1

性能测试基础知识

在学习性能测试之前，有必要先了解一些性能测试的理论基础知识，为后期的性能测试做准备。需要了解什么是软件性能，性能测试需要关注的内容；了解在性能测试过程中常用的相关术语以及性能测试过程中需要关注的指标；了解性能测试的划分和应用领域，这样可以更好地确定需要进行哪些性能测试。

本章主要包括以下内容：

- 软件性能概述
- 性能测试相关术语
- 性能测试划分
- 性能测试应用领域

1.1 软件性能概述

什么是软件性能？大多数读者朋友会认为性能是一种指标，是反映软件系统或产品处理用户请求的响应时间，在软件质量模型中，性能被定义为软件的一种特性，软件质量模型如图 1-1 所示。

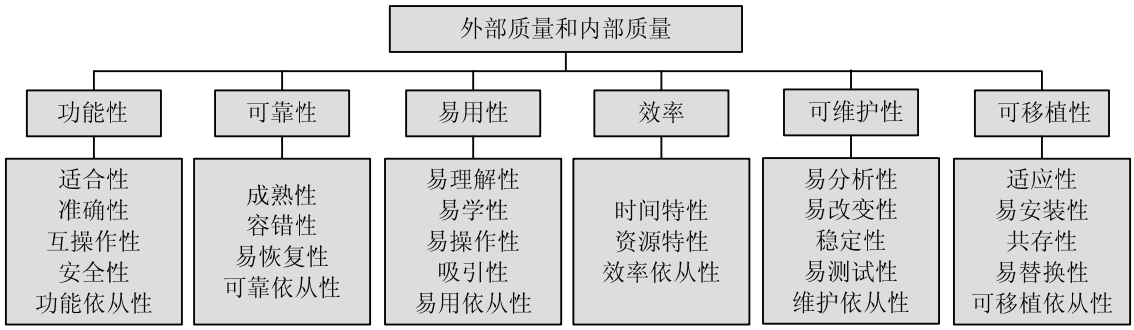

图 1-1　软件质量模型

在软件质量模型中效率特性即为软件的性能，其包含两个方面的特性：时间特性和资源特性。时间特性是指系统处理客户请求的响应时间；资源特性是指在进行性能测试过程中，系统资源消耗的情况，常见的系统资源主要包括处理器（CPU）、内存和磁盘的使用情况。所以通常说的软件性能不仅仅包括响应时间，还包括系统资源消耗。

虽然软件性能包含两个方面，但是不同的人所关注的性能层面有所不同（如用户只关注时间特性）。通常情况下，关注软件性能的人主要包括 3 类：用户、系统管理员与性能测试工程师、软件开发工程师。

1. 用户

从用户的角度来说，软件性能是软件系统对用户提交请求所响应的时间。通俗地讲，如果用户点击一个提交或输入一个 URL 地址，随后系统把结果呈现到用户眼前，这个过程所花费的时间即为用户对软件性能的直观印象，如图 1-2 所示。

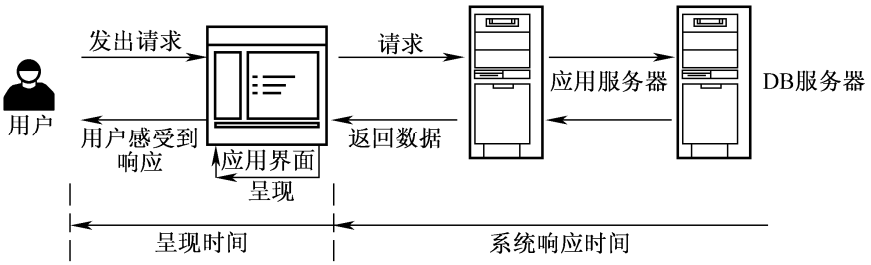

图 1-2 Web 系统响应

需要注意的是，用户体会到的“响应时间”，既有客观的成分，也有主观的成分。用户认为从提交业务到系统开始返回信息的时间为系统的响应时间。例如，用户执行某个操作，该操作会返回大量的数据，假如待返回的数据并不能同时返回到主界面，而是先将一部分数据呈现出来，再慢慢地将全部数据呈现出来，此时，用户体会的响应时间为从执行操作到已经有一部分数据呈现出来的时间，而真正的响应时间应该是系统将全部数据呈现出来的时间。

2. 系统管理员与性能测试工程师

从系统管理员和性能测试工程师的角度来说，在响应时间方面的理解与用户完全一致。但系统管理员和性能测试工程师是一群特殊的用户群体，其不仅仅关注系统的响应时间，还关注服务器系统资源的使用情况。系统管理员及性能测试工程师之所以关注资源的消耗情况，是因为系统响应时间达到要求并不代表系统就能正确地处理客户端提交的请求，例如银行系统，假设在处理存款业务时，每笔业务的响应时间为 100ms，但当前的 CPU 和内存的使用率都已达到 90%，超过正常使用的阈值，这样虽然响应时间达到要求，但是不能保证服务器不出问题，因为服务器已经处于一个临界状态，很可能出现存款不成功的情况，如果这样的话，即使响应时间更短也不能达到性能的要求。

另外，如果测试系统配置时，管理员和性能测试工程师还关注系统硬件资源的可扩展性即规划

性能部分。比如，系统现在支持 100 个用户并发没有问题，那么将来支持 200 个用户并发时是否会出现性能问题呢？

3. 软件开发工程师

从软件开发工程师的角度来说，他们关注用户和管理员关注的所有问题。另外，还关注内存泄漏、数据库是否出现死锁、中间件及应用服务器等问题。

1.2 性能测试相关术语

了解了什么叫系统性能之后，需要对性能测试过程中常用的术语有一个详细的了解，为后面的性能测试做准备。性能测试过程中的常用术语有响应时间、并发用户数、吞吐量、吞吐率、TPS（每秒事务响应数）、性能计数器等。

1.2.1 响应时间

响应时间是指应用系统从发出请求开始到客户端接收到所有数据所消耗的时间。该定义强调所有数据都已经被呈现到客户端所花费的时间，为什么说是所有数据呢？因为用户体验的响应时间带有主观性，用户认为从提交请求到服务器开始返回数据到客户端的这一段时间为响应时间。

以一个 Web 应用的页面响应时间为例，从客户端发送请求到服务处理完成的整个过程如图 1-3 所示。从图中可以看到，页面的响应时间可分解为“网络传输时间”（$N_1+N_2+N_3+N_4$）和“应用服务器处理的时间”（A_1+A_3），“数据库处理的时间”为 A_2，所以整个 Web 页面请求的响应时间为 $N_1+N_2+N_3+N_4+A_1+A_2+A_3$。

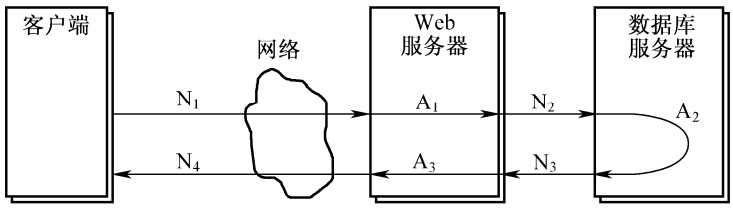

图 1-3 Web 页面响应时间分解

1.2.2 并发用户数

并发用户数是指同一时刻与服务器进行数据交互的所有用户数量。概念中有两点需要注意。第一，同一时刻，因为并发强调的是用户同时对服务器进行施压。例如，一个人同时挑两件东西，这时表示两件东西同时被这个人挑起来，而如果是先挑一件，再挑另外一件，那么就无法表现出同时的概念，这两件东西也就没有同时施压在这个人身上。第二，强调要与服务器进行数据交互，如果未和服务器进行数据的交互，这样的用户是没给服务器带来压力的。同样是上面的例子，这个人虽

然同时挑了两件东西，但其中有一件东西是没重量的，那就是说只有一件东西对这个人造成了压力。

因此对于并发用户这个概念的理解经常会出现以下两个误区：一是认为系统所有的用户都叫并发用户；二是认为所有在线的用户都是并发用户。在线用户不一定是并发用户，原因是在线用户不一定就与系统进行了数据的交互，例如，如果一些在线用户只是查看系统上的一些消息，那么这些在线用户不能作为并发用户计算，因为这些用户并没有与系统进行数据交互，不会给服务器带来任何压力。

那么并发用户数如何计算呢？目前并没有一个精确的计算公式，很多情况下都是根据以往的经验进行估算。根据行业的不同，并发用户数也会有所不同，像电信行业并发用户数为在线用户的万分之一，如果有 1000 万在线用户，那么需要测试 1000 个并发用户。OA（办公自动化）系统的并发用户数一般是在线用户的 5%～20%，所以并发用户数很大程度上是根据经验和行业的一些标准来计算的。

一般情况下，可以参考以下方法来确定性能测试时的并发用户数：

（1）参考其他同类产品。

如果不知道测试过程中需要测试多少并发用户数，那么可以分析市场上同类产品测试的情况，参考其测试的并发用户。

（2）分析历史数据。

如果有历史数据，可以分析后台统计到的历史数据，分析一年或半年的交易量，可以得到服务器每天需要处理的业务数量，进而可以确定系统需要支持的并发用户数。

（3）试上线运行。

如果没有同类产品可以参考，那么试上线运行也是一种方法，如火车票官方订票系统，这类系统就没有可参考的对象，这样可以通过试上线运行的方法来大体了解终端用户提交业务的情况，进而确定系统每秒钟需要处理多少笔业务或 HTTP 请求数。

1.2.3 吞吐量

在性能测试过程中，吞吐量是指单位时间内服务器处理的字节数，吞吐量的单位为 B/s，吞吐量的大小直接体现服务器的承载能力。

吞吐量作为性能测试过程中主要关注的指标之一，它与虚拟用户数之间存在一定的联系，当系统没有遇到性能瓶颈时，可以采用下面的公式来计算：

$$F = \frac{N_{VU} \times R}{T}$$

式中，F 为吞吐量，N_{VU} 为 VU（Virtual User，虚拟用户）的个数，R 为在 T 时间内每个 VU 发出的请求字节数，T 为性能测试所用的时间。但是如果系统遇到性能瓶颈，这个公式就不再适用。吞吐量与 VU 之间的关联图如图 1-4 所示。从图中可以看出，吞吐量在 VU 增长到一定数量时，软件系统出现性能瓶颈，此时吞吐量的值并不会随着 VU 数量的增加而增大，而是趋于平衡。

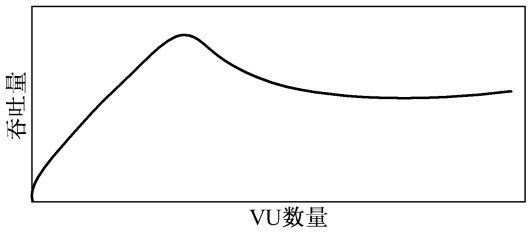

图 1-4　吞吐量与 VU 数量的关系

但在实际测试过程中，测试前吞吐量是不知道的，必须通过不断添加虚拟用户来测试，才能找到吞吐量的拐点，亦即服务器吞吐量的最大值。

实例：假设向一个水池注水，每根小管注入的水量为 $0.1m^3/s$，而水池的出水量为 $1m^3/s$（假设这个值在测试之前是不知道的），如图 1-5 所示。

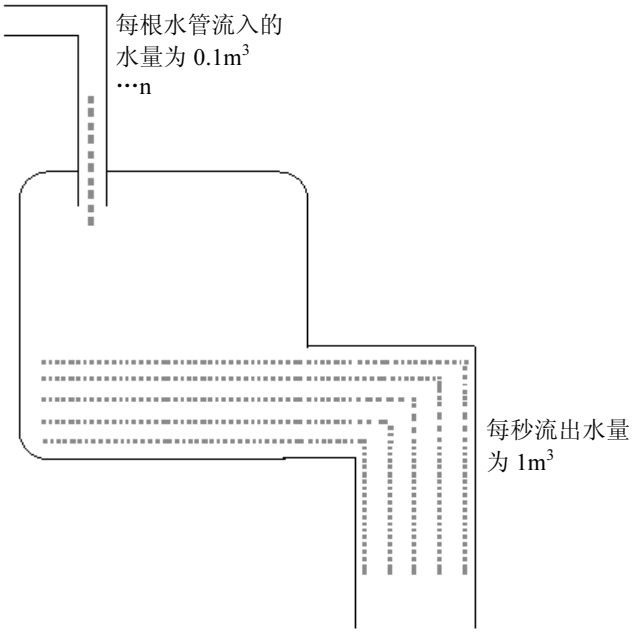

图 1-5　水池吞吐量

当只放一根注水管时，水池流出的水量为 $0.1m^3/s$，亦即当前水池的吞吐量为 $0.1m^3/s$，依此类推，当放入 10 根注水管时，水池的流出水量为 $1m^3/s$，当放入 11（或大于 10 根）根注水管时，水池的流出水量也为 $1m^3/s$，此时水池会开始积水，因为每秒排出的水比注入的水少，这说明水池的吞吐量为 $1m^3/s$，不管注入的水量为多少，这个值都不再改变。服务器也是一样的，客户端不断请求，当服务器能正确处理时，测试出服务器的吞吐量就会不断增加，但当服务器无法处理时，吞吐量的值就不再变化，此时即找到服务器最大吞吐量的值。

1.2.4 吞吐率

吞吐率（Throughout）是指单位时间内从服务器返回的字节数，也可以指单位时间内服务器处理客户提交的请求数。它是衡量网络性能的一个重要指标。吞吐率=吞吐量/测试时间，通常情况下吞吐量的值越大，吞吐率的值也越大，吞吐率的值越大系统的负载能力越强。

1.2.5 TPS

TPS（Transaction Per Second）表示服务器每秒处理的事务数，它是衡量系统处理能力的重要指标。如果每个事务对应的为一笔业务，那么 TPS 即表示服务器每秒钟处理的业务笔数，处理业务数的值越大说明服务器的处理能力越强，如银行业务，一些银行业务在性能测试时使用的是 Windows Sockets 的协议，这样每个事务对应的就为一笔业务。

1.2.6 点击率

点击率（Hit Per Second）是指每秒钟用户向服务器提交的 HTTP 数量。用户每点击一次，服务器端就要对用户提交的请求进行一次处理，从事务的角度来说，如果把每次点击作为一次提交事务来对待，那么点击率与 TPS 的概念是等同的。对于 Web 系统来说，“点击率”是服务器处理的最小单位，点击率的值越大，说明服务器端所需要承受的压力越大（如果服务器正确地处理客户端的请求）。因此通常情况下，Web 服务器都具有防刷新的机制，因为客户每刷新一次系统就要响应一次点击，如果不对服务器进行防刷新处理，当用户不停地点击“刷新”按钮时服务器将承受巨大的压力。

需要注意的是，点击一次并不代表客户端只向服务器端发送一个 HTTP 请求，客户每点击一次都会向服务器端发出多个 HTTP 请求，并且点击率仅仅反映的是客户端提交的请求数，不能表现服务器端当前承受的压力，因为客户端提交的请求服务器不一定会全部处理，有可能出现服务器拒绝客户端请求的情况，所以点击率不能直接反映服务器处理请求的能力。

1.2.7 资源利用率

资源利用率是指服务器系统中不同硬件资源被使用的程度，资源使用率=资源实际使用量/总的可用资源量。主要包括 CPU 利用率、内存利用率、磁盘利用率、网络等。资源利用率是分析系统性能指标进而改善性能的主要依据，在配置调优测试的过程中，通过比较配置调优前后系统资源的利用率来判断调优的效果。

1.2.8 性能计数器

性能计数器（Counter）是描述服务器或操作系统性能的一些数据指标。主要是通过添加计数器来观察系统资源的使用情况。性能计数器包括操作系统性能计数器、数据库计数器、应用服务器计数器等。

计数器在性能测试过程中发挥着“监控和分析”的关键作用，尤其是在分析系统的可扩展性和对性能瓶颈进行定位时，计数器的阈值起着非常重要的作用。必须注意的是，一般情况下，单一的性能计数器只能体现系统性能的某一个方面，在性能测试过程中分析测试结果时，必须基于多个不同的计数器进行分析。

在性能测试中常用资源利用率进行横向对比。例如，在进行性能测试时发现，某个资源的使用率很高，几乎达到 100%，假设该资源是 CPU，而其他资源的使用率又比较低，这时可以很清楚地知道 CPU 是系统性能的瓶颈。

1.2.9 思考时间

思考时间（Think Time），也称为“休眠时间”，是指用户在进行操作时，每个请求之间的时间间隔。对于交互系统来说，用户不可能持续不断地发出请求，一般情况下，用户在向服务器端发送一个请求后，会等待一段时间再发送下一个请求，在性能测试过程中使用思考时间来描述这段时间。

在测试脚本中，思考时间为脚本中两条请求语句之间的间隔时间，如以下代码，其思考时间为 5s：

```
web_url("mer_login.gif",
        "URL=http://localhost:1080/WebTours/images/mer_login.gif",
        "Resource=1",
        "RecContentType=image/gif",
        "Referer=http://localhost:1080/WebTours/nav.pl?in=home",
        "Snapshot=t9.inf",
        LAST);
web_concurrent_end(NULL);
lr_think_time(5);
web_submit_data("login.pl",
        "Action=http://localhost:1080/WebTours/login.pl",
        "Method=POST",
        "RecContentType=text/html",
        "Referer=http://localhost:1080/WebTours/nav.pl?in=home",
        "Snapshot=t10.inf",
        "Mode=HTTP",
        ITEMDATA,
        "Name=userSession", "Value={CSRule_1_UID2}", ENDITEM,
        "Name=username", "Value=test11", ENDITEM,
        "Name=password", "Value=1", ENDITEM,
        "Name=JSFormSubmit", "Value=on", ENDITEM,
        "Name=login.x", "Value=43", ENDITEM,
        "Name=login.y", "Value=15", ENDITEM,
        LAST);
```

当前对于不同的性能测试工具提供了不同的函数来实现思考时间，LoadRunner 工具使用的思考时间函数为 lr_think_time()，在实际的测试过程中，如何设置思考时间是性能测试工程师要关心的问题，因为设置不同的思考时间策略，在单位时间内提交的请求数就不一致，服务器的压力也不一样，具体的思考时间如何设置与当前采用的性能测试策略有关，如果是负载测试则可以直接忽略思考时间，如果是压力或可靠性测试则可以根据实际情况设置一个思考时间。一般思考时间设置为3～5s。

1.3 性能测试划分

性能测试划分有很多种，测试方法也有很多种，更确切地说是由于测试方法的不同决定了测试划分的情况，但在测试过程中性能测试的划分没有绝对的界限，常用的有负载测试、压力测试和并发用户测试等。

性能测试的方法主要包括以下几种：

- 负载测试（Load Testing）
- 压力测试（Stress Testing）
- 配置测试 (Configuration Testing)
- 并发测试（Concurrency Testing）
- 可靠性测试（Reliability Testing）
- 基准测试（Benchmark Testing）

1.3.1 负载测试

负载测试（Load Testing）是通过对被测试系统不断地加压，直到超过预定的指标或者部分资源已经达到了一种饱和状态不能再加压为止。就像举重运动员，在举重的过程中不断地增加杠铃重量，直到运动员无法举起。

该方法主要是为了找到系统最大的负载能力，为性能调优提供数据。该测试方法有以下几个特点：

（1）目的：找到系统最大的负载能力。

（2）环境：该方法需要在特定的环境下进行测试。

（3）手段：不断地对系统进行加压，直到系统中部分资源达到极限。

1.3.2 压力测试

压力测试（Stress Testing）是指系统已经达到一定的饱和程度（如 CPU、磁盘等已经处于饱和状态），此时系统处理业务的能力，系统是否会出现错误。

疲劳测试是压力测试的一种表现形式。例如，一个人很累了，但还在持续不停地工作。

该测试方法有以下几个特点：

（1）目的。测试在系统已经达到一定的饱和程度时，系统处理业务的能力。

（2）手段。使用模拟负载等方法，使系统资源达到一个较高的水平。

（3）该方法一般用于系统稳定性测试。

1.3.3 配置测试

配置测试（Configuration Testing）是通过调整系统软硬件环境，了解各种不同环境对系统性能的影响，从而找到系统的最优配置。

该测试方法有以下几个特点：

（1）目的。通过调整环境了解不同因素对系统性能的影响情况，从而找到调优的方法。

（2）手段。通过调整系统软硬件环境，使系统在不同环境下进行性能测试。

（3）该方法一般用于系统调优和规划能力。

1.3.4 并发测试

并发测试（Concurrency Testing）是通过模拟用户并发访问，测试多用户同时访问同一应用、模块或数据，观察系统是否存在死锁、系统处理速度是否明显下降等其他的一些性能问题。

该测试方法有以下几个特点：

（1）目的。当多用户并发访问时，系统是否存在一些可能的并发问题。

（2）手段。模拟多用户同时并发操作。

1.3.5 可靠性测试

可靠性测试（Reliability Testing）是当系统在一定的业务压力下，让系统持续运行一段时间，观察系统是否达到要求的稳定性，此处强调在一定业务压力下持续运行的能力，可靠性测试必须给出一个明确的要求，如系统能够持续无故障运行多少天。

该测试方法有以下几个特点：

（1）目的。测试系统在一定的业务压力下系统可持续运行的时间。

（2）环境。指明系统在一定的业务压力环境下持续运行。

（3）测试过程中要关注系统运行的情况。

1.3.6 基准测试

在一定的软件、硬件及网络环境下，模拟一定数量虚拟用户运行一种或多种业务，将测试结果作为基线数据，在系统调优或者系统评测过程中，通过运行相同的业务场景并比较测试结果确定调优是否达到效果或者为系统的选择提供决策数据。

基准测试主要包括以下两个目的：

（1）度量改善性能测试的情况。

（2）测试并且调优，保证系统达到性能要求或服务协议要求，在这个测试过程中，基准测试与性能测试迭代配合，以确定调优的情况。

1.3.7 各类测试执行阶段

针对以上 6 种性能测试的类型，在研发阶段应该如何安排呢？一般情况下在编码阶段进行并发测试、压力测试和配置测试，因为在编码阶段需要快速地发现性能的问题，编码阶段结束后，系统进入测试阶段，此时更多的是测试系统的稳定性和对系统进行调优，使系统的性能最优化，所以测试阶段主要是进行负载测试、基准测试和配置测试。各类测试执行的阶段如图 1-6 所示。

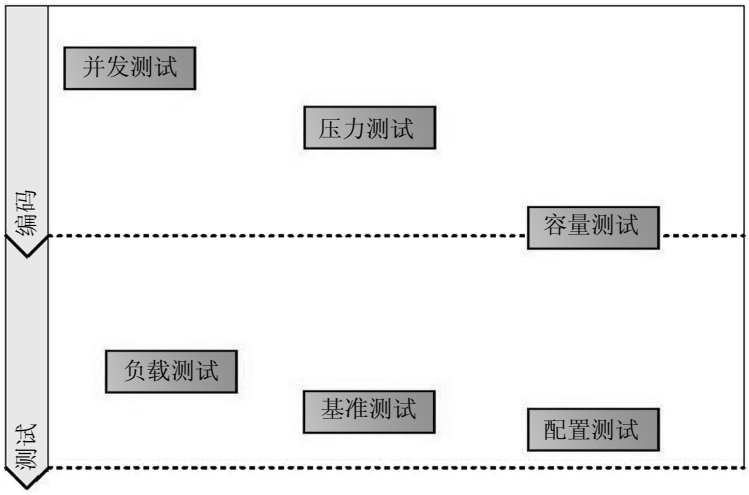

图 1-6 各类测试执行阶段

1.4 性能测试应用领域

上一节讲了常用的性能测试方法，本节将从性能测试的应用领域来讲述性能测试的分类，从应用领域来划分，性能测试分为以下四大领域：

- 能力验证
- 规划能力
- 性能调优
- 缺陷发现

1.4.1 能力验证

能力验证是性能测试最常用的一个领域。一般能力验证采用这样的描述方式：“某系统能否在条件 A 下具备 B 性能”。重点在于验证系统是否具备某种能力。

能力验证领域有以下几个特点：

（1）要求在一个已确定的环境下运行。

（2）需要根据典型场景来设置测试方案与测试用例。

1.4.2 规划能力

规划能力与能力验证有相似之处，但还是存在一些不同的地方，能力验证强调的是在某个条件下具备什么样的能力，而规划能力体现系统如何才能达到要求的性能指标。规划能力问题常常会这样描述：“系统如何才能支持未来用户增长的需要”，这里强调的是未来能力增长的一个需求，着眼于未来系统的规划。

规划能力领域的特点如下：

（1）对系统能力的一种探索性的测试。

（2）可以了解系统的性能及系统性能的可扩展性。

1.4.3 性能调优

性能调优是通过测试来调整系统的环境，最终使系统性能达到最优的状态。这是一个持续调优的过程，主要调优的对象有数据参数、应用服务器、系统的硬件资源等。

一个标准性能调优的步骤如图 1-7 所示。

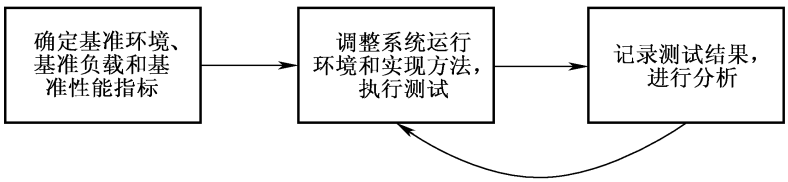

图 1-7 性能调优过程

（1）确定本次性能测试的基准环境、基准负载和基准的性能指标，目的是将这些基准数据作为后期测试数据的参考对象。

（2）对系统进行调优（调优的对象包括代码、数据库、应用服务器、系统资源等），再调整系统运行环境和测试方案，重复进行性能测试，并记录测试的结果。

（3）将调整后的测试结果与基准数据进行比较，以确定调优的效果，重复执行步骤 2 直到性能指标满足要求。

1.4.4 缺陷发现

性能测试应用领域的主要目标是通过性能测试的手段来发现系统存在的缺陷。很多系统在实验室测试环境中没有任何问题，可是当交付给客户时就出现了莫名其妙的错误。如果交付给客户后出现多人同时访问速度缓慢或宕机的现象，那么很有可能是由于系统性能问题所引起。

1.5 小结

本章主要介绍性能测试的基础知识，目的是让读者对性能测试有一个初步的认识。首先介绍了什么是软件性能，了解性能测试需要关注的内容；接着介绍了性能测试过程中常用的术语，也就是性能测试工具中常见的术语；最后介绍了性能测试的划分种类和性能测试应用领域，了解性能测试的分类，帮助确定在性能测试过程中应该如何选择测试的方法。