第一部分

入门篇

入门篇主要介绍的内容有：性能测试基础知识、LoadRunner 基础知识和 LoadRunner 的三大组件。性能测试基础知识部分主要介绍了性能测试过程中一些常见的术语、性能测试划分和性能测试应用的领域；LoadRunner 基础知识部分主要介绍了 LoadRunner 的工作原理、工作过程和内部结构，从全局的角度对 LoadRunner 工具进行介绍；LoadRunner 的三大组件部分主要介绍了 Vuser 发生器、Controller 控制器和 Analysis 分析器的常用操作及工作原理，其使用技巧和高级使用将在提高篇中进行详细的介绍。

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：

- 什么是软件性能
- 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

什么是软件性能？大多数读者朋友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。

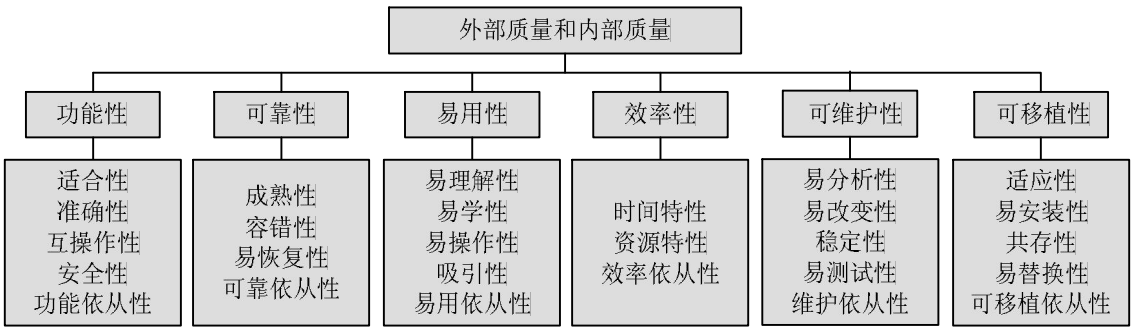

图 1-1　软件质量模型

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）、内存和磁盘等；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，但是不同的人其关注的性能层面有所不同（例如，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三类人：用户、系统管理员与性能测试工程师和软件开发工程师，这三类人各自关注的内容如下：

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。通俗的讲，如果用户单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。

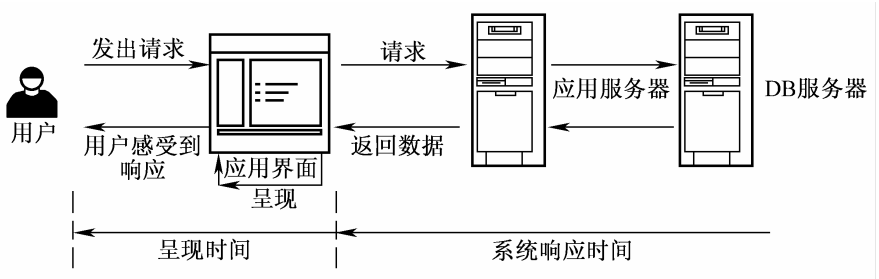

图 1-2　Web 系统响应

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，再慢慢的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. 系统管理员和性能测试工程师

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。系统管理员和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，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，系统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、中间件以及应用服务器等问题。

1.2 性能测试相关术语

了解了什么叫软件性能之后，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。下面介绍性能测试过程中的一些常用术语，性能测试过程中的常用术语有：响应时间、并发用户数、事务响应时间、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器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”（N1+N2+N3+N4）和“应用服务器处理的时间”（A1+A3），“数据库处理的时间”为（A2），所以整个 Web 页面请求的响应时间为（N1+N2+N3+N4+A1+ A2+A3）。

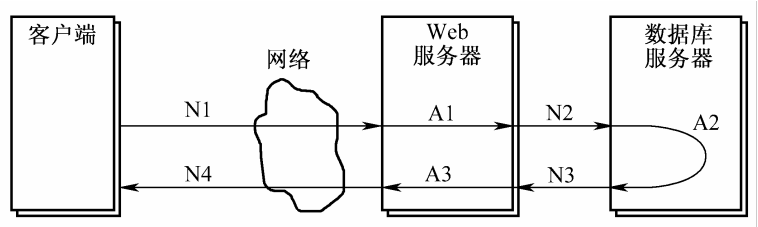

图 1-3 Web 页面响应时间分解

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，表示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种误区：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%左右，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。

一般情况下我们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，得到服务器每天需要处理的业务数量，进而确定系统需要支持的并发用户数。

3）上线试运行。

如果没有同类产品可以参考，那么上线试运行也是一种方法，如火车票官方订票系统，这类系统就没有可参考的对象，通过上线试运行的方法可以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，吞吐量的单位为字节/秒，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，可以采用下面这个公式来计算。

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

其中，F 表示吞吐量；N_{VU} 表示 VU（Virtual User，虚拟用户数）的个数；R 表示在 T 时间内每个 VU 发出的请求字节数；T 表示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。

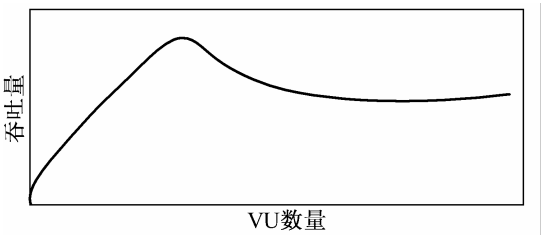

图 1-4　吞吐量－VU 数量关系图

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不断的测试，才能找到吞吐量的拐点，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 0.1 立方米每秒，而水池的出水量为 1 立方每秒（假设这个值在测试之前是不知道的），如图 1-5 所示。

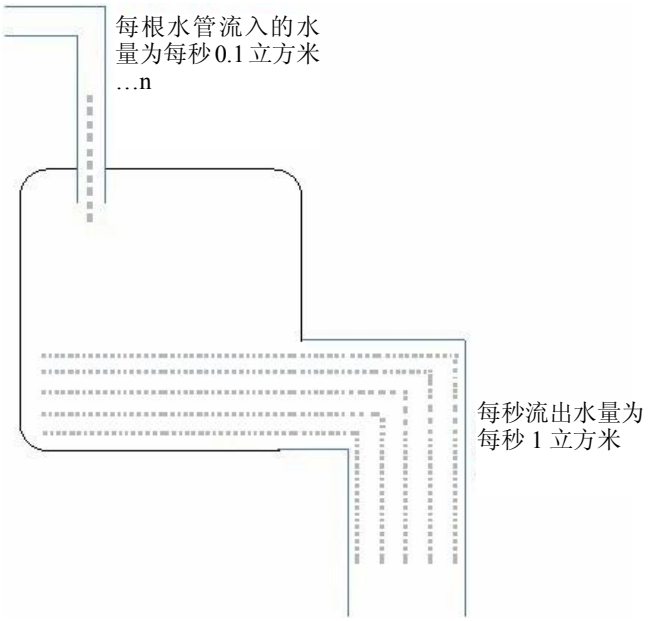

图 1-5　水池吞吐量

当只放一根注水管时，水池的流出的水量为 0.1 立方米每秒，即当前水池的吞吐量为 0.1 立方米每秒，依此类推当放入 10 根注水管时，水池的流出的水量为 1 立方米每秒，当放入 11 根（或大于 10 根）注水管时，水池的流出的水量也为 1 立方米每秒，此时水池就会开始积水，因为每秒排出的水比注入的水少。说明水池的吞吐量为 1 立方米每秒，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，测试出来的服务器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（Throughput）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。用户每单击一次，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，如果把每次单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单击，如果不对服务器进行防刷新处理，当用户不停地单击刷新按钮，此时服务器将承受巨大的压力。

需要注意的是，单击一次并不代表客户端只向服务器端发送一个 HTTP 请求，客户每单击一次，都会向服务器端发出多个 HTTP 请求，并且点击率仅仅反应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应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、网络利用率等。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。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 5 秒。

```
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集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 3-5 秒。

1.3 性能测试划分

性能测试的划分有很多种，测试方法也有很多种，更确切的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，常用的有压力测试、负载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），此时测试系统处理业务的能力，以及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的每次迭代配合，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、压力测试和可靠性测试，因为在编码阶段我们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，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。

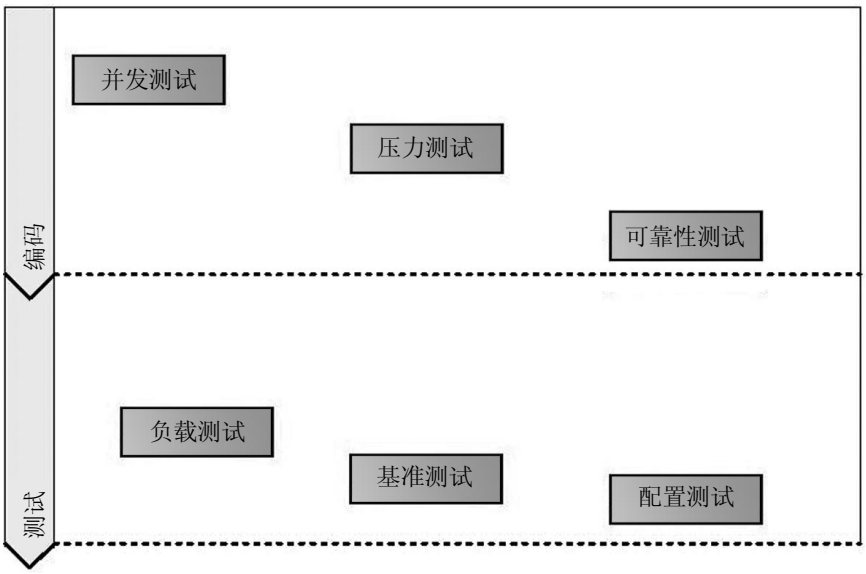

图 1-6 各类测试执行阶段

1.4 性能测试应用领域

上一小节讲了常用的性能测试方法，本小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是：

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。

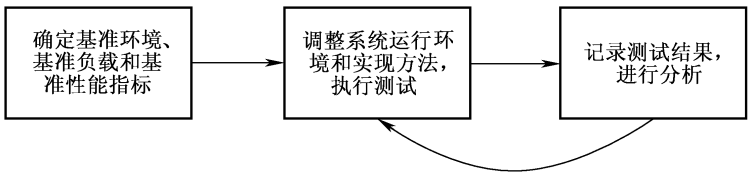

图 1-7 性能调优过程

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，目的是让读者朋友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，也即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，能帮助确定在性能测试过程应该如何选择测试的方法。

2

LoadRunner 基础知识

在使用 LoadRunner 进行性能测试时，需要先了解 LoadRunner 的工作原理、工作过程和内部结构，这样可以对 LoadRunner 有一个整体的了解，对 LoadRunner 有一个概要的认识。最后简单介绍了使用 LoadRunner 进行性能测试的过程，对于性能测试的详细过程在后面的章节中将会介绍。

本章主要包括以下内容：

- LoadRunner 简介
- LoadRunner 工作原理
- LoadRunner 工作过程
- LoadRunner 内部结构
- LoadRunner 性能测试步骤

2.1 LoadRunner 简介

LoadRunner 是一种预测系统行为和性能的负载测试工具，以模拟上千万用户并发负载并实时监测系统性能的方式来确认和查找问题。LoadRunner 能够对整个企业架构进行测试。通过使用 LoadRunner，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

目前企业的网络应用环境都必须支持大量用户同时访问，网络体系架构中包含各类应用环境及由不同供应商提供的软件和硬件产品，难以预知的用户负载和愈来愈复杂的应用环境使公司不得不担心会发生用户响应速度过慢、系统崩溃等问题，从而导致公司收益的损失。Mercury Interactive（2006 年被 HP 收购）的 LoadRunner 能让企业保护自己的收入来源，在无需购置额外硬件的情况下最大限度地利用现有的资源，并确保终端用户在应用系统的各个环节中对其测试应用的质量、可靠性和可扩展性都有良好的评价。

LoadRunner 具有以下几个特点：

1. 按需生产工作量

HP LoadRunner 能够驱动成百上千个虚拟用户，执行不同的业务流程，模拟已部署应用程序将要面临的生产条件。这有助于在投入使用前发现性能和可扩展方面的瓶颈，从而防止它们在投入生产时表现出来；还有助于极大地降低生产停机时间和不良性能，更加容易达到服务等级和正常运行要求。

2. 企业环境支持

实现 Virtual Users 后，测试过程便自动化了。HP LoadRunner 提供广泛的测试环境，支持多种协议和平台。HP LoadRunner 现在支持 60 种以上的协议。其中包括 Web、J2EE、.NET、XML、SAP、Siebel、Oracle、PeopleSoft、无线、Citrix 和客户端/服务器应用程序。因部署的应用程序类型从客户端/服务器变为 Web 和 Java，故可使用相同的工具来进行性能测试。它提供一个一致的工具和一套员工技能，即使应用程序随时间而改变也如此；它也使得降低总体拥有成本（TCO）成为可能。

3. 企业监控支持

HP LoadRunner 拥有非侵入性的实时性能监视程序，可提供被测系统所有部分的详细指标，包括 Web 服务器、应用程序服务器、数据库、企业资源规划（ERP）和 CRM 系统、防火墙和负载平衡器。HP LoadRunner 可识别硬件局限和软件配置问题，这些问题在其他情况下可能不会被检测到。

4. 诊断

HP LoadRunner 可跟踪、计时处于负载情况下的单独应用程序组件，并可排除故障。您可从缓慢的最终用户交易着手，深入查明导致变慢的瓶颈方法或 SQL 语句。这种详细的结果有助于每个负载测试向开发人员提供最终可采取的行动，减少优化 J2EE、Siebel 和 Oracle 部署所需的成本和时间。

5. 自动分析

HP LoadRunner AutoCorrelation 向导会自动整理所有的监控和诊断数据，并计算导致性能降低的最主要的五个原因。可将性能测试结果转化为可处理的精确数据，从而使开发团队大大减少解决时间，并允许执行更多的测试周期。这会帮助您将高质量的应用程序投入生产。

6. 简易使用

HP LoadRunner 是从底层为 QA 用户构建的。它提供可视化脚本语言、数据和 AutoCorrelation 向导以及 ActiveScreen 技术，使得编写脚本和运行负载测试简单易行。因此带来更短的起步时间、更快的 ROI 以及在数周培训之内就能进行性能测试。

7. 高度可扩展性

HP LoadRunner 对于在有限的硬件条件下的高度可扩展性来说，每个虚拟用户需要较低的 CPU 和内存资源。这有助于降低实施过程中潜在的硬件成本。

8. 统一的脚本引擎

HP LoadRunner 与 HP Business Availability Center 软件具有相同的脚本引擎。这将帮助您降低培训成本、脚本开发成本以及 HP 软件的 TCO（Total Cost of Ownership，总体成本）。

2.2 LoadRunner 工作原理

第 1 章介绍了性能测试的一些基础知识，本小节主要讲述 LoadRunner 的工作原理，从宏观的角度了解 LoadRunner 的总体框架、LoadRunner 四大组件是如何工作的以及它们之间是如何通信的。

LoadRunner 由四大组件组成：VuGen 发生器、控制器、负载发生器和分析器。各组件之间的关系如图 2-1 所示。

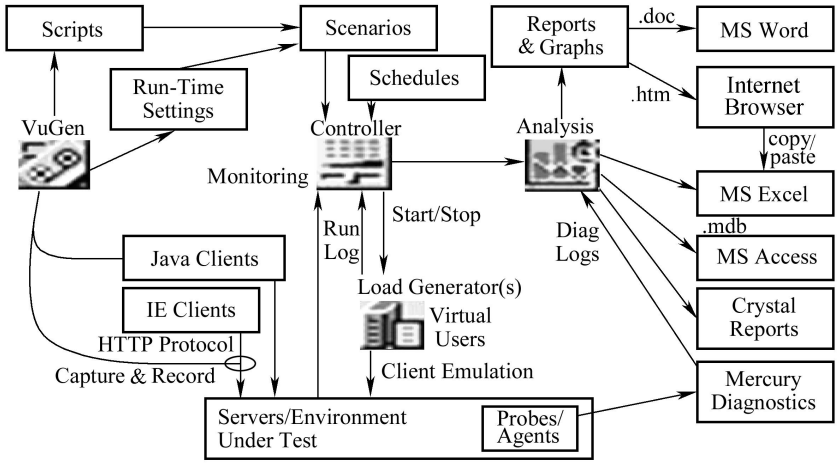

图 2-1　总体架构原理图

1）VuGen 发生器（Visual User Generator，VuGen）：它的作用是捕捉用户的业务流，并最终将其录制成一个脚本。在录制脚本前首先选择一种协议（脚本录制前如何选择协议的内容在后面章节中会详细讲述）；接着在客户端模拟客户实际使用过程中的业务流程，并录制成一个脚本；然后编辑脚本和设置 Run-Time Settings 项；最后 VuGen 通过对脚本的编译生成一个没有错误的可运行的脚本。

2）控制器（Controller）：控制器部分包含两大作用。第一：场景设计。脚本编辑完成后，必须对脚本如何运行设计一种策略。场景设计主要包括手动场景设计和目标场景设计两种方式。第二：场景监控。控制器可以实时监控脚本运行的情况。可以通过添加计数器来监控 Windows 资源、应用服务器和数据库使用情况。

场景设计的目的是设计出一个最接近用户实际使用的场景，场景设计越接近用户使用的实际情况，测试出来的数据就越接近真实值。否则测试出来的数据是不可靠的，到系统上线时可能还是会出现性能问题，场景设计也涉及很多技巧，如 IP 欺骗、负载均衡等一些手段。

执行实时监控更多的是对资源使用情况的监控，以检测这些系统资源是否可能存在瓶颈。

3）负载发生器（Load Generators）：模拟用户对服务器提交请求。正常情况下，在性能测试过程中会将控制器和负载发生器分开，即控制器使用一台独立的机器，为什么会这样做呢？因为在进

行脚本编辑时会产生大量的参数化文件，而这些参数化文件会占用系统资源，再者就是运行时会产生大量的日志文件，所以在测试过程中一般都会将控制器与负载发生器分开；最主要的原因是在模拟成百上千的虚拟用户进行性能测试时，每个虚拟用户都是需要消耗系统资源的，如果虚拟的并发用户过多，会导致测试机出现瓶颈。

负载发生器的计算：在测试时，需要计算测试过程中需要使用多少台负载发生器才算是合理的，任何一台计算机所能支持的虚拟用户数都有一个上限值，如果在测试过程中需要测试的并发虚拟用户数超过计算机所能支持的最大虚拟用户数，这时测试的负载机本身也就成为性能的瓶颈。每个虚拟用户运行时需要占用一定的系统内存资源，具体的值可以从官方文档中获得，通过这个值可以计算出一台计算机最多可以支持多少个虚拟用户。例如，假设负载发生器的计算机使用的内存容量为 512MB，在测试过程中每个虚拟用户需要的内存资源大概为 2.5MB，那么这台计算机最多只能支持 200 个虚拟用户并发测试，如果需要测试 400 个虚拟用户并发，那么就需要两台计算机。

需要注意的是当使用多台负载发生器时，一定要保证负载均衡。负载均衡是指在进行性能测试的过程中，保证每台负载发生器均匀地对服务器进行施压。如果负载处于不均衡的情况，那么在测试过程中，有的负载机很忙，而有的负载机又处于很闲状态，这样测试出来的值是不可靠的。

4）分析器（Analysis）：一个数据分析工具，主要用于对测试结果进行分析。Analysis 中提供了很多基础的数据，但是仅仅依靠这些基础的数据是不够的，客户看到这样的报告也不会满意。在这里涉及到很多分析技术，常用的分析技术有：合并、叠加、页面细分和钻取技术等，这些技术在后面会进行详细的描述。Analysis 的另一个优点就是它本身提供了很多报告的形式，包括 XML、Word 等，这是 LoadRunner 做得比较出色的一部分。

以上是 LoadRunner 四大组件的作用和特点。

2.3 LoadRunner 工作过程

LoadRunner 的四大组件有着各自的作用和特点，但只有将这四大组件联系起来，才能更好地去完成性能测试工作，如图 2-2 所示。

第一步：很多朋友可能会错误地认为 LoadRunner 工作的第一步应该是从 VuGen 开始，但其实 LoadRunner 在性能测试过程中首先是从控制器开始的，从名字中不难看出，控制器的意思为整个系统的核心，控制着其运行与停止。控制器包括两部分：场景设计和场景监控，通过设计控制器中的场景设计性能测试脚本运行的策略，同时在脚本运行过程中监控性能测试的相关指标。

第二步：确定执行策略后，控制器将控制负载机去产生压力，模拟成百上千的虚拟用户去运行脚本，那么负载机如何知道需要执行哪些脚本？以及执行脚本的策略呢？在控制器初始化时，控制器会向负载机发送一个二进制文件，在该文件中记录着如何运行脚本的信息。

第三步：控制器收集测试过程中的相关数据，在控制器执行脚本场景时，控制器会收集测试过程中相关的一些数据，并将这些数据保存在 Access 数据库中。

第四步：结果分析。当场景执行测试结束后，会生成一些分析结果的数据，这时测试工程师需要对这些数据进行分析，如果结果能满足需求，那么说明系统性能满足需求，反之，就有可能需要多次进行实验，来找到性能瓶颈并向开发工程师提出解决的建议和性能调优的建议。

一般情况会错误地认为没有负载机，这是因为在测试过程中将 Controller 和负载机设置为同一台机器，而在实际测试过程中并不允许这样做，因为 Controller 在运行脚本时，每个虚拟用户会占用系统资源，这样可能会导致测试机遇到瓶颈，故一般都会将两者分别放在不同的计算机上来执行。

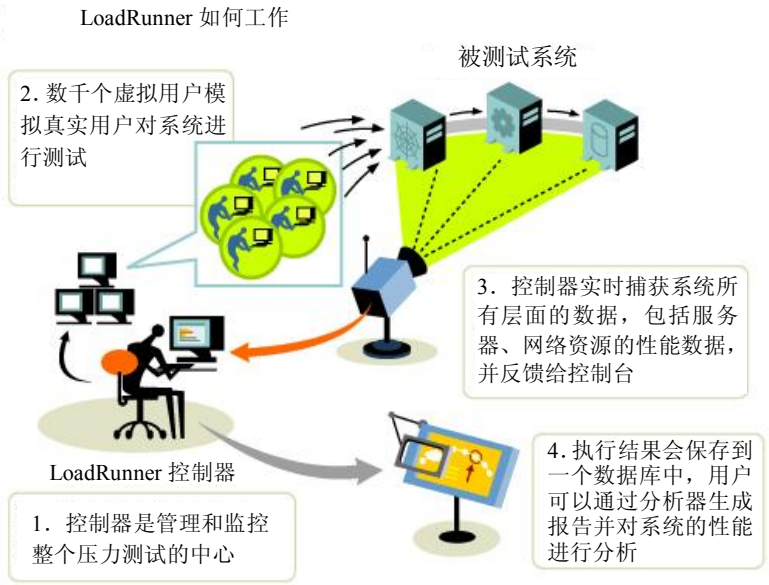

图 2-2　LoadRunner 工作过程

从图 2-2 中可以很清晰地看到四大组件如何相互协调来完成性能测试工作的。

2.4　LoadRunner 内部结构

LoadRunner 主要通过控制内部程序的调度来控制整个性能测试过程，LoadRunner 内部结构图如图 2-3 所示。该图详细地描述了 LoadRunner 执行过程中内部程序是如何调度的及内部各程序之间的关系。

下面从 LoadRunner 内部结构的层次来分析 LoadRunner 性能测试的过程。

1）首先准备好待测试的应用服务器和待测试的系统。

2）LoadRunner 中多线程驱动进程 mdrv.exe 和 r3vuser.exe 模拟产生压力，其中 r3vuser.exe 仿真应用程序的客户端，如 IE 浏览器。它执行了以下三个主要的操作：

①cci（C 语言编译器）建立 ci 文件，然后使用被测系统的协议来执行。

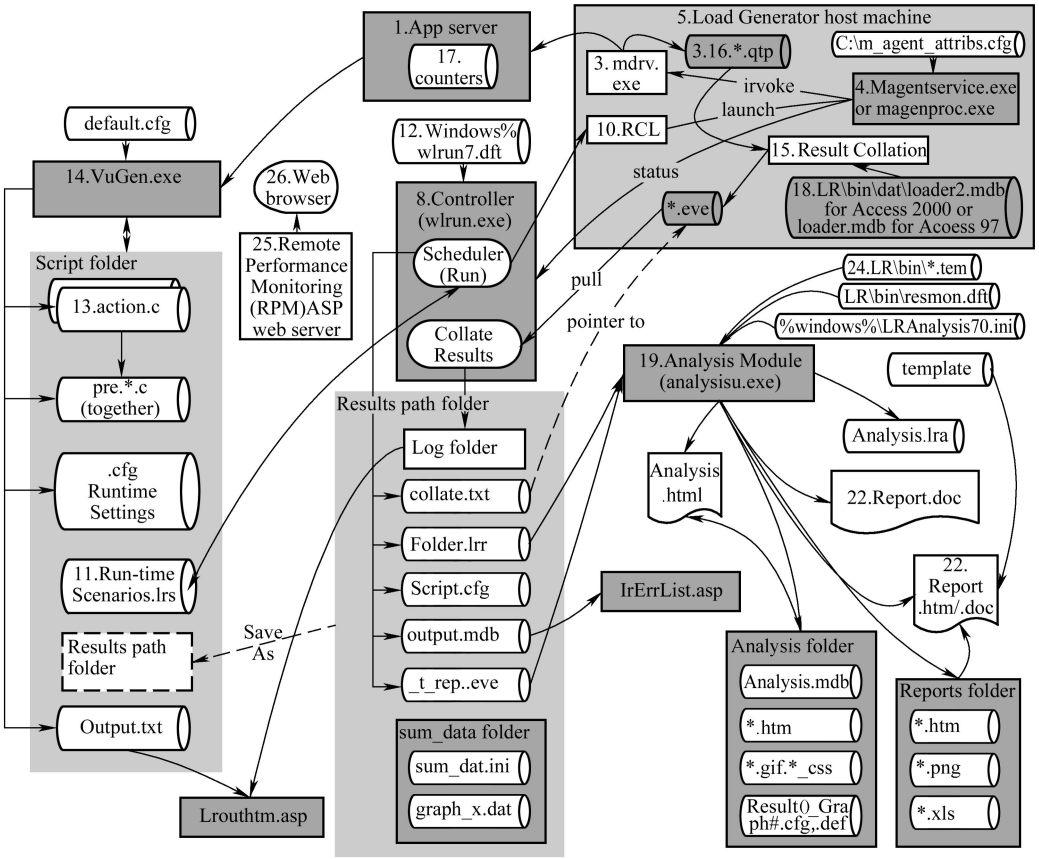

图 2-3　LoadRunner 内部结构图

②通过 Windows 批处理脚本启动 mdrv.exe 程序从而启动 LoadRunner 的运行。mdrv 能自动停止加载 Vuser，因为它们与 Vuser 和 Windows 负载发生器上的 CPU 监控器之间互相通信。

③在 Windows 机器上，对于每一个基于 Java 的 Vuser 都有一个独立的 JVM，注意 UNIX 平台不支持 Java Vuser。

3）虚拟用户在负载发生器端的计算机上使用代理作为服务或进程时，按照组启动方式启动虚拟用户，用户组是多个 Vuser 组成的逻辑集合，在 Vuser 发生器上运行相同的脚本。

4）每个负载发生器（Load Generator）都维护着一个以 qtp 为后缀名的执行日志。

5）日志服务启动后，代理会根据用户组进行隔离，在结果文件中为每个虚拟用户建立一个顺序文件。

6）在执行过程中，这些文件会在“视图”→“显示”输出窗口中显示出来。

7）在预先设置延时上，Controller 上运行的 Scheduler 指导代理（通过 Windows 54345 端口或 UNIX 上的动态端口）初始化场景会话；控制器（wlrun.exe）在发送请求时发送一份场景的拷贝。

8）代理是由每一个负载发生器上的 Remote Agent Dispatcher 进程（8.0 叫 Remote Command

Launcher（RCL））启动的。

9）每个代理根据场景（.lrs）定义文件来决定哪个虚拟用户组和脚本需要在主机上运行，这就是说控制器可以从 DOS 批处理文件（.batch）中启动。

10）控制器通过使用 Windows 操作系统根目录文件夹里的参数值来启动，因为 LoadRunner 被设计成在一个机器上并且一次只能运行一个控制器实例，所以需要使用 Windows 文件夹。

为了在几个应用之间快速地切换，Controller 工作之后会保存在 LoadRunner 的 ini 文件中，然后使用记事本来制作一个批处理文件，在执行 wlrun 之前拷贝应用程序的指定版本的 ini 文件。

11） 在 Vuser 中定义的每个虚拟用户进行的操作都是 LoadRunner 的 VuGen.exe 生成的，当这个程序启动后，它在 Windows 文件夹下存储了 comparamui.ini 文件来保存[LastTablesUsed]下文件的历史，而[ParamDialogDates]项是由“插入”→“新参数”→“数据”来指定。

12）在运行期间，执行结果存储在一个结果文件夹中。

在结果中设置“为每一个设定执行自动创建结果目录”，这样 LoadRunner 会在每次启动一个场景之后自动产生一个递增的结果名。例如，结果名称 Res1 会自动增长到 Res12 或是 Res11-1，错误信息会写到 Microsoft Access 数据库文件 output.mdb 中。

13）在每一个结果文件夹中，程序自动创建一个 Log 文件夹，在这个文件夹中包含每个组的日志文件，运行结束之后，在 Controller 中查看日志文件，单击按钮然后在组中单击右键，选择“查看 Vuser 日志”。

14）场景运行的时候，监控器在本地维护每个主机的计数器。

15)场景运行结束后，进程处理.eve 和.lrr 结果文件并且在结果文件夹下创建一个临时的.mdb（M）数据库。在处理大数据量的结果时，为了防止错误发生，通常使用（Microsoft Access）数据库文件。

16）分析模块（8,320K analysisu.exe）使用.mdb 数据库中的数据来产生分析图表和报告。

17）每次设定运行后的 LoadRunner 结果文件 result_name.lrr（也称为分析器文档文件），由分析程序来读取并显示百分位图表。

18）默认情况下，LRReport 文件夹被创建在本地分析机器的 My Documents 文件夹下用来存储分析会话文件。

19）可以使用 HTML 格式产生报告。

20）结果文件格式是由.tem 模板文件控制的。

负载测试的结果可以使用 Web 浏览器来浏览。

以上就是 LoadRunner 测试的全部过程。

2.5 LoadRunner 11.5 特性

LoadRunner 当前最新版本为 11.5 版本，在 11.5 版本中主要修改了以下内容：

改进 VuGen 脚本生成器：对 VuGen 脚本生成器界面进行改进，扩展其使用的灵活性并且用户可以自定义体验过程。

增强的功能：

- 新的外观和感觉：灵活的窗格、布局和其他更多的方面；
- 资源管理器解决方案：这样可以更方便管理多个脚本组、并且可以直接对脚本的相关属性进行设置（如参数设置、RTS 设置）、可以查看回放结果；
- 快照功能：提供了多种快照的方式、改善了快照性能、回放时同步捕捉出快照信息、提供在快照中进行检索的功能；
- 改进编辑器：上下文内容关联、对不同类型的代码使用不同的颜色进行标识；
- 调试器：支持 C 语言调试器；
- 搜索与替换：在日志和快照中支持搜索和替换功能；
- 步骤导航：新的步骤导航取代了树视图，提供单一视图的脚本，方便对脚本进行过滤和搜索；
- 新窗格：包括“错误”、“任务”、“输出信息”、“运行数据”和“快照”等；
- 关联社区：社区一体化提供了轻松访问 HP 软件社区的对话和线程；
- AJAX TruClient 协议：新的 AJAX TruClient 技术是基于浏览器的新型虚拟用户生成器，用于支持基于 JavaScript 的现代 Web 应用程序（包括 AJAX）。它嵌入在浏览器中，可以提供互动式录制和脚本撰写，从而无需在脚本撰写过程中编写程序，可以让你在各种级别进行录制和重播，从图形用户界面（GUI）级别到传输和套接字级别（具体取决于可用的技能和所需自定义级别），这有助于更加方便、快捷和健壮地进行脚本撰写，它支持常规 Web 应用程序以及各种 AJAX 工具包，可以更快捷、方便和综合地进行 Web2.0 应用程序测试。新的 AJAX TruClient 协议更好地支持了 FireFox 和 Internet Explorer 9.0 浏览器。
- Web 协议异步支持；
- 改进关联功能；
- 增强 Flex 协议；
- 支持录制移动应用程序；
- 支持数据格式扩展（DFE）；
- 支持.NET 4 框架和协议；
- 增强了 Web 服务器协议。

2.6 LoadRunner 性能测试步骤

LoadRunner 虽然只是一个性能测试工具，但使用它进行性能测试时也有其自身的一个流程。性能测试过程分为四个阶段：设计、构建、执行和分析/诊断/调节。具体的工作流程如图 2-4 所示。

图 2-4 中介绍了系统性能调优的过程，因为进行性能测试的目的是找到系统性能的瓶颈，进而帮助开发工程师对系统性能进行调优，如果不能给开发工程师提出性能调优的有效建议，那么性能

测试是做得不够成功的。实际上项目进行测试过程也是这样，性能调优是一个循环的过程，一般情况下需要经历多次测试才能达到目标能力。

图 2-4　LoadRunner 性能测试过程

四个阶段的任务如下：

- 设计阶段定义待测试的业务流程、业务的平均处理量、业务处理量的最高峰值、组合业务流程、系统的整体用户和响应时间目标。
- 构建阶段涉及设置和配置测试系统及基础设施、使用自动化性能测试解决方案构建测试脚本和负载方案。
- 执行阶段包括运行负载方案和测量系统性能。
- 分析/诊断/调节阶段主要测量系统性能并使负载测试进入下一级别，重点查找问题原因以帮助开发工程师迅速解决问题，并实时调节系统参数以提高性能。

下面对这四个阶段进行详细的描述。

1. 设计阶段

设计阶段是性能测试团队与业务领域的经理合作以收集性能要求的主要业务响应时间。可以将需要关注的问题分为四个方面：业务要求、技术要求、系统要求和团队要求。

业务要求需通过业务分析师或最终用户收集。一个全面的业务要求应该考虑以下问题：

- 应用程序情况：创建系统使用演示，让性能测试团队从整体上了解应用程序如何被使用。
- 业务流程列表：创建关键业务流程列表，以反映最终用户在系统上执行的活动。
- 业务流程列表：创建 Word 文档，以便详细记录每个业务流程的正确步骤。
- 交易列表：汇编业务流程中需要负载测量（如“登录”或“转移资金”等）的关键活动的列表。
- 业务流程图：创建业务流程图，以便描绘业务流程的分支情况。

技术要求应该通过系统管理员和数据库管理员（DBA）进行收集并确认。这些人员可能是企业开发组或运营部门的成员，或同时隶属这两个部门，一个全面的技术要求应该考虑以下问题：

- 环境预排工作：与系统或基础设施团队开展测试架构的预排工作。
- 系统范围会议：举行会议来讨论系统的哪些部分应该排除在测试流程之外，并达成一致见解。

- 生产图：创建生产基础设备的图表，以标记出从 QA 迁移到生产过程中可能影响性能的因素。

收集系统要求至关重要，这些是管控负载测试流程通过/未通过状态的系统高级目标，通常来自与业务部门经理的合作来达成一致意见。一个全面的系统要求应该考虑以下问题：

- 系统在正常和高峰期必须支持的用户数量为多少？
- 系统每秒必须处理的交易量是多少？常用的一种估算方法为 80-20 原理法。
- 对于所有的关键业务交易，可接受的最低和最高的响应时间是多少？
- 用户社区如何连接到系统？
- 生产中需要承载的系统工作量如何？交易组合如何？

最后是团队要求阶段，需要确定性能测试团队成员。提前收集完整的业务、技术、系统和团队要求，是有效和成功地进行负载测试的基础。

2. 构建阶段

在构建阶段，需要将设计阶段所确定的业务流程和工作量转变为可用来推动、可重复、真实负载的自动化组件。可以从两个方面来关注：自动化设置和环境设置。

自动化设置包括一系列由性能工程师执行的序列任务：

第一：制作脚本：将存档的业务流程记录到自动化脚本中。

第二：交易：插入计时器来产生业务所需要的逻辑计时。

第三：参数化：用数据池来代替所有的输入数据（如登录用户名和密码），以便每个虚拟用户使用唯一的数据访问应用程序。

第四：方案：通过为不同的用户组分配不同的脚本、连接和用户行为来创建生产工作量。

第五：监视：确定要监视哪些负载服务器或机器。

环境设置包括组装硬件、软件和数据，这些都是执行成功及真实负载测试所必需的，这可能要与系统人员、DBA、操作人员和业务团队协作。环境设置中最重要的是准备数据，数据来源有两种方式：一是历史数据；二是创建数据。

历史数据即是真实存在的数据，只需要从数据库抽取出来即可。

创建数据则是测试过程中通过一些方法生成批量数据，制作数据的方法通常包括：UltraEdit 结合 Excel 制作数据、数据库、Shell 编程和 Java 编程等。所有创建的数据都应该满足数据模型的要求，否则数据在调用过程中会产生错误。

构建阶段的最终结果是得到一套自动化的方案，可在配置好的可用环境中随意执行。

3. 执行阶段

刚刚接触性能测试的人员，常常误认为执行只是一个单一事件，而事实上，它是一个多步骤的流程，包括多种类型的性能测试。每种类型的测试所提供的信息对于了解发布应用程序的业务风险都是必不可少的。

常见的几类负载测试如下：

- 基线测试：用于验证系统及其周围的环境是否在合理的技术参数下运行。性能测试仅运行 5～10 名用户来对最终用户交易性能进行基线测试，这些测试应该在性能测试流程的

开始和结束时执行，以测量绝对响应时间的提高量。

- 性能测试：可模拟环境中的负载，从而提供有关系统可处理多少用户的信息，这些测试应该模拟平均和高峰时的生产用量，它们应该使用真实的用户行为（如思考时间）、调制解调器模拟和多个浏览器类型，以获得最高的准备度，应该运行所有的监视程序和诊断程序，以便于工作人员最大限度地了解系统的性能降低和瓶颈的相关信息。
- 基准测试：用于在理想的情况下测量和比较每种机器类型、环境或应用程序版本的性能，这些测试是系统进行了可扩展测试后运行的，旨在了解不同架构的性能影响。
- 渗入测试：其目的在于长时间在负载下运行系统，从而检验系统的性能状况。
- 峰值测试：其目的在于模拟一段时间内系统上的峰值负载，以帮助演示应用程序和底层硬件是否能够在合理的时间内处理高负荷。

4. 分析/诊断/调节阶段

在完成负载测试的设计、构建和执行阶段后，项目将进入分析/诊断/调节阶段，这些阶段是实时和反复进行的，负载测试解决方案应该提供有关最终用户、系统级别和代码性能数据的全面信息，同时查找导致性能降低的可能原因，这些信息能使你确定是否已经达到性能目标。

在监控、分析、诊断和调节过程中可以获取大量的信息：

- 监控：性能测试过程中的监控可显示基础设备每个层上所发生的一切，同时会更清晰地提供有关测试中数据库服务器、Web 服务器、应用程序服务器、单个应用程序或流程的信息。监控可快速获取有价值的信息，例如应用程序服务器的处理器（CPU）只能支持 150 名用户并发，远低于目标值。
- 分析：完成负载测试后，可将各种指标（如虚拟用户、CPU 或服务器）关联起来，以获取有关应用程序行为不端的其他信息。
- 诊断：高效的性能测试解决方案应该向性能工程师提供有关层、组件、SQL 语句是如何影响负载条件业务流程整体性能的单个统一视图，性能工程师应该能够看到由最终用户交易所接触到的所有组件，然后确定各组件使用的处理时间，以及调用的次数。有了这些信息，就可以针对 Web 服务器、应用程序服务器和数据库服务器瓶颈进行调优。
- 许多企业都在应用程序部署前、中和后三个阶段进行自动化性能测试。有些自动化性能测试解决方案可系统地识别并分离基础设施性能瓶颈，然后通过修改系统配置设定来解决它们，通过反复解决基础设施瓶颈，可以不断改进配置。

2.7 小结

本章主要介绍性能测试工具 LoadRunner 的工作原理、工作过程，以及内部结构，目的是对 LoadRunner 有一个总体的了解，帮助读者后面更好地学习 LoadRunner；接着介绍了 LoadRunner 最新版本 11.5 版的一些特性和新增的功能；最后介绍了性能测试的步骤，但并未对这部分内容进行详细的介绍，在后面的章节中将会专门介绍性能测试的过程。

3 Vuser 发生器

Vuser 发生器（Visual User Generator，简称为 VuGen）主要通过捕获客户端向服务器发送的 HTTP 请求，将这些请求录制成脚本，在回放时将捕获的 HTTP 请求再次发送，以达到模拟客户的行为的目的，所以 Vuser 主要用来捕获最终用户业务流程和创建自动化测试脚本，即生成测试脚本。VuGen 是录制测试脚本、编辑与完善测试脚本的一个平台，支持 C 语言语法。

本章主要包括以下内容：

- 脚本录制
- Recording Options 设置
- Run-Time Settings 设置
- 脚本完善

3.1 脚本录制

启动 Visual User Generator，创建一个新的脚本，开始录制脚本，在录制脚本过程中，VuGen 会自动捕获操作过程中客户端与服务器端进行通信的所有数据。这里涉及的关键点是如何选择录制协议。

脚本开发主要包括四大步骤：计划、录制脚本、脚本增强和单机调试脚本，如图 3-1 所示。

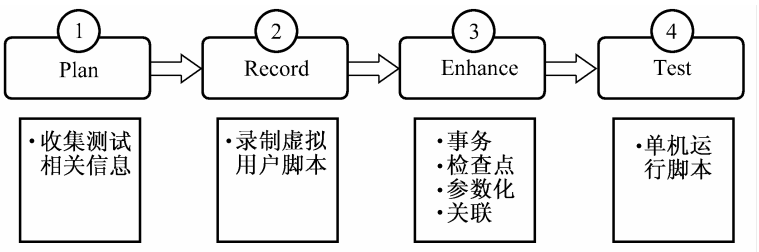

图 3-1　脚本开发过程