

**<<Nginx>>**

**测试需求说明书**



北京航空航天大学

2016-05

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2016/5/6 | 余锋伟 | 全体 | 该文档的最初版本 |
| 2.0 | 2016/5/18 | 余锋伟 | 全体 | 根据评审意见作修改，细化了测试用例类别。 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 引言 4](#_Toc451390463)

[1.1 编写目的 4](#_Toc451390464)

[1.2 过程分析 4](#_Toc451390465)

[1.3 参考文献 5](#_Toc451390466)

[2. 测试计划 5](#_Toc451390467)

[2.1 目标 5](#_Toc451390468)

[2.2 测试工具及相关部件简介 5](#_Toc451390469)

[2.2.1Webbench 6](#_Toc451390470)

[2.2.2 PHP 6](#_Toc451390471)

[2.2.3 Python 6](#_Toc451390472)

[2.2.4 JSP 7](#_Toc451390473)

[2.3 测试数据 7](#_Toc451390474)

[2.4 测试策略 7](#_Toc451390475)

[2.5 测试通过准则 7](#_Toc451390476)

[3. 安装测试 8](#_Toc451390477)

[3.1 概述 8](#_Toc451390478)

[3.1.1 测试用例 8](#_Toc451390479)

[4. 功能测试 8](#_Toc451390480)

[4.1 概述 8](#_Toc451390481)

[4.2 静态文件测试 9](#_Toc451390482)

[4.2.1 测试用例 9](#_Toc451390483)

[4.3 动态网页测试 9](#_Toc451390484)

[4.3.1 测试用例 9](#_Toc451390485)

[4.4 配置测试 9](#_Toc451390486)

[4.4.1 测试用例 9](#_Toc451390487)

[4.5 可视化模块功能测试 10](#_Toc451390488)

[4.5.1 测试用例 10](#_Toc451390489)

[5. 非功能测试 10](#_Toc451390490)

[5.1 可靠性测试 10](#_Toc451390491)

[5.1.1 可靠性测试方案 11](#_Toc451390492)

[5.1.2 测试用例 11](#_Toc451390493)

[5.2 性能测试 12](#_Toc451390494)

[5.2.1 测试方案 13](#_Toc451390495)

[5.2.2 测试用例 13](#_Toc451390496)

[6. 测试方案的自我评价 16](#_Toc451390497)

# 引言

## 编写目的

本文档是软工实验课程中对所选软件和扩展功能的测试需求分析的过程及概述。其主要描述了测试类型的选择、测试方案、测试用例产生原因和项目中的测试用例设计说明等。测试人员可通过文档的测试方案结合测试用例，对系统进行测试。

## 过程分析

小组成员在讨论后，决定主要通过对Nginx功能需求和非功能需求两方面的主要特征进行测试，测试计划的指定依据为《需求规格说明书》。Nginx是web服务器，Web服务器的特征点较多，我们主要选取了最有代表性的特征进行测试。

为了使得测试用例更加规范，我们小组约定了几条在编写测试用例时需要遵守的几条规则：首先，测试用例的前置条件应该与对应需求用例中的条件相对应；其次，测试用例的主流程应该与对应需求用例的主流程相对应，至少包括用例中参与者与系统的交互过程；最后测试用例中应该给出明确的判断测试用例是否通过的标准。

## 参考文献

《深入剖析Nginx》

《Nginx权威指南》

《Nginx中文文档》

《需求规格说明书v2》

# 测试计划

## 目标

本次软件工程实验中，涉及到的“系统”主要有两个：一个是所选择的软件A，这部分软件代码是开源的；另一个是根据实际需求对软件A的扩展，这部分的代码全部由组员来完成。对于这两个“系统”所做的测试，其目的应该是不一样的。

具体在本项目中，对于所选的软件Nginx，是一款web服务器。对于这个开源软件而言，测试的目的有两类：

1. 测试其功能是否完整，包括web服务器所应具有的基本功能，和高级web功能。
2. 测试其性能是否优于其他web服务器。

在本课程中，实现了Nginx配置和监控的可视化。对于扩展部分的测试，目的是验证其是否满足《需求规格说明书》中提到的基本功能，包括配置文件可视化、监控等功能实现的是否完整、稳定、可用。其中测试的结果可以反馈给开发人员作进一步的改进。

## 测试工具及相关部件简介

本小节主要介绍在以后的实际的测试过程中有可能用到的工具，包括各类web服务涉及到的动态语言。

## Webbench

Webbench是有名的网站压力测试工具，它是由Lionbridge公司开发(http://www.lionbridge.com)。 Webbench能测试处在相同硬件上，不同服务的性能以及不同硬件上同一个服务的运行状况。Webbench的标准测试可以向我们展示服务器的两项内容：每秒钟相应请求数和每秒钟传输数据量。Webbench不但能具有便准静态页面的测试能力，还能对动态页面（ASP，PHP，JAVA，CGI）进行测试的能力。还有就是他支持对含有SSL的安全网站，例如电子商务网站进行静态或动态的性能测试。

## PHP

PHP（外文名:PHP: Hypertext Preprocessor，中文名：“超文本预处理器”）是一种通用开源脚本语言。语法吸收了C语言、Java和Perl的特点，利于学习，使用广泛，主要适用于Web开发领域。PHP 独特的语法混合了C、Java、Perl以及PHP自创的语法。它可以比CGI或者Perl更快速地执行动态网页。用PHP做出的动态页面与其他的编程语言相比，PHP是将程序嵌入到HTML（标准通用标记语言下的一个应用）文档中去执行，执行效率比完全生成HTML标记的CGI要高许多；PHP还可以执行编译后代码，编译可以达到加密和优化代码运行，使代码运行更快。

## Python

Python是一种面向对象、解释型计算机程序设计语言，由Guido van Rossum于1989年底发明，第一个公开发行版发行于1991年，Python 源代码同样遵循 GPL(GNU General Public License)协议[1] 。Python语法简洁而清晰，具有丰富和强大的类库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。常见的一种应用情形是，使用Python快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如3D游戏中的图形渲染模块，性能要求特别高，就可以用C/C++重写，而后封装为Python可以调用的扩展类库。需要注意的是在您使用扩展类库时可能需要考虑平台问题，某些可能不提供跨平台的实现。

## JSP

JSP全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的Servlet设计，它[1] 是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP技术有点类似ASP技术，它是在传统的网页HTML（标准通用标记语言的子集）文件(\*.htm,\*.html)中插入Java程序段(Scriptlet)和JSP标记(tag)，从而形成JSP文件，后缀名为(\*.jsp)。 用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。

## 测试数据

本测试实践的测试数据来源，主要有我们自己编写的测试脚本（webbench命令）以及自己定义的配置文件（Nginx.conf文件）。

## 测试策略

我们准备进行测试的过程为Nginx系统测试，系统测试的测试策略如下：

系统测试的主要目的是在于验证软件的功能和性能及其他特性是否与用户的要求一致，主要是下列类型的测试:

1. 安装测试：确定Nginx在各种操作系统（linux，windows，MacOS）中都能够被正常编译，安装。
2. 功能测试：确定Nginx是否完全的实现了《需求说明书》中的功能点。主要内容详见第三节。确定Nginx图形化配置监控模块是否满足所需的基本功能。
3. 非功能测试：确定Nginx在一定条件下，性能的曲线图，包括强度测试，容量测试，安全性测试等。主要内容详见第四节。

## 测试通过准则

Nginx系统通过测试的准则，即当依据测试用例执行者测试结果与预期结果相符，或测试结果与预期结果虽有不符但不可归咎于Nginx本身时为测试通过，反之测试失败。

# 安装测试

## 概述

确定Nginx在各种操作系统（linux，windows，MacOS）中都能够被正常编译，安装。

## 测试用例

【】

# 功能测试

## 概述

功能测试就是对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能

Functional testing（功能测试），也称为behavioral testing（行为测试），根据产品特性、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求。本地化软件的功能测试，用于验证应用程序或网站对目标用户能正确工作。使用适当的平台、浏览器和测试脚本，以保证目标用户的体验将足够好，就像应用程序是专门为该市场开发的一样。功能测试是为了确保程序以期望的方式运行而按功能要求对软件进行的测试，通过对一个系统的所有的特性和功能都进行测试确保符合需求和规范。

功能测试也叫黑盒测试或数据驱动测试，只需考虑需要测试的各个功能，不需要考虑整个软件的内部结构及代码.一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出更加使产品达到用户使用的要求。

## 静态文件测试

该方面测试主要是针对Nginx对静态文件访问的功能测试，测试其基本的功能和对格式的支持。

## 测试用例

【补充静态文件测试用例】

## 动态网页测试

因为WEB的发展极为迅速，在当今的所有应用中，系统软件和WEB应用是两个重要的方面，我们认为该功能测试点能够完全的测试出Nginx在该方面所具有的所有功能，所以我们选择了该功能测试点。

搭建动态网页测试主要是测试在WEB页面中使用cgi接口返回WEB页面。WEB页面包括HTML代码、CSS代码、JS代码。

## 测试用例

【补充静态文件测试用例】

## 配置测试

Nginx采用nginx.conf文件对其进行大部分配置管理，配置文件使用的是自有的文件格式。对于不同的配置有不同的格式。我们需要对配置文件的选项、格式、可行性进行测试，包括其对配置文件错误的提示和鲁棒性。

## 测试用例

【补充配置文件测试】

## 可视化模块功能测试

在本项目中，主要完成一个基于python的nginx.conf可视化工具的开发。根据需求文档中的要求，完成指定配置项的可视化配置和配置文件自动生成。并且具有一定的监控系统功能。

## 测试用例

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 用户组配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx用户组 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改用户组 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 进程数配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx进程数 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改进程数 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | Pid存放路径配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx的pid存放路径 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改pid存放路径 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 日志存放路径配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx的日志存放路径 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改日志存放路径 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 日志级别配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx的日志级别 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改日志级别 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 事件类型配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx的事件类型 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改事件类型 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 最大连接数配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx的最大连接数 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改最大连接数 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 显示系统状态 | |
| Brief Description | 测试图形化配置工具是否能够正确显示系统状态 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 关闭nginx |
| 2 | 打开图形工具 |
| 3 | 查看是否能够正确检测到nginx处于关闭状态 |
| Postcondition  (Test) | 测试成功 |
| Specific Alternative Flows  (Test) | RFS 3 | |
| 1 | 出现异常 |
| 2 | ABORT |
| Postcondition  (Test) | 测试失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 启动nginx服务器 | |
| Brief Description | 测试图形化配置工具是否能够正确启动nginx服务器 | |
| Precondition | Nginx正确安装，并且没有启动 | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形工具 |
| 2 | 点击启动按钮 |
| 3 | 通过浏览器查看nginx是否启动 |
| Postcondition  (Test) | 测试成功 |
| Specific Alternative Flows  (Test) | RFS 3 | |
| 1 | Nginx没有启动 |
| 2 | ABORT |
| Postcondition  (Test) | 测试失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 关闭nginx服务器 | |
| Brief Description | 测试图形化配置工具是否能够正确关闭nginx服务器 | |
| Precondition | Nginx正确安装，并且已经启动 | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形工具 |
| 2 | 点击关闭按钮 |
| 3 | 通过浏览器查看nginx是否关闭 |
| Postcondition  (Test) | 测试成功 |
| Specific Alternative Flows  (Test) | RFS 3 | |
| 1 | Nginx没有关闭 |
| 2 | ABORT |
| Postcondition  (Test) | 测试失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 重启nginx服务器 | |
| Brief Description | 测试图形化配置工具是否能够正确重启nginx服务器 | |
| Precondition | Nginx正确安装，并且已经启动 | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形工具 |
| 2 | 点击重启按钮 |
| 3 | 通过浏览器查看nginx是否先停止服务再恢复服务 |
| Postcondition  (Test) | 测试成功 |
| Specific Alternative Flows  (Test) | RFS 3 | |
| 1 | Nginx没有重新启动 |
| 2 | ABORT |
| Postcondition  (Test) | 测试失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 监听端口配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx监听端口 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改监听端口 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 主页配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx主页 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改主页 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 根目录配置 | |
| Brief Description | 测试图形化配置工具是否能够正确配置nginx根目录 | |
| Precondition | None | |
| Primary Actor | 测试员 | |
| Dependency | None | |
| Basic Flow  (Test) | Steps | |
| 1 | 打开图形化配置工具 |
| 2 | 打开现有配置文件 |
| 3 | 更改根目录 |
| 4 | 保存配置文件 |
| 5 | 查看配置文件是否被正确修改 |
| Postcondition  (Test) | 配置成功 |
| Specific Alternative Flows  (Test) | RFS 5 | |
| 1 | 配置文件没有正确修改 |
| 2 | ABORT |
| Postcondition  (Test) | 配置失败 |

# 非功能测试

非功能主要包括可靠性测试和性能测试。

对于软件的可靠性来说，一般分为两类，一类是性能相关的可靠性，比如软件处理大文件，高并发时，是否会出现运行异常等情况。另一类是功能相关的稳定性，比如系统的BUG数，对于非标准配置或者一些异常操作呈现的的鲁棒性，以及部署在生产环境中能否保证长时间运行程序不出错。

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。[负载测试](http://baike.baidu.com/view/651437.htm)和[压力测试](http://baike.baidu.com/view/565536.htm)都属于性能测试，两者可以结合进行。通过[负载测试](http://baike.baidu.com/view/651437.htm)，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。[压力测试](http://baike.baidu.com/view/565536.htm)是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。

从广泛意义上讲性能测试包括：压力测试、稳定性测试、负载测试和可扩展测试等。在不同应用系统的性能测试中，需要根据应用系统的特点和测试目的的不同来选择具体的测试方案。

## 可靠性测试

对于以上提到的两种可靠性，不同的应用场景有不同的可靠性需求。我们提出简单的量化评估可靠性方法。如下表：

|  |  |
| --- | --- |
| 可靠性评估指标 | 描述 |
| 7\*24小时标准 | 部署在生产环境中，必须满足至少7\*24小时稳定运行。 |
| 并发数标准 | 服务器同时能够保持的连接数 |
| 鲁棒性标准 | 对于错误、不完全的http/tcp请求，保持系统的稳定性，并返回规范的信息。 |
| 系统响应标准 | 发起一次http请求得到响应的时间必须保证在一个阈值之下 |

## 可靠性测试方案

针对以上提到的可靠性测试指标，将会配置一台标准web服务器对外产生服务，对其进行7\*24小时测试。

针对鲁棒性测试，使用http请求工具构造不同的请求，并分析响应的正确性。

并发性标准和系统响应标准，放在性能测试时讨论。

## 测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 7\*24小时测试 | 测试用例标识 | TC001 | 对应测试需求文档中的需求 | 4.2节中的7\*24小时标准 | |
| 简要描述 | 本测试主要测试在生产环境中，Nginx能否提供7\*24小时的稳定性。 | | | | | |
| 前提和约束 | 生产环境中网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 鲁棒性测试 | 测试用例标识 | TC001 | 对应测试需求文档中的需求 | 4.2节中的鲁棒性标准 | |
| 简要描述 | 本测试主要测试在使用http请求工具构造不同的请求时，系统响应的正确性。 | | | | | |
| 前提和约束 | 网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

## 性能测试

Nginx 处理请求是异步非阻塞的，而Apache 则是阻塞型的，在高并发下Nginx 能保持低资源低消耗高性能。相比 Apache，Nginx 使用更少的资源，支持高达 50,000 个并发连接，体现更高的效率。Apache是同步多进程模型，一个连接对应一个进程；Nginx是异步的，多个连接（万级别）可以对应一个进程。

在性能需求中，和可靠性一样，不同的产品对web服务器的性能需求各不相同。针对不同的性能需求，我们提出如下几种性能需求标准：

|  |  |
| --- | --- |
| 性能评估指标 | 描述 |
| Web存储性能标准 | 通过http对外提供文件存储服务，客户端下载大文件的处理效率（带宽） |
| Web响应标准 | 在普通的网页访问中，响应一次http请求所需的时间 |
| 并发标准 | 在同一个Web服务器中，最高可支持多少并发链接数 |
| 单个链接占用服务器资源标准 | 对于维持一个http链接，服务器需要花费的cpu资源和内存资源 |

## 测试方案

使用上述提到的开源的http压力测试工具，针对存储性能、响应标准，并发标准，资源占用标准。设置响应的测试脚本。在搭建好的web服务器上，对Nginx的表现进行分析。

## 测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 大文件存储性能测试 | 测试用例标识 | TC001 | 对应测试需求文档中的需求 | 4.4节中的Web存储性能标准 | |
| 简要描述 | 本测试主要测试在同样的配置下，apache和nginx对于同样的大文件传输性能差异。 | | | | | |
| 前提和约束 | 测试网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 基本请求的响应速度测试 | 测试用例标识 | TC002 | 对应测试需求文档中的需求 | 4.4节中的Web响应标准 | |
| 简要描述 | 本测试主要测试在同样的配置下，apache和nginx处理一次http响应的时间差异。 | | | | | |
| 前提和约束 | 测试网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 并发支持测试 | 测试用例标识 | TC001 | 对应测试需求文档中的需求 | 4.4节中的并发标准 | |
| 简要描述 | 本测试主要测试在同样的配置下，apache和nginx处理同一时刻并发请求下，所需的响应时间性能上的差异。 | | | | | |
| 前提和约束 | 测试网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例名称 | 单链接占用资源测试 | 测试用例标识 | TC001 | 对应测试需求文档中的需求 | 4.4节中的单个链接占用服务器资源标准 | |
| 简要描述 | 本测试主要测试在同样的配置下，apache和nginx处理同一时刻并发请求下，所需的CPU和内存资源。 | | | | | |
| 前提和约束 | 测试网页搭建完成，测试网络稳定 | | | | | |
| 测试方法 | 黑盒测试 | | | | | |
| 测试过程描述 | | | | | | |
| 序号 | 测试步骤 | | 预期结果 | | 评价准则 | 测试结论 |
| 1 |  | |  | |  | 无 |
| 2 |  | |  | |  | 无 |
| 工作量估计（h） |  | | | | | |
| 测试人员： | | 监测人员： | | 测试时间： | | |

# 测试方案的自我评价

本次测试方案基本覆盖了需求文章中提到功能性、非功能性需求。对于功能性需求以功能测试为主，设计了覆盖Nginx基本功能的测试用例，对于扩展图形化配置模块，针对其读和写也都设计了测试用例。

对于非功能需求，分为可靠性和性能两个方面，并分别进行了测试。

在每个测试用例中，包含了其可追述的需求文档中位置，以及工作量估计。并且给出了实现每个测试的基本步骤。

综上所述，本测试方案较为全面地覆盖了需求规格说明书中对应需求。