# 性能测试

## 一、相关概念

性能测试方法是通过模拟生产运行的业务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。通俗地说，这种方法就是要在特定的运行条件下验证系统的处理能力。

### 基准测试

准备性测试，帮助验证性能测试脚本能够在性能环境中正确执行。通常使用一个虚拟用户执行单个用例，获得最佳的用例性能效果，然后将其和后续的测试数据做对比。

### 负载测试

这种方法是对一个系统持续不段的加压，看你在什么时候已经超出“预期要求”或系统崩溃。

最典型的性能测试类型。施加足够的负载来达到预期的并发压力，达到目标后不再增加。

目的：验证可用性，并发数，吞吐率和响应时间的性能指标。

### 压力测试

压力测试方法测试系统在一定饱和状态下，例如CPU、内存在饱和使用情况下，系统能够处理的会话能力，以及系统是否稳定会出现错误。尝试探测应用或者应用的支撑基础设施某些部分的极限能力。

目的：找到被测系统的容量天花板。

会一直增加负载，直到应用的部分功能不能正常工作。

### 并发测试

并发测试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存在死锁或其者他性能问题。

### 可靠性测试

目的：为了发现那些只有长时间运行才能发现的问题。典型的例子包括内存泄漏或者某些用例没有预见地执行次数的上限。

必须有合适的监控系统。

发现问题常常表现为响应时间缓慢增加，或某时刻系统突然变得不可用。

需要将问题发生时刻的负载数据和监控数据关联后进行分析。

### 性能冒烟测试

性能冒烟测试：只测试那些被代码变更影响到的性能测试用例。

### 隔离测试

目的：定位和排查问题。

通常会对已经定位产生性能问题的用例多次重复执行。

## 二、性能指标

可用性。（可用时长）

响应时间。（用户发起请求到应用响应完全到达用户客户端所消耗的时间）

吞吐率。（某些面向应用的事件的发生速率）

资源利用率。（对某种资源理论容量的使用百分比）

### 思考时间

思考时间指的是用户在与应用进行交互时必然存在的延迟和暂停时间。

### 响应时间

从用户发起请求到应用响应完全到达用户客户端所消耗的时间。整个过程分三个部分：**呈现时间，数据传输时间和系统处理时间。**

一般测试工具都屏蔽响应的呈现过程，只是模拟多用户并发请求，计算用户得到响应的时间，不会将服务器的每个响应做客户端渲染呈现。

对于数据传输的问题，这也是我要强调的性能测试要在局域网中进行，在局域网中一般不会受到数据带宽的限制。所以，可以对数据的传输时间忽略不计。

### 吞吐率

单位时间内网络上传输的数据量，也可以指单位时间内处理客户请求数量。它是衡量网络性能的重要指标，通常情况下，吞吐率用“字节数/秒”来衡量，当然，你可以用“请求数/秒”和“页面数/秒”来衡量。

### TPS (Transaction Per second)

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。

**点击率（Hit Per Second）**

点击率可以看做是TPS的一种特定情况。点击率更能体现用户端对服务器的压力。TPS更能体现服务器对客户请求的处理能力

需要注意的是，这里的点击不能简单的看作鼠标的一次“单击”操作，也许一次“单击”操作中，客户端可能向服务器发现多个HTTP请求。

### 并发用户数

虚拟并发用户数：从性能测试工具角度来看的活跃虚拟用户数。

应用并发用户数：活跃的虚拟用户，活跃指的是已经登录或者正在访问被测系统的虚拟用户。

虚拟并发用户数=应用并发用户数+退出用户数

### 服务器资源利用率

CPU利用率

内存使用率

I/O（磁盘和网络）

### RBI（rapid bottleneck identify）

是Empirix公司提出的快速识别系统性能瓶颈的方法。该方法基于以下事实。

    1. 发现的80%系统的性能瓶颈都由吞吐量制约；

    2. 并发用户数和吞吐量瓶颈之间存在一定的关联；

    3. 采用吞吐量测试可以更快速定位问题。

通过不断增加并发用户数和吞吐量观察系统的性能瓶颈。然后，从网络、数据库、应用服务器和代码本身4个环节确定系统的的性能瓶颈。

## 三、难点

性能测试的难点？

* 需求分析
* 场景设计
* 性能诊断调优
* 环境搭建和模拟

### 如何衡量可用性

可用性通常采用应用对于终端用户的可用时长来衡量。

## 四、性能测试流程

1. 获取非功能需求，确定性能目标
2. 构建性能测试环境
3. 编写性能测试脚本
4. 构建性能测试场景
5. 执行性能测试和分析
6. 测试结果分析和报告

### 问清性能需求

发布时间2017年8月11日[**虫师**](http://www.cnblogs.com/fnng/)

假如你一名新进公司的测试小菜鸟，在一次的早会中，领导突然分配给你了一个性能测试任务。我想此时的你是一脸的大写？？，怎么办？怎么办？ 这里会提供一个通用的测试思路，照着这个思路做，你就会顺利的完成性能测试。

**问清性能测试需求**

首先，要问本次性能测试的需求是什么，或者性能测试的目的是什么？ 我把性能测试按目的分以下几种。

**1）新系统能力验证**

比如，你们刚好开发了一个新系统，在上线前需要验证系统性能。这种情况比较简单；你可以有更多的自由选择测试环境、压力点和测试工具；测试策略上也比较灵活。并且如果性能测试结果没有明显的短板，也不需要进行调优。

**2）客户有明确要求**

这是一个好的结果，这说明客户对性能测试有一定的了解，知道他们需要的系统要达到一个什么样的标准。如：系统要求同时满足100用户登陆，平均每个用户登陆时间不能超过5秒。这个需求很明确，当然也不排除一些不懂装懂的用户，提一些不现实的要求。

不管怎么说，用户提要求了，这个比较容易，你可以对现系统做一次性能测试，至于，是通过优化系统还是增加硬件设备才能达到要求。就不是测试考虑的问题了。

**3）找出系统性能瓶颈**

这个需求的目的就很明确了，目的就是找出系统的性能瓶颈，进行调优或硬件扩容，所以性能测试的重点在系统的架构分析和业务分析上面。

**4）稳定性验证（强度测试）**

稳定性是系统的一个重要指标，因为系统一旦上线，就有可能会长期处在用户的访问状态，可能以前没发现的一些问题就会暴漏出来。比较典型的就是内存溢出，这种需求在测试策略上就要考虑性能测试的运行时长。

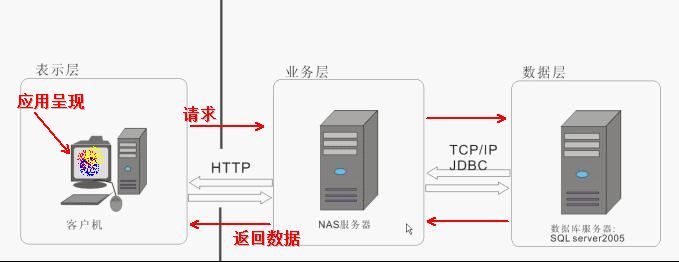
**注意：** 当拿到需求时一定要问测试的目的，一方面会显得你很专业；另一方面，我们通过测试目的可以知道后续性能测试工作的重点在哪儿？最主要的是，还可以揣摩出领导对这次测试的重视程度。^\_^!

### 了解系统架构

**发布时间 2017年8月11日**[**虫师**](http://www.cnblogs.com/fnng/)

系统架构对于测试新手来是最难的；先来了解系统所使用的技术和框架，在环境搭建阶段，你需要了解项目的部署；在性能分析与调优阶段，更要深入这些技术的细节去分析。

下面看一个基本的系统架构：



**表示层**

表示层（浏览器）通过前端技术（HTML5/JavaScript/CSS3）将系统功能和数据展示给用户，并与用户实现交互。通过TCP/HTTP协议与业务层系统通信，向应用层系统发送请求报文，并接收应用层系统返回的响应报文。

**业务逻辑层**

业务逻辑层作为中间层实现核心业务逻辑服务。

应用服务器主要运行中间件系统，中间件系统系统作为一个容器来运行各种应用软件系统。前台发来的请求报文通过中间件传递给应用程序，应用程序在处理的过程中调用数据层的数据服务器，数据服务器将查询的数据返回给应用程序，应用软件处理完成后通过中间件系统返回给客户端。

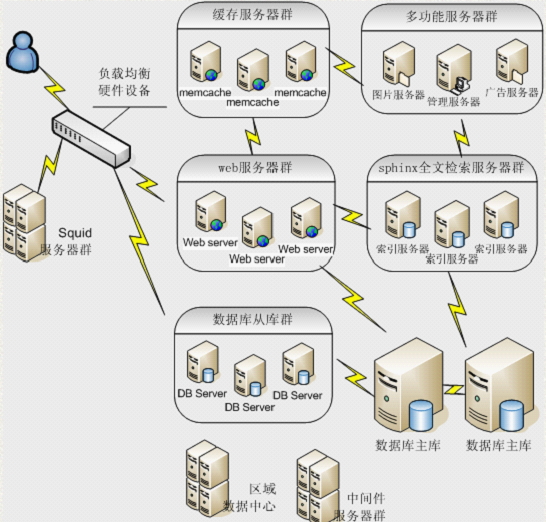
在大型的系统中，可以对应用系统进行拆分，比如拆分成交易服务，查询服务；或者通过负载均衡技术，来分散客户端发来的请求，使其能承受更大的用户访问量。

**数据层**

数据层运行在数据库主机上，负责整个系统中数据信息的存储。运行数据库服务程序，查询通过JDBC与应用程序进行通信，主要用于存储数据与提供数据查询等服务。

数据库集群技术就是对大型系统应用非常广泛的一种解决方案。

前面介绍了一般系统的架构，那么一个大型的系统在设计中使用了哪些手段或技术来提高系统的性能呢？下面通过一张图来了解一下。



常用的系统架构:

· Linux + Apache + PHP + MySQL

· Linux + Nginx + Redis +  PHP + MySQL

· Linux + Apache + Tomcat + Java+ Oracle

· Windows Server 2003/2008 + IIS + C#/ASP.NET + 数据库

· Window Server 2003/2008 + tomcat + MySql/Oracle/ + Java

· Linux + Python + uwsgi + Nginx + MySQL

### 分析测试点

**发布时间 2017年8月11日**[**虫师**](http://www.cnblogs.com/fnng/)

当你了解你的系统架构之后，接下来可以好好的分析一下性能测试点。因为这关系到你测试工具的选型。

**性能测试点的选取**

\*　发生频率非常高的（例如：某邮箱核心业务系统中的登录、收发邮件等业务，它们在每天的业务总量中占到90%以上）

\*　关键程度非常高的（产品经理认为绝对不能出现问题的，如登录等）

\*　资源占用非常严重的（导致磁盘I/O非常大的，例如某个业务进行结果提交时需要向数十个表存取数据，或者一个查询提交请求时会检索出大量的数据记录）

**对性能需求点的描述**

**准确**

如\*\*系统必须在不超过 10 秒的响应时间内，处理 20 起登录任务。再如发邮件时间最大不超过5秒以及平均时间在2秒以内。

**一致**

用户和性能测试工程师对有关术语的理解要一致，如:并发用户数、在线用户数、注册用户数:

**特定**

性能测试的需求一定是有条件的。

检查系统后台关键业务数据10G、操作数据量为20K，1500 个用户、500 个并发用户运行的负载下，连续运行12小时过程中，业务操作是否满足性能需求。

**一般性能需求描述**

1、Web首页打开速度5s以下，Web登陆速度 15s以下。

2、邮件服务支持50万个在线用户

3、计费话单成功率达到99.999%以上。

4、在100个并发用户的高峰期，邮箱的基本功能，处理能力至少达到10QPS(TPS). QPS(TPS)–每秒钟请求/事物 数量

5、系统能在高于实际系统运行压力1倍的情况下，稳定的运行12小时。

6、这个系统能否支撑200万的VU（每天登录系统的人次）VU–Virtual user(虚拟用户)

### 测试工具选型

**发布时间 2017年8月11日**[**虫师**](http://www.cnblogs.com/fnng/)

根据对系统性能测试点的分析，我们选择合适的工作。不是所有的性能需求都要用LoadRunner或Jmeter来进行的。只要能满足需求就可以了，我个人倾向于轻量级的性能工具，或自己动手写代码。

**测试工具的比较：**

软件性能测试工具分为三类，商业收费工具、开源免费工具、自主开发工具。

表：

| **–** | **商业工具** | **开源工具** | **自研工具** |
| --- | --- | --- | --- |
| 优点 | 功能强大且简单易用；提供丰富的功能，支持更多协议和并发用户数，丰富的计数器和测试结果；自动生成脚本，学习成本较低；很强的稳定性和可靠性。 | 轻量灵活容易扩展；可免费下载使用，可查看工具源代码，根须需求容易扩展。 | 贴近需求，开发出更适合需求的工具，较开源工具学习成本低。可形成特有的工具体系。 |
| 缺点 | 依赖工具本身提供的功能，较难扩展。 工具体积大，购买成本高。 | 学习成本高，不能完全满足性能测试需求，需要功能扩展。功能相对比较简陋。 | 工具开发成本高，需要专业的工具开发人员或团队，工具成本也会存在不确定。 |

通过上面的比较分析，很难判别对某类工具的取舍，各自有其明显的优势，当然也有明显的缺点，如何更好选择工具，这里就要结合实际性能的需求了。

### 测试计划

**发布时间 2017年8月11日**[**虫师**](http://www.cnblogs.com/fnng/)

#### 一．简介

  简介部分就不用过多描述了，无非项目的背景，进行此次性能测试的原因，以及性能测试覆盖的范围等等，几乎所有项目文档都在开端对项目进行简单的阐述。

#### 二．性能测试需求

寻找的被测试对象和压力点。要测试的对象不是凭空想象出来，而是经过分析与系统数据收集得到。以下取几个典型的压力点。

**登录：**对于一般的系统来说，登录是用户操作系统的前提，如果用户根本就登录不了，那么其它功能将毫无用处。例如网游戏，开新服的时候，玩家挤破了脑袋只为登录。

**查询：**查询一般比较消耗系统和数据库资源。搜索引擎的查询功能就是典型，如果你在输入框内输入内容，很久就得不到结果。我想被称为“互联网入口”的搜索引擎就不会存在。

**交易：**对于一些电子商务系统来说，交易过程的性能要求是很高的，如果交易过程消耗用户很长时间的话，用户可能会转投其它平台。当然，除了交易速度外，对交易的成功率要求也是非常高的。不然，造成的损失也是不可估量的。

被测的系统应该是最重要的最基本的功能，也是用户使用最频繁的功能。

#### 三．测试环境

 这里的测试环境主要指的软件硬件环境和网络环境。

笔者认为性能测试最好在一个独立的环境内进行，这样不会受到外界的干扰，能够保证测试的数据是独立有效的。如果现你对某个已经上线的网站进行压力测试，那么你得到的数据不是独立的，因为你在做压力测试的时候，其它散户也在访问系统。

**软件环境：**

这里的软件环境主要指项目运行的环境，比如采用什么样的操作系统、中间件、和数据库。

**硬件环境：**

这里的硬件环境除了主要包括主机内部部件，cpu、内存、磁盘以及主板、网卡等，传输介质和路由器也应该考虑在内，

**网络环境：**

网络环境除了考虑测试机与被系统服务器在一个局域网中进行，还应该保证这个网络的独立性。如果在在性能测试的过程中，其它机子也在消耗着路由器资源。那么路由器也会影响到数据库的传输速度。

#### 四．数据准备

  在很多时候，我们是要准备测试数据的，例如系统不允许相同用户的重复登录，那么必须要生成合法的用户数据。有时要对系统进行查询测试，只有在系统有一定数据量进才能验证出系统的真实性能。一个数据库中有两条数据和有两千万条数据，同相一条查询操作，对系统造成的压力是完全不一样的。

系统所需数据的分析可以参考以下方式：

历史数据分析有助于数据量级的确定。从历史数据入手，找出高峰期数据量。

从其他相似或者相同系统入手，进行数据分析，找出高峰期数据量。

无历史或者相关系统可以参考的时候，就要对系统的性能数据进行估算，包含系统容量，并发数等数据，估算以后给相关人员进行评审或者修订以后，按照大家同意的性能指标进行测试。

…………

测试数据最好和真实数据相同，如果能够获得真实系统运行3个月的数据，我们就可以在此基础上进行性能测试。

关于数据的生成，我们可以通过工具完成，如数据库数据生成工具，大小文件生成工具等。

#### 五．测试工具

  前面已经介绍如何分析需求，需求确定下来之后，我们可以考虑引入什么样的工具适合性能需求。

当然，在引入工具的时候除了考虑可以是否满足需求，还应该考虑工具的成本，这不单指工具的购买成本，还有测试人员对工具的学习成本。

关于测试工具的选择，后面会单独有一章节介绍，这里就不细说了。

如果你选择的性能测试工具不是足够的强大的话，你可能还需要其它的辅助的工具。如果Jmeter利用BadBoy来录制脚本，更能提高脚本开发效率。在压力测试的过程中也可能需要性能计数器来记录软硬件的性能。如监控服务器CPU、内存的计数器，记录中间件日志的监控中工具，监控数据库性能的监控工具等。

#### 六．测试策略

  对于一个特定的业务系统，用户一般会分散在一天的各个时间段进行访问。在不同的时间段中，用户使用业务系统的频率不同，而系统的繁忙程度不同。在一些特定的条件下，可能出现短时间内用户集中访问某个业务系统的情况。例如对于公文处理子系统而言，可能就存在短时间内大量用户查看并办理某条公文的情况。 在进行性能测试时，应当使用“考虑最坏情况的原则”。也就是应当在用户使用业务系统最频繁、对系统造成最大压力的情况下对系统的功能进行测试，判断各功能和页面是否能够满足性能的要求，系统的响应时间是否过长。

另一方面，系统性能的验证必须做到“覆盖全面”。虽然系统中各个功能的使用频率并不相同，一些功能的使用频率相对于其他功能来说比较低，但是在进行性能测试和优化时，不能忽略这些功能，编制测试用例时也不能仅仅选择最常用功能。例如可能所有的用户都会访问我的通知列表，但是一般只有5%的用户会使用通过系统设置模块查找某个用户的信息；但是在测试时，我们并不能因为查看用户信息功能的使用频率相对较少，而忽略掉这项功能的测试。所以，这里进行系统性能测试时，对于不同业务，用户的访问比例应该做一个合理分配。

在测试策略上，我们还应该考虑，同一个系统在不同硬件环境下的性能表现。从而让系统满足需求的情况下，硬件配置也能达到一个最佳的状态。过份的增加硬件来满足需求也是一种浪费。再说增加硬件设备不是能解决所有性能问题的。

#### 七．人力与时间安排

  最后一条，就是要根据项目的进度要求以及规模，来进行人力与时间的安排。对于大型的性能测试，项目前期的需求调研，环境的部署，工具的选购或开发，人员对测试工具的学习与使用，性能测试的后进行，后期数据的分析与调优。都需要人员安排的。有可以需要专业的，系统工程师、数据库工程师、软件开发工程师、网络工程师以及性能测试工程师的共同参与配合完成。不是一个性能测试人员就可以全部搞定的。

### 测试环境搭建

**发布时间 2017年8月11日**[**虫师**](http://www.cnblogs.com/fnng/)

#### 性能测试环境与功能测试环境的区别

那么性能测试环境与功能测试环境有所不同，对于一些企业为了节约资源，进行功能测试的测试环境，一台服务器可以运行多个系统，通过技术手段可以使系统之间是不会相互影响的（以前公司就是一台服务器上跑多个tomcat）。 性能测试是要对整个系统运行的软件硬件环境进行测试的，如果某环境下运行多个系统，就很难判断其中的某个环境对资源的占用情况。

#### 保证测试环境与生产环境的一致性

**1、硬件环境，包括服务器环境、与网络环境**

如服务器的型号以及是否和其它应用程序共享此服务器，是否在集群环境下，是否通过BIGIP进行负载均衡，客户使用的硬件配置情况，使用的交换机型号，网络传输速率。

**2、软件环境**

\*版本一致性

包括包括操作系统、数据库、中间件的版本，被测系统的版本。

\*配置一致性

系统（操作系统/数据库/中间件/被测试系统）参数的配置一致，这些系统参数的配置有可能对系统造成巨大的影响。所以，除了保证测试环境与真实环境所使用的软件版本一致，也要关注其参数的配置是否一致。

**3、使用场景的一致性**

* 基础数据的一致性

包括预测的业务数据量，以及数据类型的分配。很简单的一个列子，一个系统的数据库只有10条数据和一条数据库里几千万条数据，我们在对其进行性能测试时，得到的性能指标可能会有非常大的差别。

为了保证每次测试环境的更加一致性，磁盘的使用情况以及磁盘的碎片情况也会或多或少的影响的性能。

* 使用模式的一致性

尽量模拟真实场景下用户的使用情况，其实，我们在做性能测试前期的需求分析，其主要目的也就是为了更真实的模拟用户的使用情况。

#### 实施策略

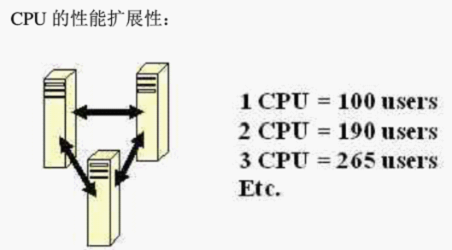
上面讲测试环境与生产环境保持一致所需要注意的内容。在实际的测试中，处于成本考虑，在很多情况下，我们很难申请到足够的且一致的资源，所以，很难搭建出与生产环境完全一致的一个测试环境。

我们一般通过两种策略来搭建性能测试环境（预估方式均有误差）

**1、通过建模的方式实现低端硬件对高端硬件的模拟**

通过配置测试来计算不同配置下的硬件性能和系统处理能力的关系，从而推导出满足系统性能的真实配置情况，这种模拟需要精确的建模，模型的采样点越多，那么得到的结果越精确，从而将在低端配置下的性能指标通过该模型转化为高端配置下的最终预计性能指标。

例如：搭建一个低端环境，首先需要对这个环境的CPU和内存进行单独的性能基准测试，同过在不同的配置的性能测试，得到一个基准信息列表，当然，在进行这个性能测试的过程中，我们要确定硬件是系统的瓶颈。如果只用一个CUP，在性能测试过程中，其使用率很低，但得到的性能数据都非常底，这起码说明CUP不是系统的瓶颈，这种情况下就无法得到想要的基准值。



如上图，在一颗CPU情况下，运行100个用户且CUP使用率接近饱和（100%）。在增加至两颗CUP的情况下，可以运行190个用户且UPU使用率接近饱和（100%），以此做记录，那么我们就可以推算出运行800个用户需要多少颗CUP。 如果你在实际应用中使用的CUP型号及其频率并非完全一样，这个时候可以使用EVEREST工具计算每种CUP的得分，对其性能进行评估。 内存也可以使用此方法进行测试推导，这里需要我们多进行试验，对硬件的性能以及对整个项目的结构都要做深入的了解，以便尽量减少误差。

**2、通过集群的方式计算**

对于较大的系统来说，单台服务器的处理能力是有限的，通常都会采用集群的方式来进行负载均衡，完成对海量请求的处理。虽然无法获得整体集群的测试环境，但是可以对集群上的一个节点进行性能测试，得出该节点的处理能力，再计算每增加一个节点的性能损失，同样也可以能过建模的方式得到大型负载均衡情况下的预计性能指标。

例如:首先在单台服务器上获得具体的性能指标，每台服务器能够承受500用户并发，平均TPS为60，响应时间为2秒，接着，添加负载均衡策略，再次测试负载策略下的数据损耗。得出数据后添加1台负载均衡服务器，测试在两台服务器下每台服务器的性能指标，以此类推，可以得到下表：

表：

| **负载服务器个数** | **并发用户数** | **平均TPS** | **响应时间** |
| --- | --- | --- | --- |
| 1 | 500 | 60 | 2 |
| 1 | 490 | 58 | 2 |
| 2 | 490 | 58 | 2 |
| 3 | 480 | 57 | 2 |

随着负载均衡服务器的添加，平均每台服务器的处理能力会逐渐稳定，从而了解在什么情况下需要多少台负载均衡服务器。