性能测试兵法

在大多数的性能测试工作中，我们可以看出很多内容都是互相关联的。这就给我们提供了一个思路：性能测试的很多内容可以经过一定的组织统一来进行。统一开展性能测试的巨大好处是可以由浅入深按照层次对系统进行测试，进而减少不必要的工作量，以实现节约测试成本的目的。为此，本文提出了“全面性能测试模型”的概念。

“全面性能测试模型”提出的主要依据就是一种类型的性能测试可以在某些条件下转化成为另外一种类型的性能测试，而这些类型的测试实施也是很类似的。例如：针对一个网站进行测试，模拟 10 到 50 个用户就是在进行常规性能测试，用户增加到 1000 乃至上万就变成了压力/负载测试。如果同时对系统进行大量的数据查询操作，就包含了强度测试。

1 全面性能测试模型

在“全面性能测试模型”中，把 Web 性能测试分为八个类别。下面首先介绍八个性能测试类别的主要内容。

（1） 预期指标的性能测试：系统在需求分析和设计阶段都会提出一些性能指标，这些指标是性能测试要完成的首要工作之一，本模型把预先确定的一些性能指标的测试称为预期指标的性能测试。

这些指标主要是指诸如“系统可以支持并发用户 1000”、“系统响应时间不得高于 10 秒”等在产品说明书等文档中中十分明确的内容，对这种预先承诺的性能要求，测试小组应该“首当其冲”完成这类测试。

（2） 独立业务性能测试：独立业务主要是指一些核心业务模块，这些模块通常具有功能比较复杂、使用比较频繁、属于核心业务等特点。这类特殊的、功能比较独立的业务模块始终都是性能测试重点。我们通常不但要测试这类模块的一些和性能相关的算法，还要测试这类模块对并发用户的响应情况。

核心业务模块在需求阶段就可以确定，在系统测试阶段开始单独

测试其性能。如果是系统类软件或者特殊应用的软件，通常从单元测试阶段就开始进行测试，在后继的集成测试、系统测试、验收测试中进一步进行测试，以保证核心业务模块的性能稳定。

用户并发测试是核心业务模块的重点测试内容，“并发”的主要内容是模拟一定数量的用户同时使用某一核心模块的“相同”或者“不同”的功能，并且持续一段时间。对“相同”的功能进行并发测试分为两种类型，一类是在同一时刻进行完全一样的操作，例如打开同一条数据记录进行查看；另外一类是在同一时刻使用完全一样的功能，例如同时提交数据进行保存。可以看出后者是包含前前者的，后者是前者的特例，这种并发测试都要持续一定的时间。

从微观角度讲，同时使用某一核心模块“不同”的功能，也是一种组合业务性能测试，只不过这种组合的相关业务大分类是一致的。

（3） 组合业务性能测试：通常不会所有的用户只使用一个或者几个核心业务模块，每个功能模块都可能被使用到，所以 Web 性能测试既要模拟多用户的“相同”操作(这里的“相同”指很多用户使用同一功能)，又要模拟多用户的“不同”操作(这里的“不同”指很多用户同时对一个或者多个模块的不同功能进行操作)，对多个业务进行组合性能测试。组合业务测试是最接近用户实际使用情况的测试，因而是性能测试的核心内容。我们通常按照用户的实际使用情况来模拟使用各个模板的人数比例。

由于组合业务测试是最反映用户使用系统情况的测试，因而组合测试往往和服务器（操作系统、Web 服务器、数据库服务器）性能测试结合起来，在通过工具模拟用户行为的同时，还通过测试工具的监控功能采集服务器的计数器信息，进而全面分析系统的瓶颈，为改进系统提供有利的依据。

用户并发测试是组合业务测试的核心内容。“组合”并发的突出特点是分成不同的用户组进行并发，每组的用户比例要根据实际情况来进行匹配。组合业务测试可以理解为包含了“核心业务模块并发”和“非核心业务模块并发”同时进行的并发用户测试。

（4） 疲劳强度性能测试：疲劳强度测试是在系统稳定运行下模拟较大的用户数量、并长时间运行系统的测试，通过综合分析执行指标和资源监控来确定系统处理最大业务量时的性能，主要目的是为了测试系统的稳定性。

（5） 大数据量性能测试：大数据量测试分为两种：一种是针对某些系统存储、传输、统计查询等业务进行大数据量的测试,主要是测试数据增多时的性能情况，这类一般都是针对某些特殊的核心业务或者一些日常比较常用的组合业务的测试。

第二种是极限状态下的数据测试，主要是指系统数据量达到一定程度时，通过性能测试来评估系统的响应情况，测试的对象也是某些核心业务或者日常常用的组合业务。例如系统的数据每年只备份转移一次，可分别选择一个季度、半年、一年作为参考，模拟输入各个时间段的预计数据量，然后测试系统的性能，进而预估系统的性能走向。

由于大数据量仍然是为了测试系统的业务处理能力，因此大数据量性能测试可以独立进行，也可以和前面的独立、组合业务测试结合起来进行，主要由性能测试策略来决定。由于大数据量测试一般在投产环境进行，因此把它单独独立出来，和疲劳强度测试放在一起，在整个性能测试的后期进行。大数据量测试可以理解为特定条件下的核心业务或者组合业务测试。

（6） 网络性能测试：网络性能测试主要是为了准确展示带宽、延迟、负载和端口的变化是如何影响用户的响应时间的。在实际的软件项目中，主要是测试应用系统的用户数目与网络带宽的关系。

（7） 服务器性能测试（操作系统、Web 服务器、数据库服务器）：服务器性能测试分为初级和高级两种形式。“初级服务器性能测试”主要是指在业务系统工作或者进行前面其它种类性能测试的时候，监控服务器的一些计数器信息，通过这些数据对服务器进行综合性能分析，找出系统瓶颈，为调优或者提高性能提供依据。“高级服务器性能测试”一般不由测试人员进行，由专门的系统管理员来进行，例如数据库服务器由专门的 DBA 来进行测试和调优。本文主要讨论

在测试中常用到的“初级服务器性能测试”，既通过工具对服务器资源进行监控的性能测试。

（8） 一些特殊测试：主要是指配置测试、内存泄漏测试一些特殊的 Web 性能测试。这类性能测试或者和前面的测试结合起来进行，或者在一些特殊情况下会独立进行，本文重点来讨论前一种情况，因为后一种情况往往通过特有的工具、较大投入的进行，可以不作为性能测试的范畴来研究。

2 性能测试通用策略

2.1 性能测试策略通用方法

本节主要介绍一下通用的性能测试策略制定方法。性能测试策略一般从需求设计阶段开始讨论制定，策略的内容决定着性能测试工作投入多少资源、什么时间开始实施等后继工作如何安排。其制定的主要依据是“软件自身特点”和“用户对性能的关注程度”两个因素，其中软件的自身特点起决定作用。

软件按照用途的不同分为两大类：系统类软件和应用类软件。系统类软件对性能一般要求比较高，因此性能测试应该尽早介入。应用类软件分为特殊类应用和一般类应用，特殊类应用主要指银行、电信、电力、保险、医疗、安全等领域类的软件，这类软件使用比较频繁，用户较多，一般也要较早进行性能测试；一般类应用主要指一些普通应用，例如办公自动化软件、MIS 系统等，一般应用类软件多根据实际情况决定性能测试策略，比如 OA 系统，可以早开始、也可以最后进行性能测试，这类软件受用户因素影响比较大。

用户一般可以分为四类：即对性能特别关注、中等重视、一般关注、不怎么关注四类。这里这么划分并不意味着用户不关注性能测试人员就可以忽略性能测试。不过，用户如果特别关注性能，测试人员也应该特别重视性能测试。表 1 列出了性能测试策略制定的基本原则。（注意：这里的用户是广义范围的用户，包括所有和我们的产品有利害关系的群体。因而不单单指最终使用产品的用户，这些用户既可以是为我们提出需求的产品部，也可以是公司的董事会，甚至是我

们研发人员自己。）

<table>
<tr><td>软件类别
用户重视程度</td><td rowspan="2">系统类软件</td><td colspan="2">应用类软件</td></tr>
<tr><td></td><td>一般应用</td><td>特殊应用</td></tr>
<tr><td>高度重视</td><td rowspan="4">设计阶段就开始针对系统架构、数据库设计等方面进行讨论，从根源来提高性能；
系统类软件一般从单元测试阶段开始性能测试实施工作，主要是测试一些和性能相关的算法或者模块。</td><td>设计阶段开始进行一些讨论工作，主要在系统测试阶段开始进行性能测试实施。</td><td rowspan="4">设计阶段就开始针对系统架构、数据库设计等方面进行讨论，从根源来提高性能；
系统类软件一般从单元测试阶段开始性能测试实施工作，主要是测试一些和性能相关的算法或者模块。</td></tr>
<tr><td>中等重视</td><td>可以在系统测试阶段的功能测试结束后进行性能测试。</td></tr>
<tr><td>一般重视</td><td>可以在系统测试阶段的功能测试结束后进行性能测试。</td></tr>
<tr><td>不怎么重视</td><td>可以在软件发布前进行性能测试，提交测试报告即可。</td></tr>
</table>

表 1 性能测试策略制定对照表

从表 1 我们可以看出（1）“系统类软件”、“特殊应用类软件”应该从设计阶段开始进行性能测试共，（2）制定性能测试策略的主要依据是由软件的特点来决定，用户的态度对策略会有一定的影响，但不是决定因素。

软件的特点决定性能测试策略另外一个重要原因就是“一般应用类软件”通常耗费资源较少，因此可以通过提高硬件配置，进而改善

运行环境来提高“一般应用类软件”的性能。从硬件方面解决性能问题往往更容易做到，同时可以降低我们的开发成本，不过也不能过分让用户进行较大的硬件投入，否则会降低我们的“客户满意度”。我们调整性能最好的办法还是软硬件相结合。

用户对待系统性能的态度影响性能测试策略，但不起决定作用的根本原因是我们最终要把产品交付给用户来使用，而不是做出来给用户欣赏。因此不管用户是否重视性能测试，即使根本不关心，对于性能要求高的软件产品我们都应该按照测试上面的策略进行合理的安排。同时，如果我们的上帝——用户如果特别重视，这意味着我们需要进行更多的性能测试方面的投入，因为我们有义务使我们的用户满意。

2.2 性能测试策略实例

下面我们可以看一些性能测试策略制定的案例。

案例一：一个银行项目的性能测试策略的制定案例，性能测试策略从立项时开始确定，贯穿整个项目的执行过程。该软件属于特殊应用软件，用户高度重视性能，因而采取的策略是从设计阶段就开始进行性能测试的准备工作，案例具体内容如下：

产品类型	银行卡审批业务系统，使用非常频繁，业务量每年达到 200 万左右，属于银行领域的特殊应用软件。
项目背景	系统属于第二次重新开发，前一开发商在系统开发完成后没有通过性能测试，100 个左右用户并发访问系统时数据库服务器崩溃。因此新的系统从项目启动开始，性能测试成为用户关注的焦点。
用户要求	用户提出性能方面首先过关，否则功能再好也不会投产。
性能测试策略	从系统设计阶段开始进行性能测试准备工作，主要是参加系统的设计、评审。重点讨论了数据库的设计，前一开发商失利的重要原因是数据库设计

	不合理。 系统设计阶段，完成了性能测试方案的设计。 单元测试阶段通过测试工具对一些重要模块的算法进行测试。主要是一些并发控制算法的性能问题，测试对象是一些核心业务模块。 集成测试阶段进行组合模块的测试。 整个系统测试阶段都在进行性能测试，性能测试和功能测试同步进行。对功能测试引起的一些相关修改，立刻进行性能测试。 验收测试阶段时，在用户现场的投产环境进行性能测试，根据测试结果对系统运行环境进行调优，达到较佳的运行效果。

表 2 某银行项目测试制定案例

案例二：一个 OA 系统的测试案例，我们可以看出性能测试策略和案例一差别很大。

产品类型	企业办公系统，用户数目在 300 人以内，主要是一些信息的发布，以及公文流转、收发邮件功能。软件系统的地位属于辅助办公功能。因此该类软件属于一般类型的应用软件，对性能要求不高，性能测试不属于重要工作。
项目背景	已有稳定产品的实施工作。主要是按照客户的个性化需求进行二次开发。
用户要求	客户提出了性能方面的需求：要求要系统响应时间不要过慢，可以满足 500 个用户来使用。
性能测试策略	系统测试阶段开始进行性能测试准备工作，性能测试在功能测试之后进行。主要是评估系统性能，根据测试结果对系统进行一定的优化。 验收测试阶段在用户现场进行测试，根据测试结果进行一定的调优工作，提交测试报告给用户以

	便进行系统验收。

表 3 某 OA 项目测试制定案例

案例三：一个门户系统的测试案例。

产品类型	主要是用于一些单位信息的发布，用户在 50 人以下。因此该类软件属于一般类型的应用软件，对性能要求很低。
项目背景	软件运行的硬件环境较好。
用户要求	用户没有提出具体的要求。
性能测试策略	验收测试阶段在用户现场进行测试，根据测试结果进行一定的调优工作，提交测试报告给用户以便进行系统验收。

表 4 某门户项目测试制定案例

三个案例不足以说明所有的性能测试策略制定的方法，但是通过这三个案例我们对性能测试策略的制定有了更进一步的了解，体会到性能测试策略制定由软件自身特点决定，同时受用户的态度影响。实际上，软件项目的背景、软件运行环境等许多方面都会影响性能测试策略的制定。因此，本节提出的也是基本的参考方案。制定测试策略是十分复杂的工作，最有效的方法就是“从实际出发”，项目的特点千差万别，我们只有把用户当成“上帝”，充分为用户考虑，综合各个方面进行考虑，才可以制定出合理的性能测试策略。

本节介绍了性能测试策略制定的基本思路和方法，读者应该认真体会这些理论，性能测试策略是后期性能测试工作的基础，决定着性能测试工作的投入。读者要充分意识到这一工作的重要性，认识到只有做好了前期的“路线”工作，才可以走对后面的“道路”。

3　Web 性能测试模型使用方法

“全面性能测试模型”是针对 Web 性能测试而提出的一种方法，主要目的是为了比较全面的开展性能测试，使 Web 性能测试更容易

组织和开展。本模型包含了测试策略制定的通用方法和测试用例设计通用方案，测试用例的设计覆盖了应用软件、服务器、操作系统多方面的内容，按照由浅入深的顺利对性能测试进行了合理的组织。

“全面性能测试模型”是一种经很多性能测试项目抽象出来的方法论，主要用来指导测试，它一般不适合具体的性能测试项目，因为任何一个项目都会有它的特定背景。要想通过“Web 全面性能测试模型”做好性能测试工作，首先要制定好性能测试策略，同时还要按照一些基本指导原则来使用“Web 性能测试用例设计模型”的内容。这些原则主要包括如下的内容：

测试策略遵从最低成本原则。Web 性能测试本身是一种高投入的测试，而实际中国内公司通常在测试上进行较低的投入，Web 性能测试只是全部测试工作的一部分，很多项目只能进行一些重要的性能内容，这就决定测试经理制定性能测试策略时在资源投入方面一定要遵从最低成本化原则。最低成本的衡量标准主要指“投入的测试成本能否使系统满足预先确定的性能测试目标”，只要我们经过反复的“测试——系统调优——测试”后，系统符合性能需求并有一定的扩展空间，就可以认为性能测试工作是成功的。反之，如果系统经过测试后不满足性能需求或者满足性能需求后仍然投入资源，都可以认为是不合理的。

策略为中心原则。本原则不但对性能测试工作有效，对其他类型的测试工作都具有同样的指导意义。测试策略不但决定了我们测试用例设计时的主要内容，还决定着我们实施测试工作时如何根据项目的实际情况进行处理，例如：如果项目时间比较紧张，我们完全可以按照测试用例的优先级只执行一部分性能测试用例。因此，性能测试策略应该贯穿整个性能测试过程。

适当裁剪原则。裁剪原则主要是针对性能用例设计而言。Web 性能测试用例设计模型主要是针对电信、银行级的应用而提出的，包含的测试内容比较全面，而这类项目的性能测试一般周期较长、投入较大，作者曾亲身经历过测试周期为一年的银行项目。要想性能测试用

例设计模型在大多数测试项目中使用，必须对测试用例模型包含的内容进行合理的裁剪，这样做的主要目的是为了适合特定项目的测试需求，进而节约测试成本。

裁减的主要依据是性能测试策略，我们根据策略制定方法制定出测试策略，然后依据策略从“五类性能测试用例”中选择适当的内容来编写测试用例。例如有些要求不高的静态门户网站，用户也没有提出性能方面的要求，我们完全可以只测试一下用户并发情况作为系统性能的参考。

完善模型原则。本模型只是作者的工作经验总结，由于不同的性能测试任务都有自己的项目背景，因而需要对模型内容进行不断的调整、补充、完善，才能使之适合更多的性能测试工作。不断完善具体来说就是要在工作中不断总结自己的经验，形成自己的“Web 全面性能测试模型”。只有“自己的”测试模型，才是最符合自己需要的模型。

模型具体化原则。模型具体化主要是指把本模型运用到具体的项目中，是前面所有原则的目标。如果只记住模型的这些条条框框，然后生搬硬套框架来设计测试，只能得到适得其反的结果。要想使模型在 Web 性能工作中发挥作用，只有根据实际项目的特点制定合理的性能测试策略、编写适当的性能测试用例，并在测试实施中灵活的执行测试方案。

综合上面的分析，可以看出模型使用完全可以概括成两个字——“活用”。要想真正做好性能测试工作，最有效的办法就是在掌握基本理论和方法后，在工作中不断的去探索和总结，不断的完善该模型，形成自己的“Web 全面性能测试模型”。

“全面性能测试模型”是一种很有效的做 Web 性能测试的“兵法”，这正是本篇命名为“兵法篇”的目的。“兵法”是“将帅”们“打仗”的必备知识，学会了兵法才可以指挥战争，但是兵法毕竟不能代替具体的“战术”，它只是“打胜仗”的前提条件，具体的“仗”怎么去“打”，仍然要根据具体的战场情况来指挥。因此，具体的 Web

性能测试工作仍然按照项目的自身特点去组织和实施。