境下运行。

### 2.2用户并发测试

并发测试方法通过模拟用户并发访问，测试多用户并发访问同一个应用、同一个模块或者数据记录时是否在死锁或者其他性能问题。

目的：1.发现系统中可能影藏的并发访问的问题。

2.关注系统可能存在的并发问题，例如系统中的内存泄露、线程锁和资源争用方面的问题。

### 2.3大数据量测试

Bugfree的主要功能就是管理大数据量的bug，所以这个是最为关键的，在处理大数据量是确定是否达到了将使软件发生故障的极限。

目的：1.确定软件发生故障的极限。

2.确定测试对象在给定时间内能够持续处理的最大负载和工作量。

3.可以在开发的各个阶段使用需要相关测试工具的配合和支持。

### 2.4负载能力测试

通过给被测系统不断的加压，直到性能指标达到极限，例如“响应时间”超过预定指标或某种资源以及达到饱和状态。

目的：1.找到系统处理能力的极限。

2.需要在给定的测试环境下进行，通常也需要考虑被测试系统的业务压力量，和典型场景，使得测试结果具有业务上的意义。

### 2.5测试指标及期望

在本次性能测试中，各类测试指标包括测试中应该达到的某些性能指标，这些性能指标均是来自于应用系统开发是遵循业务需求，当某个测试的某一类指标已经超出业务需求的范围，则测试已经达到目的，即可终止测试。

### 2.6应用软件级别的测试指标

* Cpu利用率小于40%
* 内存占用小于80%
* 吞吐量大于90%
* Response time小于1s
* 业务执行的平均响应时间（期望<15s）

### 2.7网络级别的测试指标：

* 吞吐量：单位时间内网络传输数据量
* 冲突率：在以太网上监测到的每秒冲突数

# 3测试过程及结果描述

### 3.1测试描述

在测试数据准备以后，测试将进行。记录每次测试的结果数据，分析测试结果对系统进行全面评估。

### 3.2测试场景

|  |  |  |
| --- | --- | --- |
| 步骤 | 说明 | 备注：Action；平均响应时间 |
| 1 | 打开主界面 | Action：访问首页；5 |
| 2 | 输入用户名密码（参数化），登录系统，进入首页 | Action：登录；5 |
| 3 | 查询bug（模块，路径，创建者，等于否等，提交查询），进入要查询的bug页面 | Action：提交查询；5 |
| 4 | 点击bug进入bug页面 | Action：bug；5 |
| 5 | 点击case进入case页面 | Action：case；5 |
| 6 | 点击result进入result界面 | Action：result；5 |
| 7 | 新建bug（点击新建bug进入页面，填入必填项，保存） | Action：新建bug；5 |

测试中，使用逐步加压的模式，测试运行场景安排如下：

1. 每隔一秒增加一条测试用例，最多增加到100条，查看并记录运行情况；
2. 每隔一秒增加两条测试用例，最多增加到200条，查看并记录运行情况；
3. 每隔一秒增加一条测试用例，最多增加到300条，查看并记录运行情况；
4. 每隔两秒增加一条测试用例，最多增加到400条，查看并记录运行情况；

每个场景都包括：用户登录-新建测试用例-输入必填项目-保存-退出

### 3..3测试结果标准

1.所有计划测试都已经完成；

2.所有计划收集的性能数据已经获得；

3.所有性能瓶颈得到改善并达到设计要求。

**执行每个场景时需要记录下相应的数据**

1. 服务器主机上的cpu利用率；
2. 在数据库服务器上主机上的cpu利用率
3. IO和CPU利用率对照：
4. 服务器监控的网络流量；
5. DB服务器上监控的网络流量；
6. 运行的并发用户数目
7. 测试中完成的各种操作的平均响应时间
8. 测试中每秒的点击率
9. 保存测试用例的吞吐量