软件性能测试
从这里开始

版本更新记录

版本号	说明	作者	修改日期	备注
V0.1	创建	晏宾	2007-5-10	
V1.0.0.0	发布	晏宾	2007-6-6	

目录

第 1 章 性能测试概论

写一点性能测试心得，总结一下前辈们的资料，给后来加入测试行业的朋友们。

注意：本为部分内容引用了一些网络上的资料，由于文章经常被网络转载多次，很多文章无法找到其作者，故在本文中部分文章无法提及其作者；本文未曾引用标有不得转载字样的文章；本文中也有些地方对引用文章做了一些改动，请见谅。

第 1 节 简介

提供符合功能需求但不符合性能需求的软件是不能被接受的。性能测试就是用来检查软件在系统中的运行性能的，性能测试在软件的质量保证中起着重要的作用。

系统的性能是一个很大的概念，覆盖面非常广泛，对一个软件系统而言，包括：执行效率、资源占用、系统稳定性、安全性、兼容性、可靠性、可扩展性等等。

中国软件评测中心将性能测试概括为三个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。通常情况下，三方面有效、合理的结合，可以达到对系统性能全面的分析和瓶颈的预测。

性能测试是通过自动化的测试程序或工具模拟多种正常、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。它对响应时间、事务处理速率和其他与时间相关的需求以及系统资源使用情况进行检查。目标是核实性能需求是否都已满足、是否会存在一些隐藏的性能风险。

第 2 节 定义

- **性能（performance）：**

计算机系统或子系统实现其功能的能力。

对计算机系统或子系统执行其功能的能力的度量。例如，响应时间、吞吐能力、事务处理数。

- **性能测试：**

性能测试是为描述测试对象与性能相关的特征并对其进行评价，而实施和执行的一类测试，如描述和评价计时配置文件、执行流、响应时间以及操作的可靠性和限制等特征。

通俗的讲：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成性能要求。即在特定的运行条件下验证系统的能力状况。

第 3 节 目的

目的是验证软件系统是否能够达到用户提出的性能指标，同时发现软件系统中存在的性能瓶颈，优化软件，最后起到优化系统的目的。

一般分为以下几个目的。

- **评估系统的能力**

一般分为两种情况：一、评估系统能达到什么样的性能；二、给定了性能指标，检测系

统是否能符合要求；

测试中得到的负荷和响应时间等数据可以被用于验证或评估系统的处理能力。

◆ **识别体系中的弱点（检测系统是否存在性能问题）**

受控的负荷可以被增加到一个极端的水平，并突破它，从而修复体系的瓶颈或薄弱的地方。

◆ **系统调优：（已知或未知系统存在性能问题，进行测试去定位其位置。）**

重复运行测试，验证调整系统的活动得到了预期的结果，从而改进性能。

检测软件中的问题：长时间的测试执行可导致程序发生由于内存泄漏引起的失败，揭示程序中的隐含的问题或冲突。

◆ **验证稳定性（resilience）可靠性（reliabIlity）**

在一个生产负荷下执行测试一定的时间是评估系统稳定性和可靠性是否满足要求的唯一方法。

第 4 节 注意事项

1．服务器端和客户端一定要同一个局域网内，否则网络因素会成为性能测试的瓶颈。

2．在性能测试脚本中要注意检查点的设置，否则都不清楚脚本是否真的成功执行操作。

3．设置参数化和关联是性能测试脚本调通的关键。但是要考虑参数化和关联的资源消耗对性能的影响。

4．测试程序中可以加入一些等待时间，以使测试和真实情况的一致。但是统计响应时间时一定要过滤这些时间。

5．尽量统计出每个页面、每个动作的响应时间以及响应时间的细分（域名解析、服务器时间、网络传输时间、客户端时间等）以方便更好的分析测试结果。

6．运行性能测试时关闭日志功能，调试脚本时可以打开日志功能。

7．性能测试前的数据准备： 尽量保证数据库数据和真实环境数据同步或略多于真实环境数据；如果数据是会不断累加的，要考虑软件生命周期内可能的最大数据量。

第 5 节 误区

误区 1：应用程序必须通过功能性测试后才可以测试性能。

应该尽早的进行性能测试。性能测试可以发生在各个测试阶段中，即使是在单元层，一个单独模块的性能也可以使用白盒测试来进行评估，然而，只有当整个系统的所有成分都集成到一起之后，才能检查一个系统的真正性能。

性能测试从早开始，完成一个小模块，对小模块的接口进行性能测试、一般耗费资源很少，但可以防止问题在项目在最后出现、花费很大的精力去修改。

而有些资料中提到的：在系统代码开发和功能测试完成之后，进行性能测试的说法，是为了检查系统整体性能的做法。一般经常出现在验收性性能测试中。

误区 2：性能测试要向功能测试一样，覆盖到所有功能。

性能测试的主要目的是为了系统调优。不可能对所有的系统功能都进行性能测试。在测试设计时需要结合当时的实际系统，先分析软件可能存在的瓶颈，此时可依据 80/20 原则分析：对系统资源的利用、数据大量传输、数据转换、用户使用频率、逻辑复杂度等进行分析，选择要执行的功能和场景，再依此制定性能测试的方案。

误区 3： 系统吞吐率随着并发量增加而增加。

随着并发量的增加吞吐率并不是线性增长的。并发量从小逐渐增大，开始阶段吞吐率随着并发量的增加线性变化；当并发量达到某一值时，系统处理能力趋于饱和（也可能某一软硬件条件达到临界值），此时再逐渐增大并发，会有一些请求处于等待状态，所以响应时间变慢，吞吐率趋于稳定；当并发量达到系统的最大处理能力后，再增加并发，系统处理能力会下降，吞吐率也会下降，最终可能发生宕机。

误区 4：客户给出性能指标，我们一定要想法设法达到。

根据用户提供的指标进行可行性分析，分析这些指标在理想状态下是否可以达到。比如有这么一个需求：有一台服务器，希望能承载 10000 个用户每秒 200kb 的传输。从 Cpu、disk、网卡等方面分析都是很难达到的，也是很难测试的。需要和客户商讨增加硬件配置或者通过其他途径来解决。

误区 5： 压力测试、负载测试、容量测试等这些不同类型的测试一个一个分开来执行。

现实场景是复杂的，测试也需要尽可能的模拟复杂的场景。在一个整体的系统性能测试场景中，应该包括各个类型的测试。而需要检查某一个方面的指标或分析某个性能问题时，尽量保证场景简单、单一、容易模拟。

误区 6：做性能测试主要就是测试工具的使用；我做不好性能测试，是因为对测试工具不熟悉；测试工具可以自动生成我所需要的报表；依靠性能测试工具就能准确定位系统瓶颈；

测试工具在测试中只起到辅助性作用。而测试方案、测试场景的分析、问题的定位这才是性能测试的关键。不要期望测试工具能够生成你想要的所有东西（报表、瓶颈分析），工具只能近可能多的提供给我们分析的依据。

误区 7：在线用户数就是并发用户数。并发用户数高意味着 PV（页面浏览量）大。

并发用户数*用户访问页面数=PV

误区 8：提高一下硬件配置就可以提高性能了，因此性能测试不重要。

随着软件规模的扩大，提高硬件配置只是解决性能问题的一个基本手段。因为如果软件自身存在性能问题，再多的资源可能也不够用，例如：内存泄漏问题，随着时间的增加，内存终究会被耗尽，最后导致系统崩溃；数据库链接数等配置信息、数据库死锁是和硬件很难挂钩的；算法逻辑问题导致程序缓慢。

即使要提高已经配置，也要首先用性能测试的方式得出哪些硬件可能存在瓶颈。

误区 9：性能测试独立于功能测试。

一方面，整体性能测试的场景设计要求的系统功能非常熟悉；另一方面，功能测试可以发现性能问题，性能测试也能发现功能问题。很多性能问题是由软件自身功能缺陷引起的。如果应用系统功能不完善或者代码运行效率低下，通常会带来一些性能问题。功能测试可能会发现这些问题。

误区 10：随便找个环境下进行一下性能测试就可以了。

做性能问题分析可以在类生产环境上进行，配置可以有些差别，但是，整体性性能测试、验收性性能测试要尽量在用户生产环境下进行。

第 6 节 前景

有人说问“性能测试工程师能有 8K 的工资吗？”，回答曰：“我们公司的性能测试工程师都是 1W 起的。”

目前来看，无论是 B/S 结构、C/S 结构软件,性能测试工程师是很受欢迎的，随着 B/S 结构软件的普及，他更会显得尤为重要。

第 7 节 主要因素

影响性能的主要因素：

1. 用户：数目、类型、活动频率、思考时间、访问次数、点击数；；
2. 会话 每个会话页面大小，使用时间长度；
3. 并发问题 缓存交换任务优先级；
4. 吞吐量；
5. 每个请求的逻辑处理；
6. 应用体系架构设计、软件性能；
7. 客户端速度；客户端本地网络速度；
8. 公共网络延迟；公共网络吞吐量、服务器本地网络速度带宽吞吐量；
9. 服务器响应时间、服务器硬件速度、服务器分布模式、系统配置、安全有关的配置;

第 8 节 评测

任何测试的目的都是确保软件符合预先规定的目标和要求。性能测试也不例外。所以必须制定一套标准。

1.8.1 性能测试评测模型

• 线性投射：用大量的过去的，扩展的或者将来可能发生的数据组成散布图，利用这个图表不断和系统的当前状况对比。
• 分析模型：用排队论公式和算法预测响应时间，利用描述工作量的数据和系统本质关联起来
• 模仿：模仿实际用户的使用方法测试你的系统
• 基准：定义测试和你最初的测试作为标准，利用它和所有后来进行的测试结果进行对比。

1.8.2 性能测试评测指标

- 动态监测 - 在测试执行过程中，实时获取并显示正在执行的各测试脚本的状态。
- 响应时间/吞吐量 - 测试对象针对特定主角和/或用例的响应时间或吞吐量的评测。
- 百分位报告 - 数据已收集值的百分位评测/计算。
- 比较报告 - 代表不同测试执行情况的两个（或多个）数据集之间的差异或趋势。
- 追踪报告 -主角（测试脚本）和测试对象之间的消息/会话详细信息。

第 2 章 测试分类

对于性能测试的分类，业界有很多标准，而对每个类型的诠释也有一些差别。我们不必去深究每个类型的确切定义。只需要清楚性能测试涵盖了哪些方面，而在某一次测试中，我们可以清晰的判断应该做哪些方面的性能检查。

全面性能测试包括：预期目标的性能测试、独立业务性能测试、组合业务性能测试、服务器性能测试、网络性能测试、疲劳测试、大数据量测试等等。

第 1 节 常用类型

2.1.1 性能测试

狭义的性能是指软件系统或构件对于其及时性要求的符合程度。及时性用响应时间或吞吐率来衡量。

狭义的性能测试，是一种“正常”的测试，主要是测试正常使用时，系统及时性（响应时间、吞吐率）是否满足要求，同时可能为了保留系统的扩展空间进行一些稍稍超出“正常”范围的测试。

2.1.2 负载测试

负载测试（Load Testing）在给定的测试环境下，通过逐步增加系统负载，直到性能指标超过预定指标或某种资源使用已经达到饱和状态，从而确定系统在各种工作负载下性能容量和处理能力，以及持续正常运行的能力，确定系统所能够承受的最大负载量。负载测试的主要用途是发现系统性能的拐点，寻找系统能够支持的最大用户、业务等处理能力的约束，为系统调优提供数据。

当负载逐渐增加时，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等来决定系统的性能。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接收的性能过程。

负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。 此外，负载测试还要评估性能特征，例如，响应时间、事务处理速率和其他与时间相关的方面 。

◆ **负载测试与压力测试的区别**

负载测试是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，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等来决定系统的性能。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接收的性能过程。压力测试是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

2.1.3 容量测试

容量测试是负载测试的补充，用来确定程序的最终临界点。测试系统能够处理的最大会话能力。确定系统可处理同时在线的最大用户数。

即使系统处理会话超过了临界点，系统仍需要稳定运行。

2.1.4 压力测试

压力测试（Stress Testing）通过逐步增加系统负载，测试系统性能的变化，并最终确定在什么负载条件下系统性能处于失效状态，通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

同时在测试系统的能力最高实际限度时，即软件在一些超负荷的情况，功能实现情况。如要求软件某一行为的大量重复、输入大量的数据或大数值数据、对数据库大量复杂的查询等。

压力测试的目的是找出因资源不足或资源争用而导致的错误。如果内存或磁盘空间不足，测试对象就可能会表现出一些在正常条件下并不明显的缺陷。而其他缺陷则可能由于争用共享资源（如数据库锁或网络带宽）而造成的。压力测试还可用于确定测试对象能够处理的最大工作量。

压力测试并不是简单的为了一种破坏的快感而去破坏系统，实际上它是可以让测试工程师观察系统对出现故障时系统的反应。系统是不是保存了它出故障时的状态？是不是它就突然间崩溃掉了？它是否只是挂在那儿啥也不做了？它失效的时候是不是有一些反应*？在重启之后，它是否有能力可以恢复到前一个正常运行的状态？它是否会给用户显示出一些有用的错误信息，还是只是显示一些很难理解的十六进制代码？系统的安全性是否为因为一些不可预料的故障而会有所降低？

◆ **压力测试的一些反常规操作**

压力测试是在一种需要反常数量、频率或资源的方式下运行系统。例如：

- 当平均每秒出现 1 个或 2 个中断的情形下，应当对每秒出现 10 个中断的情形来进行特殊的测试；
- 把输入数据的量提高一个数量级来测试输入功能会如何响应；
- 应当执行需要最大的内存或其他资源（如 CPU，内存，磁盘，网络）的测试用例；
- 运行一个虚拟的操作系统中可能会引起大量的驻留磁盘数据的测试用例。
- 两倍的已经基线的并发用户数或者 HTTP 连接数
- 随机的关闭及重开连接到服务器上的网络上集线器/路由器的端口（例如，可以通过 SNMP 命令来实现）
- 把数据库断线然后再重启；

2.1.5 强度测试

强度测试检查程序对异常的处理能力。它总是迫使系统在异常的资源配置下运行。压力测试注重的是外界不断施压，强度测试注重的是系统的极限或者系统异常情况下的测试。

2.1.6 并发测试

◆ **定义**

并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程。它重点关注：多个用户同时访问同一个应用、模块或者数据时是否存在思索或者其他性能问题，如内存泄漏、线程锁、资源争用问题。

几乎所有的性能测试都会涉及并发测试。而在并发测试的同时，会兼顾到负载、压力、稳定性等测试。

用户并发测试是性能测试的最主要部分，包含了负载测试和压力测试的过程。主要是逐渐增加用户数量来加重系统负担，直到出现不能接收的性能点或者瓶颈。一般要测试正常数量的用户并发和极限数量下用户并发的情况。

◆ **目的**

并发性能测试的目的在于寻找到瓶颈问题。主要体现在三个方面：

以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，以评价系统的当前性能；

当扩展应用程序的功能或者新的应用程序将要被部署时，负载测试会帮助确定系统是否还能够处理期望的用户负载，以预测系统的未来性能；

通过模拟成百上千个用户，重复执行和运行测试，可以确认性能瓶颈并优化和调整应用；

◆ **场景设计**

并发用户测试主要是对系统的核心功能和重要业务进行测试，要以真实的业务数据作为输入，选择有代表性和关键的业务操作。测试场景包括：

单一场景测试（有人也称作基准测试）：

单一场景测试是逐一对业务模型中的业务或个别重要业务进行单一场景多并发测试，目的是检查单一业务功能的性能和稳定性。

混合场景测试：

混合场景测试是按照业务模型所有业务的多种不同类型操作，所有用户同一时间执行的是相同的操作。通过性能测试，得到系统性能表现数据如：务的平均交易响应时间、应用服务器、数据库服务器的资源使用情况、交易正确率等。

业务组合测试

业务组合测试是按照业务模型所有业务的多种不同类型操作、以及各个业务操作的用户使用数等信息设计测试场景，各个用户执行不同的操作，模拟实际生产环境中在业务处理系统的压力情况。通过性能测试，得到系统性能表现数据如：务的平均交易响应时间、应用服务器、数据库服务器的资源使用情况、交易正确率等，为系统的实际上线运行提供可靠的参考。

测试方法：按照业务模型比例设置测试场景。并逐步增加并发量，记录每次测试环境参数：包括数据库配置参数，应用系统配置参数。收集系统性能变化曲线。

业务组合测试是更接近用户实际操作系统的测试，因此用例编写要充分考虑实际情况，选择最接近实际的场景进行设计。业务组合测试可以执行如下的情况：所有用户同时使用多个模块的测试、模拟多个用户并发来做不同的事情、分用户做同样的操作。

2.1.7 稳定性测试

◆ **定义**

稳定性测试，通过给系统加载一定的业务压力的情况下，让应用持续运行一段时间，测试系统在这种条件下是否能够稳定运行。

在整个软件测试中，软件的可靠性包含很多的内容，比如：成熟性、容错性等等，性能测试里所说的可靠性测试一般和稳定性是类似的。

◆ **场景设计**

稳定性测试主要是为了验证系统是否支持长期稳定的运行。在场景设计上重点考虑以下几个问题：1.涵盖功能点；2.并发量；3.持续时间；4.系统监控。

功能点

如果是为了验证某一特定功能点的稳定性，测试方案可以只涵盖这个功能点或者这个功能点下面的某个操作。而系统整体稳定性检查时，功能点的选择一般是与性能负载压力测试的功能点一致的，这样即节省了测试脚本开发时间，又能保证了测试覆盖。当然，也可以添加一些分析认为在长时间运行极可能出现问题的场景。

并发量

如果是单一场景稳定性测试，根据实际需要调整并发量。在系统整体稳定性测试中，并发量的选择一般是有如下四种形式：

1.以系统的平均并发量或吞吐率为依据，再乘以一个权值；

2.以峰值时的并发量和吞吐率为依据，再乘以一个权值；

3.场景中全部时间是以均值加权值的并发量为基数，在某些时间点或时间段再加入峰值加权值的并发量。这样更大限度的模拟了真实环境；

4. 增加压力使系统资源到达某一特定值（如：使 Cpu 资源在 70-90％的使用率），维持这个压力，持续运行一定时间。

场景执行时间

根据系统的需求而定，一般是保证系统 7*24 或 30*24 小时不间断运行。这个时间根据实际情况来调节。

值得注意的是：一般的业务系统，每周的使用时间是 5*8 小时，峰值出现的时间在和其他产品也不同；互联网产品每天的有效使用时间在早 9 点至晚 9 点，其他时间的系统负载很低。

系统监控

和其他性能测试类型一样，稳定性测试主要是监控系统 CPU、MEM、disk、i/o、网络以及软件系统的日志、数据库状态、请求准确率、响应时间、吞吐量等等。稳定性测试要重点关注这些值的变化趋势。

◆ **作用**

稳定性测试可能帮助找到一些大型的问题，如死机、崩溃、内存泄漏等，因为有些存在内存泄漏问题的程序，在运行一两次时可能不会出现问题，但是如果运行了成千上万次，内存泄漏得越来越多，就会导致系统崩滑。

2.1.8 疲劳度测试

疲劳强度测试是在系统稳定运行下模拟最大用户数量、长时间运行，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。

它和稳定性测试不同，稳定性通常模拟的是平均用户数量（或加权值），而疲劳测试通常是使用峰值时用户数量（或加权值）；在运行时间上，疲劳度测试运行时间以小时为单位，而稳定性测试运行时间以天为单位。如出现错误导致测试不能成功执行，需要及时调整测试指标，例如降低用户数、缩短测试周期等，以综合衡量系统系统。

疲劳强度测试的目的就是检验系统长时间、大用户量运行后的性能，对服务器、软件、网络进行不同条件下的综合测试分析，测试时要记录系统发生故障的信息作为测试结果。

2.1.9 大数据量测试

大数据量测试可以分为三种类型：

针对某些系统存储、传输、统计、查询等业务进行大数据量的独立数据量测试；

与压力性能测试、负载性能测试、疲劳性能测试相结合的综合数据量测试方案；

单独的数据库或文件系统性能测试；

通常来说，我们采用第二种测试方案。把多种测试类型结合在一起，节省测试时间。如果怀疑或者已经发现大数据量情况下存在问题，那么，需要采用方案一、三进行深入测试。大数据量测试的关键是测试数据的准备，如何准备足量而且有效的数据是值得去思考的。

2.1.10 配置测试

配置测试是通过对被测系统的软硬件环境的调整，了解各种不同环境对性能影响的程度，从而找到系统各项资源的最优分配原则。配置测试是系统调优的重要依据。

配置测试和负载、压力测试也是分不开的，测试中要考虑在不同配置环境中，不同负载压力的系统性能状况。

举一个例子：

在互联网产品中，使用 cache 能很大程度上提高系统的性能。那么如何确定 cahce 的各个参数值呢？是使用 memcache、filecache 还是两者结合，各占多大的比例？cache 的大小多少合适？cache 中单个文件的大小应该是多少？cache 的内容占总内容的百分比？不同用户数访问上述配置的 cache 时，相应时间等指标如何变化？以上只是从软件方面分析了几个影响 cache 的因素，那么硬件因素是否需要考虑？

第 2 节 非常用类

2.2.1 峰谷测试

峰谷测试是确定系统从高负载到低负载、甚至空闲，然后再攀升到高负载、再降低的能力。这里的峰值，不是系统所能承载的最大吞吐量。而是，系统正式运行时会出现的峰值（或这个峰值的加权值），峰谷测试一定是在压力测试已经完成并且测试通过后进行的。

测试中需要注意：第二次高峰中各项性能指标是否与第一次的峰值时一致？达到各次峰值和降低到低谷所耗费的资源和时间是否一致？是否会有性能下降的迹象？多次执行峰值峰谷操作，比较这些信息。

2.2.2 异常测试

通过这样的测试来检验如果系统局部发生故障,，用户能否继续正常使用系统。如：一个有负载均衡系统中，部分服务机器宕机，并发访问的用户能够全部使用为宕机的服务。同时也需要考虑，当部分用户宕机时，剩余服务的压力情况。

2.2.3 速度测试

通常是对一些耗时较长的功能进行的专项测试，如下载或传输的速度、创建索引的速度等等。

我们所说的速度测试不是单纯的功能测试，它是检查系统在不同负载状态下，某个功能在不同数据量时的执行速度。最常见测试模型：网站下载测试、IM 工具的文件传输测试等等。

2.2.4 服务器性能测试

把服务器性能测试单独提出来服务器性能对整个系统性能的影响是巨大的，在某些测试中，单独为服务器性能设计测试场景是非常有必要的。

服务器性能场景可以在并发测试、压力测试、负载测试的基础上，加大对服务器各种资源的监控，以便能够更深入的分析服务器性能。也可以单独的设计一些场景测试某些设备的性能。如：设计需要复杂逻辑计算的场景，检测我们设计的逻辑运算在这个 Cpu 上的表现;设计顺序读写、随机读写的场景，检查存储系统、disk 的读写能力等等。

2.2.5 网络性能测试

与服务器性能一样，网络性能对整个系统性能的影响也是很大的，在某些测试中，单独为网络性能设计测试场景是也非常有必要的。

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。可以独立测试，也可以和并发测试、压力测试、负载测试结合起来，在原有的基础上采用工具来设置并监控网络，从而达到监视网络性能的目的。

在系统中通常是单位时间请求数、和每个请求传输的数据量与网络带宽存在着关系。另外，还需要考虑网络协议、网络状态等情况。

2.2.6 防火墙测试

这里提到的防火墙测试不是验证防火墙的安全性，而是测试防火墙对系统性能的影响。验证系统加上防火墙前后,系统的性能变化是否在可接受的范围之内。

测试场景可以参照并发测试、压力测试、负载测试等等。测试过程中统计各项性能指标，测试完成后，比较加防火墙前后，系统性能变化。

第 3 节 其他分类一

来源于网络：中国软件评测中心对性能测试的概括

2.3.1 应用在客户端性能的测试

应用在客户端性能测试的目的是考察客户端应用的性能，测试的入口是客户端。它主要包括并发性能测试、疲劳强度测试、大数据量测试和速度测试等，其中并发性能测试是重点 。

2.3.2 应用在网络上性能的测试

应用在网络上性能的测试重点是利用成熟先进的自动化技术进行网络应用性能监控、网络应用性能分析和网络预测。

2.3.2.1 网络应用性能分析

网络应用性能分析的目的是准确展示网络带宽、延迟、负载和 TCP 端口的变化是如何影响用户的响应时间的。利用网络应用性能分析工具，例如 Application EXPert，能够发现应用的瓶颈，我们可知应用在网络上运行时在每个阶段发生的应用行为，在应用线程级分析应用的问题。可以解决多种问题：客户端是否对数据库服务器运行了不必要的请求？当服务器从客户端接受了一个查询，应用服务器是否花费了不可接受的时间联系数据库服务器？在投产前预测应用的响应时间；利用 Application Expert 调整应用在广域网上的性能；Application Expert 能够让你快速、容易地仿真应用性能，根据最终用户在不同网络配置环境下的响应时间，用户可以根据自己的条件决定应用投产的网络环境。

2.3.2.2 网络应用性能监控

在系统试运行之后，需要及时准确地了解网络上正在发生什么事情；什么应用在运行，如何运行；多少 PC 正在访问 LAN 或 WAN；哪些应用程序导致系统瓶颈或资源竞争，这时网络应用性能监控以及网络资源管理对系统的正常稳定运行是非常关键的。利用网络应用性能监控工具，可以达到事半功倍的效果，在这方面我们可以提供的工具是 Network Vantage。通俗地讲，它主要用来分析关键应用程序的性能，定位问题的根源是在客户端、服务器、应用程序还是网络。在大多数情况下用户较关心的问题还有哪些应用程序占用大量带宽，哪些用户产生了最大的网络流量，这个工具同样能满足要求。

2.3.2.3 网络预测

考虑到系统未来发展的扩展性，预测网络流量的变化、网络结构的变化对用户系统的影响非常重要。根据规划数据进行预测并及时提供网络性能预测数据。我们利用网络预测分析

容量规划工具 PREDICTOR 可以作到：设置服务水平、完成日网络容量规划、离线测试网络、网络失效和容量极限分析、完成日常故障诊断、预测网络设备迁移和网络设备升级对整个网络的影响。

从网络管理软件获取网络拓扑结构、从现有的流量监控软件获取流量信息（若没有这类软件可人工生成流量数据），这样可以得到现有网络的基本结构。在基本结构的基础上，可根据网络结构的变化、网络流量的变化生成报告和图表，说明这些变化是如何影响网络性能的。 PREDICTOR 提供如下信息：根据预测的结果帮助用户及时升级网络，避免因关键设备超过利用阀值导致系统性能下降；哪个网络设备需要升级，这样可减少网络延迟、避免网络瓶颈；根据预测的结果避免不必要的网络升级。

2.3.3 应用在服务器上性能的测试

对于应用在服务器上性能的测试，可以采用工具监控，也可以使用系统本身的监控命令，例如 Tuxedo 中可以使用 Top 命令监控资源使用情况。实施测试的目的是实现服务器设备、服务器操作系统、数据库系统、应用在服务器上性能的全面监控。

第 4 节 其他分类二

来源于网络

1. 性能测试:测试正常使用时,系统是否满足要求,同事可能为了保留系统的扩展空间进行一些稍微超出"正常"范围的测试.
2. 负载测试: 负载测试是一种性能测试指数据在超负荷环境中运行，程序是否能够承担。通过再被测系统上不断增加压力,知道性能指标(如响应时间、资源状态).此测试考验找到系统的处理极限,为系统调优提供数据,一般负载测试的压力比较大.
3. 压力测试:对系统不断的施加压力,确定一个系统的瓶颈或不能接受的性能点来获得系统能够提供最大的服务级别.

负载测试和压力测试两者可以结合进行。通过负载测试，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。压力测试是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

4. 配置测试:通过测试找到系统各项资源的最佳分配原则,是系统调优的重要依据.
5. 并发测试:测试多个用户同时访问一个应用程序、同一模块或者数据记录时是否存在思索或者其他性能问题.几乎所有的性能测试都设计到并发测试.
6. 容量测试:测试系统能够处理最大会话能力,确定系统可处理同时在线的最大用户数,通常和数据库有关.
7. 可靠性测试:通过给系统加载一定的业务压力情况下,运行一段时间,检查系统是否稳定.
8. 失败测试:对于有冗余备份和负载均衡的系统,通过此测试来检验如果系统局部发生故障,用户是否能够继续使用系统,用户受到多大的影响.

第 5 节 其他分类三（RUP）

来源于网络

• 基准测试 ：比较新的或未知测试对象与已知参照标准（如现有软件或评测标准）的性能。
• 争用测试 ：核实测试对象对于多个主角对相同资源（数据记录、内存等）的请求的处理是否可以接受。
• 配置测试 ：核实在操作条件保持不变的情况下，测试对象在使用不同配置时其性能行为的可接受性。
• 负载测试 ：核实在保持配置不变的情况下，测试对象在不同操作条件（如不同用户数、事务数等）下性能行为的可接受性。
• 强度测试 ：核实测试对象性能行为在异常或极端条件（如资源减少或用户数过多）之下的可接受性。

第 3 章 测试步骤

系统性能测试中的几大步骤：

0. 明确测试目标；了解性能测试需求；
1. 编写性能测试计划；
2. 分析性能测试需求；
3. 编写性能测试方案、设计测试场景；
4. 相关资源准备(人力资源、硬件资源、软件资源)；
5. 测试程序开发；脚本维护、测试数据准备、测试监控准备；
6. 执行性能测试并收集测试结果；
7. 分析结果；
8. 系统调优及再测试；

3.1.1 明确测试目标

性能测试启动阶段要确定测试的负责人和组织结构。明确测试的总体目标和范围，确认资源情况。获取性能测试需求：业务列表、性能指标、测试环境、数据量等详细需求。为策划规划做准备。

性能调优是是无止境的，所以在测试之前应确定一个明确性能调优目标，这也是后面“评估性能验证”的一个基准，也是测试终止的一个基准。

在这个阶段要明确以下信息：

确定本次测试对象的相关背景；

总体目标和范围：1.客户需求和期望；2.实际业务需求；3.系统需求；

系统构架；

以前测试的历史记录；

系统目前的情况；

与之相关的非性能需求；

以上提到的这些内容在初期只是从整体上做一个了解，具体的需求内容，将在测试需求分析章节讲到。

3.1.2 测试计划

性能测试计划中包含测试目的和测试目标的相关信息，还确定了实施和执行测试时使用的策略，方法；同时确定了测试工具、所需资源、日程表等。

3.1.2.1 性能测试计划关键点

阶段任务描述（阶段，子阶段划分清晰；阶段关联关系明确；里程碑定义准确）

时间安排（满足项目预期周期要求，具有弹性）

文档定义（各阶段输入输出文档定义清晰）

所需资源（人力资源，资金资源等符合项目要求）

3.1.3 测试需求分析

3.1.3.1 需求获取

需求的获取非常复杂，而且要求测试人员具有挖掘可能造成系统瓶颈因素的洞察力和敏锐感，这个过程在测试整体过程中是非常关键的一环。要获取的需求在后面几节中将会讲到 。

3.1.3.2 需求分类

性能测试需求分析主要目的是要找出可能造成系统瓶颈的因素，为后面的测试场景设计提供依据。影响系统性能有很多种原因，在此应关注如下几个关键点：

1、环境配置性能需求：

应用配置需求：例如应用整体框架、涉及到哪些第三方的组件、应用层与数据库层的接口、使用了什么数据库等；

系统配置需求：例如用户客户端配置、客户端与服务器端的网络配置、应用服务器或数据库服务器操作系统等等；

2、服务器性能指标要求：

预期的在上线系统中服务器资源使用情况、吞吐量、软件运行情况等等。

3、系统设计需求：

系统架构、系统的技术实现、与其它系统接口关系及其技术实现、本系统测试数据及其与相关系统测试数据关系等等。

4、工作负载需求：

用户使用情况需求：例如用户分布情况；哪些模块用户使用比较频繁；在用户操作的数据有哪些特点等等；这些需求需要具体定位到系统的哪些功能模块、功能点；

5、客户端性能指标要求：

请求响应时间分布；请求的准确率等等。

根据性能需求的提出，需求又可以分为：

用户提出的性能需求：例如业务量大、使用频繁的功能；

开发提出的性能需求：例如系统处理相对较复杂的位置；

系统本身的理想化性能展现：例如理想化的系统性能状况；

3.1.3.3 测试环境需求

环境需求主要包括：服务器环境配置要求、客户端机器配置要求、网络环境等等硬件因素，还包括一些软件因素：如服务器使用的数据库、中间件类型和版本，客户端的类型版本 、网络传输协议版本等等，这些因素都会对系统系统产生不同程度的影响。

3.1.3.4 工作负载需求

根据测试项目类型的不同，要求的测试指标也不一样。最能反映系统性能情况的是：点击数、处理量、响应时间、硬件的使用情况。测试指标包括如：并发用户数、事务、事务响应时间、每秒点击数、每秒处理业务数、连接数、每秒增加连接数、系统吞吐量等等。

需要客户提供或自己设法采集性能测试需要的一些信息，可以参考后面两节。

功能流程性能测试需求

1.吞吐量要求；
2.响应时间要求；
3.系统资源使用情况要求（内存/cpu 等）；
4.用户并发量数据：

性能参数 功能点	用户数量	活跃用户数	每个用户使用频率	使用天数	使用小时数	平均值笔/小时	峰值	
							日峰值（笔/天）	每小时峰值（笔/小时）
模块一								
功能一								
功能二								
...								
模块二								
功能一								
功能二								
...								

数据计算的参考公式：
平均天处理并发数=用户数×使用次数÷（使用天数×使用小时数）
平均天处理并发数=用户数×使用次数÷（8×使用小时数）[8 是参考值]
日高峰=用户数量 × 使用次数 × 月集中访问访问比例 ÷ 集中使用天数
小时高峰数=日高峰数 × 天集中比例÷集中使用小时数

- **工作负载估算时采用如下原则：**
  全年的业务量集中在 8 个月完成，每个月 20 个工作日，每个工作日 8 个小时；
  每个工作日中 80%的业务在 20%的时间内完成，即每天 80%的业务在 1.6 小时内完成；
  有 80%的请求集中在 20%的业务功能。

网页访问性能测试指标

1.吞吐量要求；
2.响应时间要求；
3.系统资源使用情况要求（内存/cpu 等）；
4.用户并发量数据：
业务逻辑不强的 Web 的访问量,一般以点击数来衡量. 选择有代表意义的页面,衡量其他平均

响应时间的大小,主要是首页、二级主页以及各类模版页面。

页面名称	页面连接数	页面大小	平均用户数	峰值用户数	访问时间需求	响应时间需求(秒)
页面 1						
页面 2						
...						

总用户数	访问页数(页)	每天访问时间	峰值天数	每月天数	页平均连接数	每个连接下载量 K

平均日访问量(次)=用户×访问页数×页平均连接数÷每月天数	
平均每小时访问量(次)= 平均日访问量÷每天访问时间	
平均每秒访问量(次/秒)= 平均每小时访问量/3600	
峰值日访问量(次/日)=用户×访问页数×页平均连接数÷峰值天数	
峰值小时访问量(次/小时)= 峰值日访问量÷每天访问时间	
峰值秒访问量 (次/秒)= 值小时访问量/3600	
平均日吞吐量(MBytes)=平均日访问量×每个连接下载量÷1024	
平均每秒吞吐量(k Byets/s)= 每秒平均访问量 × 每个连接下载量	
高峰时每秒吞吐量(k Byets/s)= 每秒访问量峰值 ×每个连接下载量	

3.1.4 测试方案设计

测试内容一般包括并发性能测试、疲劳强度测试、大数据量测试以及系统资源监控等，性能调优测试时主要是做并发性能测试以及系统资源监控这些方面的工作。从对系统产生并发性能测试过程中监控系统中各种资源的变化，来判断导致性能瓶颈的原因。

3.1.4.1 注意事项

测试案例主要是根据测试需求分析的结果制定出在测试执行时系统的执行方法，在测试案例设计时应注意如下几点：

明确测试目的、测试范围；明确项目功能需求以及负载分布情况。

分析测试环境中可能出现瓶颈的位置，重点测试。

综合业务场景中：各个操作的并发量、所占百分比、准备完成请求数量、频率等等典型行为是明确的；虚拟用户的操作步骤要尽量类似于真实用户的操作； 操作的数据要类同于真实用户实际使用数据；在案例设计时要充分考虑到需求中用户对模块的使用频率。使得在模拟时每个模块使用情况尽量地类似于真实环境。

明确的通过指标：为每个典型业务制定测试通过的指标。

有效的测试工具：需要产生负载的应用可以在合理的时间内达到你期望的负载，然后再缓慢增加。

详细的结果记录：对于每个请求的响应时间、开始时间、结束时间、响应时间细分、传

输数据量、请求内容等等做详细的记录，以方便对性能测试进行分析。

资源利用率的监测：有效的资源监控方式，使能获得所有需要的资源数据。

软硬件环境一致性。

数据库或文件系统中测试数据与正式环境数据的数量和内容的一致性。

3.1.4.2 测试场景

性能测试在软件测试中占有重要的地位，性能测试包含的内容是很多的。例如针对一个网站进行性能测试，常规性能测试、压力、负载测试、大数据量测试、强度测试等等都应该包含在内的。

测试方案中可能把测试按照类型划分，每个类型下又设计 n 个场景。测试中需要注意的内容在前面章节都已经讲到，这里不再重复阐述。

通常测试中会使用的一些场景：

预期性能指标测试

通常系统在设计前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。针对每个指标都要编写多个测试用例来验证是否达到要求。

单一功能加压测试

这类场景主要为了验证某一特定功能的性能状况，或者检测该功能是否存在隐藏的性能问题。

复杂场景测试

根据业务数据分析，设计与真实情况类似的场景，测试系统整体性能。一般，预期性能指标验证、整个系统性能评定、整个系统的性能调优测试都采用这种场景。

其他如压力测试、负载测试、大数据量测试、配置测试等测试类型在这里不再重复介绍。

3.1.5 相关资源准备

主要包括以下测试资源：

测试环境准备及配置：

包括被测应用的主机和应用环境的申请、部署，压力发生环境准备，监控系统准备和网络环境申请和部署，提供符合测试需求可使用的测试环境；

包括被测应用系统、压力发生系统、监控系统、网络系统的配置等等；

测试场景准备：

根据业务模型确定测试场景；

测试脚本开发、数据准备、参数化数据、脚本预验证：

测试脚本的开发；

基础数据的获得、数据量评估和基础数据设计；

设置测试程序中需参数化的字段，使所有参数化数据可以正确执行场景。保证参数化的测试脚本与基础数据结合能够在测试执行环境下正确运行；

验证最终可是脚本可以在测试环境正确执行，并且自身无性能问题。

监控系统与数据分析系统准备：

设计合理的监控方案，并设计相应的监控工具实施监控；

对监控数据和结果数据进行分析的脚本或者程序。

3.1.5.1 测试环境准备

测试环境：配置测试环境是测试实施的一个重要阶段，测试环境的适合与否会严重影响测试结果的真实性和正确性。测试环境包括硬件环境和软件环境，硬件环境指测试必需的服务器、客户端、网络连接设备以及打印机/扫描仪等辅助硬件设备所构成的环境；软件环境指被测软件运行时的操作系统、数据库及其他应用软件构成的环境。

一个充分准备好的测试环境有四个优点：一个稳定、可重复的测试环境，能够保证测试结果的正确；保证达到测试执行的技术需求；保证得到正确的、可重复的以及易理解的测试结果；与真实环境一致或尽量保持一致。

性能测试环境，要求和真实环境一致或存在可比性。

3.1.5.2 测试程序开发

性能测试场景设计和性能测试脚本设计中需要注意以下几个问题：

场景真实性：

每个脚本的具体操作步骤，是和真实环境操作类似的，每个场景内的测试程序之间的逻辑关系、压力比重也是与真实环境类似的。在测试方案设计中已经将这些真实环境信息量化到可用程序模拟程度。

场景中各个脚本所占的比例主要是通过线程组来控制的，即可以控制多少个线程在某一阶段运行哪一个或几个脚本。

具体操作步骤需要注意两个相邻的操作动作之间的习惯性时间间隔、某些特定操作的集合在特定时间同时发起请求等等。

数据驱动：

确定某个操作步骤中需要将输入数据进行参数化，参数化数据尽量选择读文件的方式、在数据库中读取会严重影响速度。如果数据量不大，可以在并发程序开始执行前先将数据读如内存。

值得注意的是，不是所有的数据都可以通过读取预先定制的文件来获得。而是在程序执行过程中，在返回结果中获取这个数据信息，然后运用到下一个操作中。测试界把它称作“关联”常用测试工具都提供的关联功能，自己编写测试程序时也可以把它作为单独的程序模块来编写，方便在后续的项目中复用。

事务：

一个复杂的操作应该是有很多个事务组合而成的。使用事务可以更方便灵活的进行数据统计。

检查点：

检查点的作用是验证操作结果是否正确。它可以写在程序中，可以通过操作返回结果来判断。具体实现的方式不固定的。只要能够方便的检查，我们运行的并发请求是否都是有效的。但是检查点不宜设置过多，因为在做检查时会有系统消耗，影响并发性。

保存登录信息

很多系统性能测试场景是需要在用户登录状态下操作的。那么保存登录用户信息也是一个很重要，需要我们注意。

3.1.5.3 测试数据准备

性能测试中用到的数据有两类：

1. 测试环境中系统应该具备的初始数据以及和正式环境同等数据量（或加权值）的有效数据，或者是在系统生命周期内预期能达到的最大数据量的数据，尽量保证其真实性。

2. 运行测试脚本需用到的数据，参数化数据。

在测试需求调研过程中也要明确数据量要求，数据准备一定要关注数据的质量和数量，不要出现一些不符合业务逻辑的废数据，并且数据量要满足测试运行的需要。否则会导致测试执行结果出现大的偏差。数据的部署也应该尽量保证真实性。

如果是配置测试或者测试数据对测试结果影响很大的场景，要保证测试过程基准一致。否则测试结果将没有可比性。所以，测试数据创建完成后，要及时备份，以便多个场景中重复使用这些数据。可以创建一个测试数据的版本库来管理测试数据。

3.1.5.4 测试监控准备

根据测试的目的不同，测试监控的对象不同，测试的主要指标也不相同。

测试指标一般分为两大类：业务处理指标、系统资源指标。其中，业务处理性能指标主要包括：业务结果的正确性、每分钟处理请求数、事务响应时间（Min、Max、Avg、90％相应时间范围、响应时间变化、分布等等）、并发用户数、系统吞吐量等等；系统资源指标主要是指 CPU、内存、硬盘、网络等系统资源使用及变化情况。

具体需要选择的测试指标，参见测试指标章节。

3.1.6 测试执行

在性能测试启动之前，需要就被测系统是否符合准入标准，和实施性能测试的可行性和必要性进行分析。考察被测系统是否具备性能测试的条件。

测试执行过程中，要特别注意测试脚本和场景的运行是有效的。在测试程序开发过程中，我们加入了检查点就是为了保证程序正确的执行、产生我们所需要的负载。但是，在测试程序长期执行过程中，我们还需要经常检查测试正确是否是正确执行。否则，得出的测试结果可能是错误的、无意义的。

测试执行与监控的主要目的是根据设计方案去验证系统是否存在瓶颈，给测试分析提供各种分析数据。此过程会与测试分析过程不断进行重复执行，直至真正确定出系统瓶颈所在 。

3.1.6.1 测试执行关键点

执行测试的过程中，有一些需要重点关注的事件：

在测试执行前，需要确认：用例和场景确认、测试环境确认、测试数据确认、测试脚本确认、测试工具和监控工具确认。

执行过程中注意：执行每个场景时都需要做执行记录、结果搜集等工作。

执行完成后注意：数据恢复、环境清理、结果整理。

3.1.6.2 测试记录

测试过程中要记录如下信息：运行前的系统配置、运行前的参数或者软件调整情况；运行过程中的系统资源的使用情况；运行过程中的故障现象；请求的响应时间 ；单位时间内的处理请求数 ；请求的成功率；等等。

3.1.7 测试分析

测试分析的主要目的是要根据测试执行获取到的数据去判断造成系统出现瓶颈的位置，挖掘造成系统瓶颈的真正原因。这个过程是技术含量最高的一环。

性能分析通常要围绕三个方面进行：软件、服务器、网络。

软件主要是分析具体事务执行时间，尤其并发用户部分，根据测试工具测试出的结果分析那些事务执行的慢，然后可以分析执行较慢的代码，针对网页可以分析具体的页面元素执行情况。

服务器的分析要结合软件的运行情况进行分析，着重分析硬件的执行参数，CPU、硬盘、内存、中断、内存等情况，分析尤其要注意对这些参数进行综合分析，往往各个参数之间会互相影响，最后在调查、分析整体系统的基础上，找出影响服务器整体性能的瓶颈，确定相应的升级需求。

网络性能分析要结合结合服务器和测试目标软件，通常网络传输慢会有软件和服务器方面的原因，甚至有时候会有客户端方面的原因。不过目前网络的环境普遍可以，不管是局域网还是广域网，网络的环境越来越好。

3.1.7.1 测试分析建议

- 测试分析小组应该由具有以下素质的人员组成：开发、测试、应用服务器、数据库、操作系统、网络、存储等等。
- 在测试分析时使用较多的一种方式是排除法，根据开始获取到的信息大概能将问题定位在某一层面上。但具体在什么地方，就可以采取排除的方法去定位。
- 尽量使用一些比较成熟的工具协助分析工作，这样将大大减轻工作负担。
- 在确定出真正的性能瓶颈时，可以做一些小的测试模型去做验证，确定分析的正确性。

3.1.7.2 瓶颈分析

处理器分析方法

首先查看 System\%Total Processor Time 计数器的值。该值体现的是 CPU 的平均利用率，若超过 90%，则说明存在处理器方面的瓶颈。

其次查看每个 CPU 的 Processor\%User Time 计数器的值。若应用服务器的%User Time 值较大，可以考虑是否能通过算法优化等方法降低这个值。若数据库服务器的%User Time 值较大，可考虑对数据库系统进行优化。

查看 System\Processor Queue Length 计数器的值。当该值大于 CPU 数量的总数+1 时，说明存在处理器方面的问题。

磁盘 I/O 分析方法

查看%Disk Time 计数器的值。该值较大，则可能存在磁盘瓶颈问题。

与 Processor\Privileged Time 合并进行分析。若%Disk Time 值较大，而 Processor\Privileged Time 的值适中，则可判断存在磁盘问题。若 Processor\Privileged Time 较大，持续超过 80%，则可能是内存泄漏。

根据 Disk sec/Transfer 进行分析。该值超过 60ms，则磁盘存在问题。

网络分析方法

查看 Network Interface\ Bytes Total/sec 计数器的值。用 Bytes Total/sec 计数器的值和网络的带宽进行比较，若超过 50%，则说明网络存在性能瓶颈问题。

软件瓶颈分析方法

分析事务响应时间、吞吐量，确定是否存在性能问题，若发现存在性能问题，则找出响应时间不符合要求或者出现多个失败的事务，对其进行分解，然后对其进行网页细分，以确定影响性能的元素。

3.1.7.3 内存泄漏

对于 C++或 java 系统，如果 GC 处理不合理，会引起很多内存堆栈问题。常见有：

1、 Array Bounds Read (ABR) ：数组越界读；
2、 Array Bounds Write (ABW)：数组越界写 ；
3、 Beyond stack Read (BSR)：堆栈越界读 ；
4、 Free Memory Read(FMR)：空闲内存读 ；
5、 Invalid pointer Read(IPR)：非法指针阅读 ；
6、 Null Pointer Read(NPR)： 空指针阅读 ；
7、 Uninitialized Memory Read(UMR)：未初始化内存读写 ；
8、 Memory Leak：内存泄漏。

3.1.7.4 Web 响应时间细分

响应时间：

Time = DNS + Conn + Handshaking + FTPAuthentication + Send + ServerTime + Rec +Client；

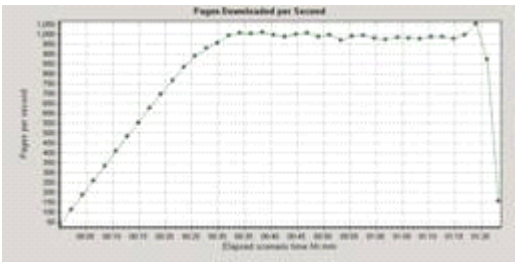
ServerTime =logic+DB*n+.....；

3.1.7.5 吞吐量等指标变化情况

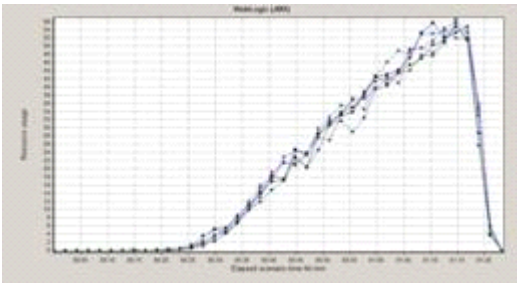
不断的增加用户数，提高压力。各性能指标变化情况：

响应时间和吞吐量会受到负载的影响。随着服务器上负载的增加，吞吐量会不断攀升，直到到达一个点。

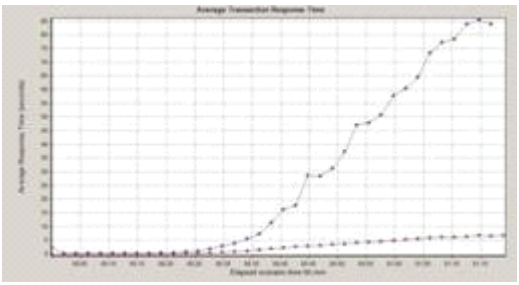
最初的一段时间，执行队列的长度为零，然后就开始以稳定的速度增长。



在某一点上，执行队列开始增长，因为服务器上所有的线程都已投入使用，传入的请求不再被立即处理，而是放入队列中，当线程空闲时再处理。



当系统达到饱和点，服务器吞吐量保持稳定后，就达到了给定条件下的系统上限。但是，随着服务器负载的继续增长，系统的响应时间也随之延长，虽然吞吐量保持稳定。



进入饱和状态后继续增加负载，服务器吞吐量保持稳定、系统的响应时间延长，最终会出现系统不可用、宕机等情况。

3.1.7.6 配置优化

缓存机制，可是设置多级缓存；
数据压缩机制，减小网络传输；（减少每个请求的大小）
大数据块分块传输；
负载平衡，用来水平缩放服务器的结构；
数据库集群，达到水平缩放群；
分为读写服务器和只读服务器，还要对服务器群负载平衡；
增加更多的硬件资源：CPU，内存，磁盘等；
增加网络的带宽；等等。

3.1.7.7 瓶颈分析工具

应用层：开发人员可以通过 profilers 来发现低效率的代码，比如说较差的查找算法。
数据库层：开发人员和数据库管理员（DBA）可以通过特定的数据库 profilers 及事件探

查器（query optimizers）。

操作系统层：系统工程师可以使用一些工具如在 Unix 类的操作系统中的 top,vmstat,iostat,在 Windows 系统中的 PerfMon 来监控 CPU，内在，swap,磁盘 I/O 等硬件资源；专门的内核监控软件也可以在这一层面上被使用。

网络层：网络工程师可以使用报文探测器（如 tcpdump）,网络协议分析器（如 ethereal），还有其它的工具（如 netstat,MRTG,ntop,mii-tool）。

第 4 章 性能测试实例分析

性能测试：测试软件的运行性能。这种测试常常与强度测试结合进行，需要事先对被测软件提出性能指标，如传输连接的最长时限、传输的错误率、计算的精度、记录的精度、响应的时限和恢复时限等。

第 1 节 综合分析

4.1.1 系统结构分析

服务环境一般都是集群方式，测试需要考虑可能出现瓶颈的位置。系统由客户端、网络、防火墙、负载均衡器、 Web 服务器、应用服务器 (中间件) 、数据库等等环节组成。

最能反映系统性能情况的是：点击数、处理量、响应时间、硬件的使用情况。

处理量一般受以下几个方面的影响：网络带宽，服务器的配置，应用，服务的配置等。从上图来看，影响处理量的因素有以下：对外的网络带宽，服务配置(APCHE，WEBLOGIC)，服务器的性能及配置（WEB，应用）。

响应时间一般受以下几个因素影响：网络带宽，服务器的配置，应用，服务的配置等。响应时间会因公众使用的带宽而异，难以用公众端的响应时间来衡量，因此，响应时间可以定义为在没有网络延迟的情况下的响应时间。影响响应时间的主要有几个方面：应用，防火墙等。

服务器资源的使用情况，也能反映出系统的性能，如：CPU 使用率，内存使用率等。使用率的增加最为理想的情况下应该是线性增长的，如果 CPU 使用率，内存使用率增长比用户量的增长大得多的情况下，说明应用需要进行调优。

一些隐含的问题：如内存泄漏，日志空间不够等。针对这些问题，需要进行一个长时间的测试。

4.1.2 测试指标

- **系统资源指标：**
  - ProcessorTime： 指服务器 CPU 占用率，一般 平均达到 70%时，服务就接近饱和；
  - Memory Available Mbyte ： 可用内存数，如果测试时发现内存有变化情况也要注意，如果是内存泄漏则比较严重；
  - Physicsdisk Time ： 物理磁盘读写时间情况；
- **业务指标：**

指标	解释	说明	
并发用户数	是指同时访问系统的用户数，反映了系统所能承受的压力。	稳定	系统在长时间正常响应的情况下，所能支持的用户数。
		最大	在网页出错率<10%（HTTP 500、connect 和超时错误），所能支持的虚拟并发用户量。
事务	是指一个完整的请求动作。如登录包括：验证用户密码、请求主界面及其上面		

	的图片等。 与事务相关的指标有：事务量，成功事务量，失败事务量，平均事务量。
响应时间	响应时间（Response Time）从用户的角度来看，系统处于良好的性能状态是指系统能够快速响应用户的请求，即系统响应时间短。 具体地说，响应时间是指发出请求的时刻到用户的请求的相应结果返回用户的时间间隔。 定义事务响应时间的标准如：< 5s：优；5～15s：可接受；>30：不可接受；
点击率	系统 WebServer 每秒所能响应的请求数
交易率	系统每秒响应交易的笔数
连接数	当前测试打开的连接数。一般在测试用户数据稳定时，会达到一个均衡值，如果出现突然起伏，说明系统性能存在问题。
每秒增加连接数	此参数与连接数密切相关，只有关闭数与新增数达到一致时，连接数才能保持一个均衡水平。如果新增的连接数较大，系统资源消耗会很大，同时，系统连接数存在冗余，重复利用率太低。
吞吐量	吞吐量(Throughput)：系统传输的数据总量。 相关指标：吞吐率：系统单位时间传输的数据量。
其他	服务器的 Cpu，内存，核心进程数，Weblogic 的日志文件增长情况。

◆ **数据库指标**

略

4.1.3 测试类型

- 负载压力测试
- 疲劳度测试

等等。

4.1.4 测试需求

根据以前系统的运行纪录(并发量，访问量，点击率等)，或者根据同类产品的性能指标 。也可以是相关人员预计的系统性能指标。

◆ **需求采集和整理**

测试需求采集和整理是性能测试最关键的步骤之一，需求不合理，后面的工作都是徒劳的。

测试需求采集，采集合理的信息才能使用等价类的方式去判断哪些功能是必须要测试的，还决定了测试中这些功能脚本对应的各项指标的设置。比如查询功能，我们是不是要考虑以下的参数：数据库中相关表的数据量；数据库软硬件参数；用户使用频率和使用高峰期各项参数值；用户常用查询条件和查询结果集的大小；性能测试（正常的负载、容量变化）；压力测试（临界的负载、容量变化）。

另外需要注意的是，客户提出了需求后，首先要验证在现有条件下，需求的可模拟性。

需求采集完成后，下面的工作就是对需求进行整理和细化，使之成为可模拟的数据。根据特定的业务特定的计算公式对客户的需求进行计算。选择有代表性的功能或页面编写脚本，计算出每次测试各个脚本对应的功能所需要达到的并发数。

4.1.5 测试环境

一般需要在真实环境做测试，或者与真实环境资源配制相同的环境，需要纪录所有相关服务器和测试机的详细信息。方便对测试数据的比较分析。

- 服务器：web 服务器、应用服务器、数据库服务器、磁盘阵列服务器等等及其相关的软硬件配置信息；
- 测试机：测试 server、测试客户端、监控机、数据记录机器、备份数据机器等等及其相关的软硬件配置信息；

4.1.6 测试方案

一般网站测试是按照点击率为依据来执行，而工作流测试是按照事务率来执行的。
评估并发量的方式：

- 按照点击率测试：逐渐加大并发量，找出系统点击率最高时的并发量；
- 按照并发量来测试： 逐渐加大并发量，直至达到使系统的处理量及响应时间在达到用户不可接受的程度，并且记录下这个并发量，作为系统的临界点；

4.1.7 测试场景

测试场景的设计是很重要。测试场景一般包括：各个功能操作的执行频率、并发量(或百分比)、启动方式、结束方式、运行时间，监控指标，是否取缓存，脚本之间的逻辑，脚本内部的逻辑，成功和失败标准等等。

4.1.8 执行测试

- 执行测试前，需要要检查：各项系统资源是否足够；是否能保证测试环境不间断运行；
- 执行测试过程，需要实时监控系统运行状况，纪录系统出现的异常，纪录测试过程中的各项数据。
- 执行完成后，对执行过程中记录的数据作分析。

4.1.9 测试结果分析

参见第二章，这里不做介绍。

第 2 节 网站

4.2.1 需求分析

首先要明确的是本次测试的目的：为客户提供一个系统整体性能验收测试报告，证明我

们的系统是完全符合各项性能指标的。

对于一般客户，对于性能测试的各项指标并不了解。他们给出的性能标准可能是这样的：日流量：5000 万/天、数据库存储内容数：1 亿、响应时间：1s、持续运行：7*24 小时。

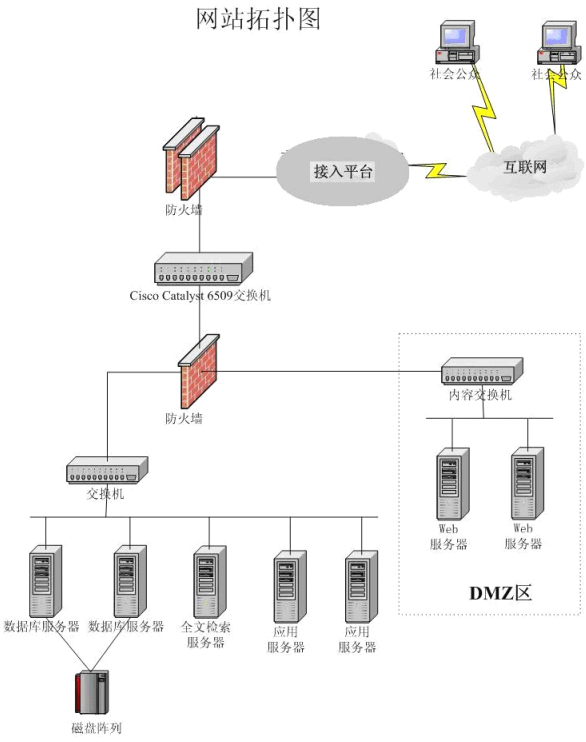
以上信息很难帮助我们完成性能测试，测试人员首先要做的是充当需求人员，介绍各项测试指标给客户，明确每个指标的阈值，以及如何得到这些数据，如何设计场景，让客户觉得你的设计，要得出的结果是很有道理的，这样整个测试已经成功了一半。当然，在介绍各种性能指标、阈值、场景等之前，要对系统的性能、测试技术实现做一个评估。

◆ **要掌握哪些需求？**

吞吐量要求、响应时间要求、系统资源使用情况需求、用户并发量及场景设计数据。详细信息见后面章节。

4.2.1.1 架构分析

分析系统架构，了解系统网络情况，分析系统可能瓶颈。通常系统由客户端、网络、防火墙、负载均衡器、 Web 服务器、应用服务器（中间件）、数据库等等环节组成。如下某网站拓扑图。



4.2.1.2 并发量与测试场景设计分析

从网站的业务类型分析，流量的主要压力来源：用户的读操作，随机读。网站页面可以根据前台页面类型、后台实现逻辑、存储位置等，划分等价类。比如：首页、二级首页、内容页（需考虑各种模板）、各种业务页、登录页等等，甚至可以细分到页面的一个 iframe 区

域。根据每种类型页面的历史访问量或预估访问情况，可以得出一个基本场景设计和并发量的需求表：（仅供参考）

业务逻辑不强的 Web 的访问量,一般以点击数来衡量. 选择有代表意义的页面,衡量其他平均响应时间的大小,主要是首页、二级主页以及各类模版页面。

页面类型	页面连接数	页面大小	平均用户数	峰值用户数	访问时间需求	响应时间需求(秒)
首页						
二级主页 1						
二级主页 2						
内容页 1						
业务流页 1						
...						

总用户数	访问页数(页)	每天访问时间	峰值天数	每月天数	页平均连接数	每个连接下载量 K

平均日访问量(次)=用户×访问页数×页平均连接数÷每月天数	
平均每小时访问量(次)= 平均日访问量÷每天访问时间	
平均每秒访问量(次/秒)= 平均每小时访问量/3600	
峰值日访问量(次/日)=用户×访问页数×页平均连接数÷峰值天数	
峰值小时访问量(次/小时)= 峰值日访问量÷每天访问时间	
峰值秒访问量 (次/秒)= 值小时访问量/3600	
平均日吞吐量(MBytes)=平均日访问量×每个连接下载量÷1024	
平均每秒吞吐量(k Byets/s)= 每秒平均访问量 × 每个连接下载量	
高峰时每秒吞吐量(k Byets/s)= 每秒访问量峰值 ×每个连接下载量	

若站点内包含一些涉及业务流程的频道，需要收集以下信息：

性能参数 功能点	用户数量	活跃用户数	每个用户使用频率	使用天数	使用小时数	平均值笔/小时	峰值	
							日峰值（笔/天）	每小时峰值（笔/小时）
模块一								
功能一								
功能二								
...								
模块二								
功能一								
功能二								
...								

数据计算的参考公式：

平均天处理并发数=用户数×使用次数÷（使用天数×使用小时数）；

平均天处理并发数=用户数×使用次数÷（10×使用小时数）；

日高峰=用户数量 × 使用次数 × 月集中访问访问比例 ÷ 集中使用天数；

小时高峰数=日高峰数 × 天集中比例÷集中使用小时数；

4.2.1.3 吞吐量要求

系统吞吐量有两种说法：1、单位时间完成的请求数；2、单位时间传输的字节数，都是对系统处理能力的评估标准。一般会关注 吞吐量随着访问用户量的增加而发生变化的情况，或者吞吐量随着固定用户量持续访问而变化的情况。

对于吞吐量，客户会希望通过测试得到它的变化情况。而不能给出预期的结果。我们最终提供给客户的结果类似下表：（也可以是曲线图等形式）

压力测试：

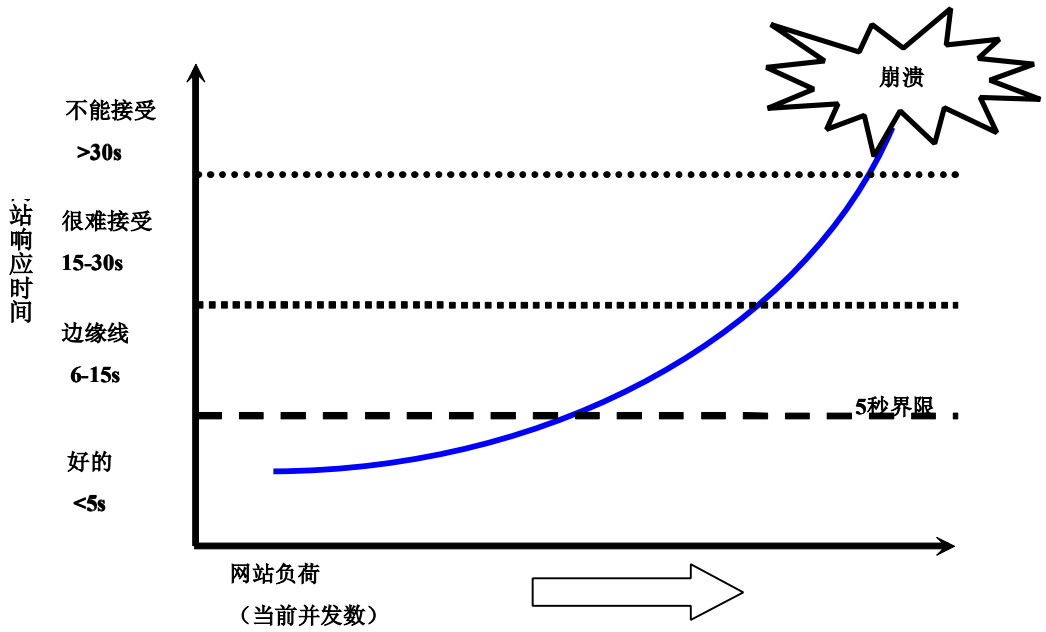
并发数	吞吐量	响应时间	系统资源
100VU			
200VU			
…			

稳定性测试：

时间	吞吐量	响应时间	系统资源
1hour			
2hour			
…			

4.2.1.4 响应时间要求

响应时间是很重要的一项指标，随着并发数的增加，网站的响应时间也会逐渐增加，我



们可以看到，响应时间是不断增加，但是能被用户所接受的时间始终在一个区间内。

以上标准只是一个参考值，响应时间接受度的边界，根据产品的不同也不一样，同一产品中，各个不同的页面，不同的业务流页面的响应时间要求也会不一样。需要根据实际情况做相应 的调整。

对于响应时间的关注，除了平均响应时间外还包括一下内容：响应时间最大值、最小值、平均值、响应时间分布情况、90%请求响应时间分布范围、响应时间随用户数量或时间变化的情况等等。

影响响应时间的因素，除了程序设计之外，还有网络传输、服务器架构策略、客户端因素等等。

4.2.1.5 系统资源使用要求

服务器资源的使用情况，也能反映出系统的性能，如：CPU 使用率，内存使用率等。使用率的增加最为理想的情况下应该是线性增长的，如果 CPU 使用率，内存使用率增长比用户量的增长大得多的情况下，说明应用需要进行调优。

一些隐含的问题：如内存泄漏，日志空间、 数据库死锁不够等。针对这些问题，需要进行一个长时间的测试。

4.2.2 场景设计

本次测试从系统整体性能入手，主要在压力、稳定性、大数据量三方面进行测试。

本次负载压力测试计划分为 5 个测试步骤来完成：

第一步，检验我们的系统的处理是否达到一个预计的水平，xUesrs 同时访问系统，每秒完成请求数 x，运行 x 小时；

第二步，通过模拟并发情况，从 xUsers 到 xUsers 个并发逐步增加，看本系统随着压力增大，响应时间的增长趋势是否正常，可以找到允许响应时间等指标内的最优处理能力（并发量、事务率），同时可以找到一个最大的压力值（并发量、事务率）；

第三步，考查最大的并发情况下的系统运行状况，使用系统允许最大并发量运行一段时间，检查系统运行状况；

第四步，稳定性测试，使用最优处理能力时的并发量事务率（或加权值），长时间运行（7*24）小时。

第五步，大数据量测试，场景略。

4.2.3 行业指标

2006 年底，Aexa 上统计的各个门户网站的响应时间：

编号	网站	Avg Load Speed	类型
1	qq.com	4.3 Seconds	门户
2	Sina.com	5.5 Seconds	门户
3	Yahoo.com.cn	3.8 Seconds	门户+搜索

第 3 节 搜索

本节介绍的搜索压力测试，重点在产品级的测试，对于引擎级的压力测试这里不做介绍。

4.3.1 需求分析

4.3.1.1 业务分析

目前主要常用的搜索服务：网页搜索、图片搜索、音乐搜索、地图搜索等等。在用户的众多操作中，我们认为搜索动作是压力的主要来源。一个搜索频道可能有多个搜索不同类型的搜索结果页面，这个需要我们测试人员和产品人员进行沟通，确认测试的重点和覆盖范围。

对于网页搜索、音乐搜索，一般的测试场景只需要包括：发送搜索请求、返回 html 页面；而对于图片搜索、地图搜索（除了发送搜索请求、返回 html 页面），还要包括从 html 中解析 url 并请求、返回相应的图片。

4.3.1.2 服务器架构

一般的请求过程：用户 HTTP 请求—>Apache 服务器—>应用服务器—>搜索引擎服务器—>应用服务器—>Apache 服务器—>用户；

需要注意：

为了满足庞大的互联网用户，在正式环境中 Apache、应用服务等可能有几十个或几百个集群环境，而在测试环境中可能很难构建这样的环境，往往只有几台前台应用服务器。这时候再使用线上环境的压力进行测试，可能会使前台应用出现瓶颈，而这个瓶颈是在正式环境中不会出现的。

使用正式环境的引擎测试，要尽量在“闲时”进行，同时对引擎服务有足够的信任；如果是构建引擎的测试环境，引擎服务器是否能搭建其像正式环境一样的集群，也是我们要考虑的问题。

4.3.1.3 测试数据

使用正式的引擎库和真实的用户搜索关键词进行测试是最好的。

4.3.2 场景分析

4.3.2.1 脚本步骤

对于网页搜索、音乐搜索，一般的测试场景只需要包括：发送搜索请求、返回 html 页面；

对于图片搜索、地图搜索（除了发送搜索请求、返回 html 页面），还要包括从 html 中解析 url 并请求、返回相应的图片；

测试数据

测试脚本中“关键词”是可参数化的。使用真实的用户搜索关键词作为参数化数据。

4.3.2.2 测试场景

测试脚本是单一的，测试场景也相对简单，重点是检查在不同压力条件下，各项性能指标的变化情况。

压力条件，可以根据每日流量来计算。具体的计算方式这里不再介绍。

性能指标统计，压力场景比较简单，但是结果分析也相当复杂。需要根据最终想要得出的结果，在测试过程中记录不同的数据。

4.3.3 行业指标

2006 年底，Aexa 上统计的各个搜索网站的响应时间：

编号	网站	Avg Load Speed	类型
1	Baidu.com	0.9 Seconds	搜索
2	Yahoo.com.cn	3.8 Seconds	门户+搜索
3	Google.com	0.8 Seconds	搜索
4	Soso.com	2.4 Seconds	搜索
5	Sogou.com	2.1 Seconds	搜索

第 4 节 社区

4.4.1 需求分析

4.4.1.1 业务分析

社区性的站点的主要用户操作可以归结为：随机读、顺序写、随机写。笔者总结几个站点的各个功能流量发现，这三个操作的 pv 占了总流量的 95%以上。

随机读主要包括：读帖；顺序写主要包括：发表主题贴；随机主要包括：回帖。当然，也包含其他操作，这里不做详细介绍。

一般读、写操作的比例约为 10：1，但是写操作一般是系统中的重要功能，涉及到文件系统和数据库的操作，可能产生性能瓶颈。所以也是压力测试的重点。

同时要注意的是未登录用户读写操作和登录用户读写操作的差异性。有些系统，登录普通用户浏览帖子时可能会比未登录用户多连接几次数据库。

社区的其他操作如：投票、检索、搜索等在整体性能评测中也需要测试。但在这里不对他们进行介绍。

4.4.1.2 服务器分析

社区性系统，服务器架构一般都比较复杂，涉及很多单独的服务器。测试前先要分析他

们的关联关系。这里不做详细介绍。

主要的服务器可能包括：Apache 服务器、应用服务器、后台服务器、数据库服务器、文件系统服务器、消息服务器、投票服务器、自然语言检索服务器、引擎服务器等。

测试过程可以准对特定的服务器进行测试，也可以做整体性的测试。根据测试需求而定 。

4.4.1.3 测试数据

2006 年底，某国内大型社区数据如下：
活跃版块：1000；
主题数：12968280；
平均每个主题的回帖数：8 个；
吧名总数：1 万；帖子总数：50 万；活跃吧：500；活跃吧平均帖子数：100 个；
每天登录用户数约：6000－10000 人/次；

4.4.2 场景分析

主要以下几个场景：
场景一：未登录用户进入版块、读 10 个帖子；
场景二：登录用户进入版块、读 10 个帖子；
场景三：登录用户写新的帖子；
场景四：登录用户写回帖；
场景五：以上四个场景的混合场景；
注意，每个场景要考虑不同的压力情况。具体的压力范围根据预计流量来计算。

第 5 节 行业-税务

4.5.1 需求分析

4.5.1.1 业务数据需求

本次性能测试的目标是：在预期的压力下，系统的响应时间要在预期的范围值之内。下面，是确定的进行性能测试的功能点，其他未列入的功能将不作性能测试。

主要功能需求数据如下：

项目 功能点	用户数量	使用频率	使用日期	使用时间	备注
申报	如:20000	如:1 次/月	如:每月 1-15 日	如：每天 7 小时	60%-70% 集中在三天
增值税					.
企业所得税					
消费税					

纳税人申报					
申报上传					
申报表查询					
发票认证					
认证上传					
...					

计算公式如下：

平均天处理并发数=网上用户数÷（使用天数×使用时间）

备注：使用天数是使用日期的范围内的正常工作天数。如：1-15 号，除去两个星期天共 4 天，所以使用天数为 11 天。

使用时间为正常的工作时间，使用时间一般为 6 小时。

日高峰数=网上用户数量 × 月集中访问访问比例 × 使用次数 ÷ 集中使用天数；

备注：如用户量为 100，70%集中在最后三天做最后一次访问，则：

日高峰处理数=100 × 70% × 1 ÷ 3=24 笔/天；

小时高峰数=日高峰数 × 天集中比例÷ 集中使用小时数；

备注：如，接上例，一天中，大概有 80%的处理集中在 4 小时内，则：

小时高峰数=24 × 80% ÷ 4=4.8 笔/小时；

计算后得出如下结果：

性能参数 功能点	平均值（笔/小时）	峰值	
		日峰值（笔/天）	每小时峰值（笔/小时）
申报			
增值税			
企业所得税			
消费税			
纳税人申报			
申报上传			
申报表查询			
发票认证			
认证上传			
...			

4.5.1.2 测试环境

略

4.5.2 场景分析

4.5.2.1 测试场景

场景一：

目标：核实所指定的业务功能在正常的预期工作量的性能行为。

各个业务模块里面的虚拟用户的多少，以及详细脚本设计，直接是根据需求而得来的，性能评价的目标是核实性能需求是否都已满足。

场景二：

目标：核实所指定的业务功能在预期的较为繁重工作量的性能行为：

场景三：

目标：验证系统在超出最大预期工作量的情况下是否能正常运行，同时评估性能特征，性能特征主要是：响应时间及网络流量。

并发的虚拟用户数是大于客户的预期值的，同时不同的功能承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。

场景四：

目的：验证系统是否支持一个比较理想的同时在线人数，预期值为 2000 个。

测试中，分两种情况：1、2000 用户同时登录，访问系统；2、逐渐增加用户数，直到增加到 2000 个。

4.5.2.2 测试报告及分析

收集测试中的数据，包括：脚本运行时间，各功能点的响应时间点。用这些数据与需求进行对比，如果运行时间及各功能点的响应时间点均大于需求，则需要进行性能优化；如果不存在运行时间及各功能点的响应时间点均大于需求，则测试完成。

第 6 节 数据库

对于数据库和文件系统的操作主要包括：读、写、查询、修改、删除等。

4.6.1 需求分析

4.6.1.1 数据库基本目标

- ✧ 验证系统瓶颈是否在数据库；
- ✧ 同其他大型数据库比较；
- ✧ 数据库的增删改查的平均响应时间；
- ✧ 并发用户访问时，数据库的增删改查的平均响应时间；
- ✧ 表分区对数据库响应时间的影响；
- ✧ 系统正常使用时，数据库各个监控参数变化情况；
- ✧ 数据库服务器的系统资源使用情况；

4.6.1.2 测试数据

利用原型系统后台数据库的数据作为测试数据，保证测试结果更加真实。

4.6.2 测试方案

4.6.2.1 场景分析

原型系统业务中各个操作的比例，以及各个操作中的 SQL 使用为场景设计依据。
测试采用的查询语句用原型系统的业务语句，保证测试兼顾实际应用。

4.6.2.2 监控指标

主要监控数据包括：
各种并发量下增删改查的数据库响应时间；
记录 I/O、Cpu 等系统数据
记录数据库各个计数器的值。各种计数器请参照第三章。
测试结束，通过对测试结果进行纵向和横向的比较，来综合分析各家数据库的性能。

第 7 节 文件系统

在很多产品中，数据库存储很难满足它们的要求，需要自己特有的文件系统来存储数据。也有一些即使用文件系统，又使用数据库的例子。

文件系统的主要功能类似于数据库，读、写、删除、修改、索引等等。在功能测试中，除了基本功能和数据正确性检查外，备份和灾难恢复是需要重点检查的。

文件系统的分级存储、文件拆分合并、关系数据、索引等比较复杂。如果需要了解，建议找一本专门的通用存储系统设计书籍来学习。

读写操作是文件系统两个主要功能，响应时间是重要的指标。

4.7.1 需求分析

主要确定如下信息：

测试数据：文件系统中的数据量、单个文件大小、文件存放位置、各种类型（或位置）的文件的比例；读写文件的大小等等。、

其他指标，如并发量、每秒完成请求数、系统资源等等这里不再介绍。

4.7.2 测试场景

顺序写、随机写、顺序读写、局部随机读写、全随机读写等等。不详细介绍了。

注意：读写数据量大、速度快，测试时尽量使用千兆网卡，否则，网卡很容易出现瓶颈。

第 8 节 缓存配置

4.8.1 项目及需求介绍

项目：为了优化服务质量，在系统内加入内存缓存和文件缓存，在正式使用之前，需要通过测试的方式收集各种配置下的性能指标的数据。

需求：主要针对以下配置测试：

配置参数	参考值
并发量	10、50、100、300（次/秒）；<根据测试情况调节>
Cache 大小	2G(filecached)、10G(filecached)、20G(filecached)； 2G(Memcached)、4G(Memcached)、6G(Memcached)；
单个 cache 文件大小	5K、10K；
读取原则	随机读取、在某个范围内读取（5000/10000/50000/更多个文件内）
网络环境	电信、网络 或者其他

测试中收集接收数据的响应时间、吞吐量、出错率以及服务器资源使用情况等等。

4.8.2 场景设计

◆ **场景**

测试场景主要是按照上面配置参数的不同组合来进行设计。

值得注意的是，以上参数的组合非常多，我们不是执行所有的组合，根据测试情况、预期性能指标来确定后面的测试场景。每个场景执行时间在半个小时以内即可。

在所有配置对比测试完成后，选择几组作为最终配置参考的数据，来设计场景，进行对缓存长时间读取的性能指标对比测试。执行时间应该在 1-7 天时间。一般是利用周末 2-3 天的时间。测试详细测试场景这里不做描述。

其他要注意的是：cache 提供的接口不单单是读操作，还可能有插入、删除、修改等操作。在测试中也可以捎带测试。这些功能一般产生压力不大，但是可能存在功能问题，导致 cache 数据错误。

◆ **测试程序**

Cache 数据通过程序读取，这个测试程序一定要和以后的正式使用 cache 的程序保持一致。所以，在开发测试程序时要和架构、开发人员作好沟通。

◆ **测试环境**

在测试 cache 时，系统和各级应用也可能做了自己的 cache，所以要主要每次执行测试之前，都需要重启机器，必要时要重写 cache 数据，然后再重启机器。确保每次测试环境的一致性。

◆ **监控指标**

本次测试中，响应时间是最重要的指标，例如对 cache 的读取时间在 0.01s 左右，那么对附近范围的请求时间需再做细分，划分出每个区间内请求时间的分布；还要注意监控响应时间随时间变化的规律。

第 9 节 接口测试

很多产品是有不同项目组或是公司共同开发完成的，通过相互调用接口实现通讯。那么对方提供的接口、我们提供给其他项目组的接口是否是可靠的、稳定的？这些都需要通过测试来检查。

例如：某项目要用到搜索引擎技术，需要测试 Google 提供的搜索引擎接口，是否满足项目。

4.9.1 需求分析

一般的接口测试都是收集接口的性能指标数据，测试完成后再做分析。主要指标：响应时间、吞吐率、吞吐量、每秒处理请求数、成功率。

开始测试之前先要了解以下这些信息：

测试场景单一：发送搜索请求给 Google，接收 Google 返回结果；

每秒处理请求数根据项目需要而定；

发送请求数据根据实际数据得出；

需要确定接口输入、输出的格式；

确定可用于调试的服务地址端口，和最终测试的服务地址端口；

接口测试一般是在正式环境下进行的，所以要确定测试时间和压力情况等等。注意，由于是合作测试，时间确定时一定要留出充足的时间进行测试设计和测试开发工作，测试执行的时间点一定要保证。

4.9.2 测试注意事项

具体的测试场景设计主要就是压力、负载等等这里不做描述。简单介绍几个注意事项：

- ✧ 如果是要测试 Google 的服务，多准备点测试机吧，少了你摆不平它。^_^ 建议一台测试机每秒处理 100 个请求即可。
- ✧ 要考虑 Google 服务器在电信还是在网通，我们的服务器在电信还是在网通。
- ✧ 需要验证并发时返回结果的正确性。
- ✧ 测试机和 Google 服务器的对应关系。

第 5 章 性能指标

性能测试指标一般可以划分为：业务指标和系统资源指标两大部分，对于一般用户而言，对于系统性能的要求主要是业务指标，而系统资源指标是系统性能的一个反应，它可以帮助分析系统系能瓶颈、优化系统或者去发现一些隐性问题。

对业务指标的要求主要是：请求响应时间、最大并发量等等。

对于系统资源的指标，如：资源使用率是指在系统负载运行期间，数据库服务器、应用服务器、Web 服务器的 CPU、内存、硬盘，外置存储，网络带宽的使用率。低于 20%的使用率为资源空闲，20%-60%的使用率为资源使用稳定，60%-80%的使用率表示资源使用饱和，超过 80%使用率的资源使用率必须尽快进行资源调整和优化。

第 1 节 业务指标

下面列出了一些性能测试中常见的业务指标，这些性能指标如何与需求结合请参照第三章和第四章的内容。

点击率：

点击率除程序处理速度，还受带宽的限制，每个请求的大小情况。请求越小，每秒完成的请求越多。在排除带宽影响的情况下，做了缓存的系统比没做缓存的系统的点击率要高很多。在网络传输到达一定的程度后，点击率就不会随并发量的增长而增大。一般可以在限定的带宽情况下对最大点击率进行估算，公式如下：

最大点击率=带宽/（估算平均每个请求大小 ×8）

一般来说，日访问量是在点击率基础上计算出来的：

日访问量=点击率 ×3600× 日访问小时数（可按 8 小时算）

响应时间：

网页类页面响应时间比较理想的均应该在 5 秒以下，但这个受不同的事务处理情况而不同响应时差有所不同.而上传和下载一般需要较长时间，根据文件大小而不同，我们可以确定一个传输速率作为指标。

事务吞吐量：

每秒完成的事务数。

业务指标中的吞吐量，一般是指单位时间内完成的请求数。而系统资源监控指标中的吞吐量一般是指单位时间内的网络传输量。

在线用户数：

在线用户数是指同时使用应用系统的用户数量。

并发用户数：

并发用户数是指在系统运行期间同一时刻进行业务操作的用户数量。使用频度较低的应用系统并发用户数一般为在线用户数的 5%左右，使用频度较高的应用系统并发用户数一般为在线用户数的 10%左右。

最大并发用户数：

应用系统在正式环境下能承受的最大并发用户数。在运行中，如果出现了频繁业务操作失败、响应时间远远超出用户所能承受的最大值或出现了服务器宕机等情况，则说明系统承载量已经超出了最大并发用户的范围。

事务：

测试中常常引入事务这个概念，一般一个事务包含一个请求。这样可以很方便的统计出每个请求的时间。

业务层面的事务是指完成一次完整的业务操作，例如进行一次取款、查询操作。技术层面的事务是指客户端对服务器的一次请求。

分析系统性能时，事务相应时间是一个很重要的指标。事务相应时间即从发起这个事务开始计时直到事务完成所耗费的时间。在实际测试中，需要统计出每个事务或者请求的响应时间，以便分析相应时间的变化和分布规律。很多时候需要对请求响应时间再做细分，分析出请求中包含的每个组件的响应时间、或者一个请求的各个阶段所消耗的时间（域名解析、建立连接、发送请求、接收请求等等），在需要分析程序中的时间消耗情况来优化程序的测试时，会在程序中加入各个时间统计点，统计程序中各个事务的时间消耗情况。

第 2 节 资源指标

根据测试目的不同，需要统计的系统资源指标也不同。主要包括以下一些：服务器操作系统资源使用情况、各种服务的资源消耗情况等等。下面举例一些系统资源指标。

◆ **内存**

Paging rate：内存页交换速率

如果该值偶尔走高，表明当时有线程竞争内存。如果持续很高，则内存可能是瓶颈。也可能是内存访问命中率低。

Memory\Available bytes：

如果 Memory\Available bytes 计数器的值持续降低，同时 Process\Private Bytes 计数器和 Process\Working Set 计数器的值在长时间内持续升高，则很可能存在内存泄漏。需要更详细的内存监控工具来定位是否有内存泄漏和存在内存泄漏的代码。

内存资源成为系统性能的瓶颈的征兆：

1.很高的换页率(high pageout rate)；

2.进程进入不活动状态；

3.交换区所有磁盘的活动次数可高；

4.可高的全局系统 CPU 利用率；

5.内存不够出错(out of memory)；

◆ **处理器**

CPU 占用率（CPU utilization）

如果该值持续超过 95%，表明瓶颈是 CPU。可以考虑增加一个处理器或换一个更快的处理器。一般可接受的最大上限是 80-85% 合理使用的范围在 60%至 70%以下。

System\Processor Queue Length

如果 System\Processor Queue Length 大于 2，而处理器利用率（Processor Time）一直很低，则存在着处理器阻塞。

CPU 资源成为系统性能的瓶颈的征兆：

1.很慢的响应时间(slow response time) ；

2.CPU 空闲时间为零(zero percent idle CPU) ；

3.过高的用户占用 CPU 时间(high percent user CPU) ；

4.过高的系统占用 CPU 时间(high percent system CPU) ；

5.长时间的有很长的运行进程队列(large run queue size sustained over time) ；

◆ **磁盘 I/O**

Disk rate：磁盘交换率

磁盘交换率（Disk rate），如果该参数值一直很高，表明 I/O 有问题。可考虑更换更快的硬盘系统。

Disk Time、Avg.Disk Queue Length

如果这两个值很高，而 Page Reads/sec 页面读取操作速率很低，则可能存在磁盘瓶径。

I/O 资源成为系统性能的瓶颈的征兆：

1.过高的磁盘利用率(high disk utilization)；

2.太长的磁盘等待队列(large disk queue length)；

3.等待磁盘 I/O 的时间所占的百分率太高（large percentage of time waiting for disk I/O)；

4.太长的运行进程队列，但 CPU 却空闲(large run queue with idle CPU)；

5.太高的物理 I/O 速率（large physical I/O rate(not sufficient in itself)；

6.过低的缓存命中率(low buffer cache hit ratio(not sufficient in itself))；

◆ **网络**

需要监控网络流量、吞吐量，确定网络流量没有达到网络传输的极限。并监控网络传输的速度等数据。

第 3 节 其他指标

◆ **数据库指标**

参见第九章

第 4 节 硬件设备

对系统性能的影响，硬件是很重要的。那么各大厂商的不同的设备工作原理如何、有什么优势、极限指标是多少等等是需要测试人员掌握的。关于硬件的知识需要了解一些，这里只做一个简单的介绍，想了解详细的内容，请参考专业书籍。

第 5 节 行业标准

一般厂商，都会按 TPC（事务处理性能委员会）标准组织的规范进行性能测试，并发布一个报告。我们可以通过流量指标(Throughput，简称 tpmC)来评估该产品的性能状况。

而对于一些未能找到该项报告，或者我们公司自行开发的项目，我们可以参考 TPC 标准组织的性能测试模型及规范来进行测试，发布 TPC-C 规范报告，独立审计机构将负责对基准测试结果进行公证，同时，TPC 将出据一份全面彻底的测试报告。这份测试报告可以从 TPC Web 站点(http://www.tpc.org)上获得。

我们可以从这个网站上得到大部分硬软件的性能状部报告。

◆ **TPC-C**

TPC-C 是一种旨在衡量联机事务处理（OLTP）系统性能与可伸缩性的行业标准基准测试项目。这种基准测试项目将对包括查询、更新及队列式小批量事务在内的广泛数据库功能进行测试。许多 IT 专业人员将 TPC-C 视为衡量“真实”OLTP 系统性能的有效指示器。

TPC-C 基准测试针对一种模拟订单录入与销售环境测量每分钟商业事务（tpmC）吞吐

量。特别值得一提的是，它将专门测量系统在同时执行其它四种事务类型（如支付、订单状态更新、交付及证券级变更）时每分钟所生成的新增订单事务数量。独立审计机构将负责对基准测试结果进行公证，同时，TPC 将出据一份全面彻底的测试报告。这份测试报告可以从 TPC Web 站点(http://www.tpc.org)上获得。

tpmC 定义: TPC-C 的吞吐量，按有效 TPC-C 配置期间每分钟处理的平均交易次数测量，至少要运行 12 分钟。

TPC-C 规范概要

TPC-C 是专门针对联机交易处理系统（OLTP 系统）的，一般情况下我们也把这类系统称为业务处理系统。

TPC-C 测试规范中模拟了一个比较复杂并具有代表意义的 OLTP 应用环境:假设有一个大型商品批发商，它拥有若干个分布在不同区域的商品库；每个仓库负责为 10 个销售点供货；每个销售点为 3000 个客户提供服务；每个客户平均一个订单有 10 项产品;所有订单中约 1%的产品在其直接所属的仓库中没有存货，需要由其他区域的仓库来供货。

该系统需要处理的交易为以下几种：

New-Order：客户输入一笔新的订货交易；

Payment:更新客户账户余额以反映其支付状况;

Delivery:发货(模拟批处理交易);

Order-Status:查询客户最近交易的状态；

Stock-Level:查询仓库库存状况，以便能够及时补货。

对于前四种类型的交易，要求响应时间在 5 秒以内；对于库存状况查询交易，要求响应时间在 20 秒以内。

评测指标

TPC-C 测试规范经过两年的研制，于 1992 年 7 月发布。几乎所有在 OLTP 市场提供软硬件平台的厂商都发布了相应的 TPC-C 测试结果，随着计算机技术的不断发展，这些测试结果也在不断刷新。

TPC-C 的测试结果主要有两个指标：

- **流量指标(Throughput，简称 tpmC)**

按照 TPC 的定义，流量指标描述了系统在执行 Payment、Order-status、Delivery、Stock-Level 这四种交易的同时，每分钟可以处理多少个 New-Order 交易。所有交易的响应时间必须满足 TPC-C 测试规范的要求。

流量指标值越大越好！

- **性价比(Price/Performance，简称 Price/tpmC)**

即测试系统价格（指在美国的报价）与流量指标的比值。性价比越小越好！

要想了解更多的信息，请从 TPC Web 站点(http://www.tpc.org)上获得。

第 6 章 Unix/Linux

对于应用在服务器上性能的测试，可以采用工具监控，也可以使用系统本身的监控命令，例如可以使用 Top 命令监控资源使用情况。实施测试的目的是实现服务器设备、服务器操作系统、数据库系统、应用在服务器上性能的全面监控。

UNIX 资源监控指标和描述：

- 平均负载：系统正常状态下，最后 60 秒同步进程的平均个数 ；
- 冲突率：在以太网上监测到的每秒冲突数 ；
- 进程/线程交换率：进程和线程之间每秒交换次数 ；
- CPU 利用率：CPU 占用率（％）；
- 磁盘交换率：磁盘交换速率 ；
- 接收包错误率：接收以太网数据包时每秒错误数 ；
- 包输入率：每秒输入的以太网数据包数目 ；
- 中断速率：CPU 每秒处理的中断数 ；
- 输出包错误率：发送以太网数据包时每秒错误数 ；
- 包输入率：每秒输出的以太网数据包数目 ；
- 读入内存页速率：物理内存中每秒读入内存页的数目 ；
- 写出内存页速率：每秒从物理内存中写到页文件中的内存页数目或者从物理内存中删掉的内存页数目 ；
- 内存页交换速率：每秒写入内存页和从物理内存中读出页的个数 ；
- 进程入交换率：交换区输入的进程数目 ；
- 进程出交换率：交换区输出的进程数目 ；
- 系统 CPU 利用率：系统的 CPU 占用率（％）；
- 用户 CPU 利用率：用户模式下的 CPU 占用率（％） ；
- 磁盘阻塞：磁盘每秒阻塞的字节数；

第 1 节 常见工具

目前，对系统进行性能调试的工具有很多，这些可以两大类：一类是标准的分析工具，即所有的 UNIX 都会带的分析工具； 另一类是不同厂商的 UNIX 所特有的性能分析工具，比如 HP-UX 就有自己的增值性能分析工具。

标准的分析工具，即所有的 UNIX 都会带的分析工具： sar、iostat 、vmstat 、time 、ps 、bdf 、top 、ipcs、uptime ；

HP-UX 自己的增值性能分析工具： glance/gpm 、puma 、xps ；

按监控对象来说，它可以分为： CPU 的使用情况：sar,time,top,ps,puma,xps ； 内存的使用情况：vmstat,ipcs ；文件系统状态：dbf,iostat,sar,swapinfo,nfsstat ；I/O 子系统状态：iostat ；网络性能：netstat；

cat /proc/meninfo：查看系统的总 men 大小

cat /proc/cpuinfo：查看系统总 CPU 大小

df –k：查看系统硬盘大小

（2）查看内存使用情况的命令 51Testing 软件测试网y g D(V!{ D%d%^ S

用 free 命令查看内存占用情况：$ free51Testing 软件测试网;r T$X M W/F b.`l

top 然后按 Shift+M, 按照进程内存占用率排序：$ top

（3）查看网络流量

可以用工具 iptraf 工具：$ iptraf -g

针对某个 Interface 的网络流量可以通过比较两个时间网络接口的 RX 和 TX 数据来获得：$ date; ifconfig eth1 或$ date; ifconfig eth1

6.1.1 iostat

在命令行收集有关 CPU、磁盘、终端和磁带输入/输出操作的数据；

它的语法为： iostat [-t] [interval [count]] ；

#iostat -x 60 10 在 60 秒内产生 10 个统计报告，输出

iostat –x

extended device statistics

device	r/s	w/s	kr/s	kw/s	wait	actv	svc_t	%w	%b
fd0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
md20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
md21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0

……

device 显示设备名；r/s 显示每秒读磁盘操作的次数；w/s 显示每秒写磁盘操作的次数；kr/s 显示每秒读数据总量、单位 K；kw/s 显示每秒写数据总量、单位 K；wait 显示平均的等待事务数量；actv 显示正在处理的平均事务总量；svc_t 显示凭据服务周期、单位 ms；%w 显示等待时间的百分数；%b 显示磁盘工作时间的百分数；

通过查看 bps 列和 sps 列的值我们可以知道哪些磁盘比较忙，哪些磁盘比较闲。

6.1.2 vmstat

它的语法： vmstat [-dnS] [interval [count]] ；vmstat -f | -s | -z ；各个命令参数解释略，请读者自己参考命令帮助信息。

在命令行收集虚拟内存性能的数据。

kthr			memory			page				disk				faults		cpu			
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	f0	m2	m2	m2	in	sy	cs	us sy id
0	0	0	156352	168880	29	94	147	0	0	0	67	0	0	0	0	441	481	76	6 4 89

procs：

r—>在运行队列中等待的进程数 ；b—>在等待 io 的进程数 ；w—>可以进入运行队列但被替换的进程

memoy ：

swap-->现时可用的交换内存（k 表示） ；free—>空闲的内存（k 表示）；

pages ：

re—>回收的页面 ；mf—>非严重错误的页面 ；pi—>进入页面数（k 表示） ；po—>出页面数（k 表示） ；fr—>空余的页面数（k 表示）；de—>提前读入的页面中的未命中数 ；sr—>通过时钟算法扫描的页面 ；

disk：

显示每秒的磁盘操作： s 表示 scsi 盘，0 表示盘号

fault：

显示每秒的中断数：in－>设备中断；sy－>系统中断 ；cy－>cpu 交换 ；

cpu ：

表示 cpu 的使用状态 ：us－>用户进程使用的时间 ；sy－>系统进程使用的时间 ；id－>cpu 空闲的时间 ；

如果 r 经常大于 4 ，且 id 经常少于 40，表示 cpu 的负荷很重。

如果 pi，po 长期不等于 0，表示内存不足。

如果 disk 经常不等于 0， 且在 b 中的队列 大于 3， 表示 io 性能不好。

在不带参数的 vmstat 的命令时，我们首先要关注的是 avm(active virtual memory)列和 free(free list zise)列的值。如果 avm 的值很大，而 free 的值却很小，这时，系统可能有内存瓶颈，我们 必须用带-S 选项的 vmstat 命令查看系统是否有 deactivation/reactivation 活动正在发生。

6.1.3 vmstat/iostat

指标名称	指标描述	指标范围
Processor Queue Length	处理器队列的线程数量。此计数器只显示就绪线程，而不是正在运行的线程。	
Processes Blocked	等待 I/O 资源的进程数	
Processes Swapped out	被 swap 到磁盘的进程数	
Free Swap Space	交换分区的可用空间	
KB transferred/sec	每秒磁盘读写量。 每个磁盘有一个指标值。	
Transfers/sec	每秒磁盘传输次数。 每个磁盘有一个指标值。	
Service time	磁盘平均处理时间	

6.1.4 sar

利用 SAR 进行 CPU 的利用率分析的命令形式：

#sar -u，这时数据是通过 sa1 在后台定时生成；

#sar -u 5 100，每隔 5 秒取样一次，共取 100 次；

◆ 对结果的分析 ：

首先，我们看%idle 列的值，如果为接近零，则再看对应%wio 列的值，如果这列的大于 7，则表明系统的磁盘或其他 I/O 可能有问题，需要进一步的分析：

如果%idle 列很小，而对应的%wio 列的值也很小，这时，我们查看%usr 列和%sys 列的值。如果%usr 列的值很大，说明有用户进程占用很多 CPU 时间；如果%sys 列的值很大，则说明系统管理方面花了很多时间。需要进一步的分析：

利用 SAR 工具分析运行进程队列长度

利用 SAR 进行运行进程队列长度分析的命令形式：

#sar -q，这时数据是通过 sa1 在后台定时生成；

#sar -q 5 100，每隔 5 秒取样一次，共取 100 次；

◆ 对结果的分析：

这些数据越小越好。

如果 runq-sz 大于 4，或者%swapocc 大于 5 时，则表明系统的 CPU 或内存可能有问题，需要进一步的分析：

用 sar -u 命令分析 CPU 的使用情况；

用 sar -w 命令分析进程的 deactivation/reactivation and switching activities of the system;

利用 SAR 工具分析系统调用

利用 SAR 进行系统调用分析的命令形式：

#sar -c，这时数据是通过 sa1 在后台定时生成；

#sar -c 5 100，每隔 5 秒取样一次，共取 100 次；

◆ 对结果的分析：

如果 scall/s 列的值很大，那么这么多的系统调用的原因就必须仔细分析了。

我们可以查看 fork/s 和 exec/s 列的值，看看系统是否在创建大量新的进程。

利用 SAR 命令分析磁盘活动

通过命令 sar -d，我们可以分析系统中的每个磁盘和磁带的活动情况。

◆ 对结果的分析：

如果某个磁盘的%busy 列的值大于 50%，则说明该磁盘可能存在瓶颈；

如果某个磁盘的 avwait 珍的值大于 avserv 列的值，也说明该磁盘可能存在瓶颈；

利用 SAR 命令分析缓冲区的活动

通过命令 sar -b，我们可以分析系统中的缓冲区的活动情况。

◆ 对结果的分析：

如果%rcache 列的值小于 90%，并且%wcache 列的值不在 70-70%之间，我们必须观察系统中什么应用在做什么样的读/写操作，我们是否需要增加缓冲欧的大小。

利用 SAR 命令分析交换区的活动

通过命令 sar -w，我们可以分析系统中的交换区的活动情况。

◆ 对结果的分析：

如果 swpin/s 的值大于零，那么 swpot 的值必须引起注意；

同时必须注意 pswch/s 的值，如果很大，说明进程切换频繁。

6.1.5 top

利用 top 命令查看最耗 CPU 资源的进程

我们可以利用 top 命令来查看最耗 CPU 资源的进程。top 命令还会根据进程占用 CPU 资源的多少而动态改变。

它的语法为： top [-s time] [-d count] [-q] [-u] [-h] [-n number]

其中各选项的含义为：

-s time: 屏幕刷新的时间间隔 time，缺省为 5 秒； -d count: 屏幕刷新 count 次后，top 命令自己也退出； -q；-u: User ID (uid) ；-n number: ；

在 top 命令运行时，我们可用以下几个快捷键来翻屏： j: 向前翻； k: 向后翻； t: 回到第一页；

对结果的分析：

通过 top 命令，我们可以快速了解到目前系统的 CPU 资源使用情况，尤其是占用 CPU 资源最多的进程是我们必须关注的对象。

我们通过 RES(the current size of the process resident in memory)列可以知道每个进程占用内存的数量。

我们通过 NICE 列可以知道系统是否使用 NICE 值来调节该进程的工作负载平衡。

6.1.6 Rstat

指标名称	指标描述	指标范围
Context Switches/sec	指计算机上的所有处理器全都从一个线程转换到另一个线程的综合速率。当正在运行的线程自动放弃处理器时出现上下文转换，由一个有更高优先就绪的线程占先或在用户模式和特权(内核) 模式之间转换以使用执行或分系统服务	如果此计数器的数值较大，则表明锁定竞争很激烈，或者线程在用户和内核模式之间频繁切换。
Interrupts/sec	每秒钟设备中断处理器的次数。在完成一个任务或需要注意时，装置会发出中断讯号给处理器。可以产生中断的装置包括系统定时器、鼠标、数据通讯联机、网络卡以及其它的外部装置。在中断过程中，一般的执行绪执行将被暂停，而且一个中断可以使处理器切换到另一个具有较高优先等级的执行绪。频率中断是频繁和周期性的，并且中断动作在背景执行。	取决于处理器，越低越好；不宜超过 1,000；如果该值显著增加而系统活动没有相应的增加，则表明存在硬件问题，需要检查引起中断的网络适配器、磁盘或其他硬件。
CPU 利用率（CPU Usage：System,User,Idel,wait）	CPU 利用率的相关指标： System：处理系统调用所占的比率； User：处理用户应用调用所占的比率 Idle：空闲率 Wait：等待率	
磁盘读写次数	磁盘每秒读写次数。每个磁盘一个指标值，对比不同的值，发现相应的操作繁忙程度。	
页面切换率（Pages In/Pages Out）	每秒从磁盘读入或从写入磁盘的页面数	
系统平均负载（Load Average）	系统平均负载被定义为在特定时间间隔内运行队列中的平均进程树。如果一个进程满足以下条件则其就会位于运行队列中：1）它没有在等待 I/O 操作的结果；2）它没有主动进入等待状态(也就是没有调用'wait')；3）没有被	只要每个 CPU 的当前活动进程数不大于 3 那么系统的性能就是良好的，如果每个 CPU 的任务数大于 5，那么就表示这台机器的性能有严

	停止(例如：等待终止)。 Load Average 根据时间段不同，分为 1、5、15 分钟三个子指标。	重问题。从 Load Average 指标看的话，则是：指标值/CPU 个数 ＜5。
网络使用情况：：（Received Packets，Sent Packets， Input Errors，Output Errors）	网络使用情况统计	

第 2 节 CPU 性能分析

6.2.1 衡量 CPU 闲忙程度的指标

要分析系统的 CPU 资源是否够的前提谁占用了 CPU 资源，占用了多少，时间多长。 下面是一些衡量 CPU 闲忙程度的经用指标：

1)用户使用 CPU 的情况 ： CPU 运行常规用户进程；CPU 运行 niced process ； CPU 运行实时进程；

2)系统使用 CPU 的情况 ：用于系统调用 ；用于 I/O 管理：中断和驱动 ；用于内存管理：paging and swapping ；用于进程管理：context switch and process start ；

3)WIO：由于进程等待 I/O 而使 CPU 处于空闲状态的比率，这些 I/O 主要指 block I/O,raw I/O,VM paging/swapins；

4)CPU 的空闲率，即除了上面的 WIO 以外的空闲情况；

5)CPU 用于上下文交换的比率(Context Switch CPU utilization)

6)nice

7)real-time

8)运行进程队列的长度，即处于可运行状态的进程个数的大小，不过我们关心的是这些在等待 CPU 调度执行时所花的时间；

9)平均负载(load average)

6.2.2 CPU 资源成为系统性能的瓶颈的征兆

CPU 就像人的大脑，完成各种交给它的任务。如果任务太多，CPU 就要忙不过来，它的运行效率就要下降。就像人生病会有一典型症状一样，当 CPU 资源 成为系统性能的瓶颈时，它也有一些典型的症状：

很慢的响应时间(slow response time)

CPU 空闲时间为零(zero percent idle CPU)

过高的用户占用 CPU 时间(high percent user CPU)

过高的系统占用 CPU 时间(high percent system CPU)

长时间的有很长的运行进程队列(large run queue size sustained over time)

必须注意的是，如果系统出现上面的这些症状并不能说一定是由于 CPU 资源不够，事实，有些症状 的出现很可能是由于其他资源的不足而引起，如内存不够时，CPU 会忙内存

管理的事，这时从表面上， CPU 的利用是 100%，甚至显得不够，如果据此就简单地认为增加 CPU 就可以解决问题是大错特错了。

◆ 哪些进程是占用 CPU 资源的大户?

在操作系统中，并不是所有的进程都以同样的方式使用 CPU 资源。通常情况下，有些进程需要比其他进程更多的 CPU 时间片才能顺利地完成任务。下面是一些典型的占用 CPU 资源的大户：

进程创建(process creation)

终端字符进程(teminal character processes(MUX- and LAN-based)

计算密集型进程和实时进程

X-终端和 X-服务器进程(X-terminals and X-servers)

6.2.3 对 CPU 需求密集型系统的性能优化

基于硬件的方法

升级到更快的 CPU； 升级到更大的高速缓存； 增加 CPU 个数； 把用分布到多个系统中； 使用无盘结点； 增加浮点处理器；

基于软件的方法

在不是高峰时间运行批处理； Nice umimportant application; 使用 rtpio 命令来帮助重要的应用； 使用 plock 命令来帮助重要的应用； Turn off system accounting; Consider using Taskbroker or DCE;

优化应用

考虑使用进程资源管理器(Process Resource Manager)，不过 PRM 只有在 HP-UX 平台上有。

在 linux 中,process 有两种状态:

1.runnable

2.blocked waiting for an event to complete

一个 blocked 状态的 process 可能在等待一个 I/O 操作获取的数据,或者是一个系统调用的结果

如果一个 process 在 runnable 状态,这就意味着它将同其他 runnable 状态的 process 等待 CPU 时间,而不是立即获得 CPU 时间,一个 runnable 状态的 process 不需要消耗 CPU 时间,只有当 Linux 调度进程从 runnable 队列中选择哪个 process 下次执行.当 process 在 runnable 状态,当时等待 CPU 时间时,他们形成的等待队列称作 Run Queue.Run Queue 越大,表示等待的队列越长.

性能工具通常显示 runnable processes 的数目和 blocked processes 的数目.还有一个很常见的系统状态是 load average,系统的 load 是指 running 和 runnable process 的总和.例如:如果有两个 processes 在 running 和有三个在等待运行(runnable),那么系统的 load 为五.load average 是指在指定时间内 load 的平均值.一般 load average 显示的三个数字的时间分别为 1 分钟,五分钟和十五分钟.

◆ **Context Switches**

大部分现在的 CPU 在同一时间只能运行一个 process.虽然也有一些 CPU,例如超线程技术的 CPU,能实现同时运行超过一个 process,linux 把这种 CPU 看作多个单线程 CPU.

linux 内核不断的在不同 process 间切换,造成一个错觉,让人感觉一个单 CPU 同时处理多个任务.不同 process 之间的切换称作 Context Switch.当系统做 Context Switch 时,CPU 保存所

有 old process 的 context 信息并获得 new process 的所有 context 信息.Context 信息包括大量的 linux 追踪每个 process 信息,尤其是一些资源:那些 process 正在执行,被分配了哪些内存,它打开了那些文件,等等.切换 Context 会触发大量的信息移动,这是比较高的开销.如果可能的话尽量保持很小的 context switches.

为了尽可能的减小 context switches,你首先需要知道它们是怎么产生的.首先,kernel 调度触发 context switches.为了保证每个 process 平等的共享 CPU 时间,kernel 周期性中断 running 的 process,如果合适,kernel 调度器会开始一个其他的 process 而不是让当前的 process 继续执行,每次的周期性中断或者定时中断都可能触发 context switch.每秒定时中断的次数因不同架构和不同的 kernel 版本而不同.获取每秒中断次数的一个简单办法是通过监控 /proc/interrupts 文件,看下面的例子:

root@localhost asm-i386]# cat /proc/interrupts | grep timer; sleep 10 ; cat /proc/interrupts | grep timer

0: 24060043 XT-PIC timer

0: 24070093 XT-PIC timer

上面可以看到在指定的时间内 timer 次数的变化,每秒产生的中断次数为 1000 次.如果你的 context switch 比 timer 中断大很多.那么 context switch 更多的可能是 I/O 请求或者其他长时间的系统调用(比如 sleep)产生.当一个应用请求一个操作不能立即实现时,kernel 开始 context switch 操作:存入请求的 process 并且试着切换到其他 runnable process.这将使得 CPU 保持工作状态.

- **Interrupts**

其他方面,CPU 接收硬件驱动发出的中断请求.这种中断通常被触发当一个驱动器有一个时间需要被 kernel 操作时.例如:如果一个磁盘控制器从磁盘上取得了一个数据块和 kernel 需要读取使用这个块,那么磁盘控制器会触发一个中断.kernel 接收每个中断,一个中断处理器运行如果这个中断被注册,否则,这个中断被忽略.在系统中,中断处理器的优先级非常高,而且执行速度非常快.很多时候,有些中断处理并不需要很高的处理优先级,所以也有 soft- interrupt handler.如果有很多的中断,kernel 需要花费大量的时间去处理中断.可以检查/proc/interrupts 能够知道中断发生在哪个 CPU 上.

- **CPU Utilization**

CPU Utilization,一个很直观的概念,在任意时间内,CPU 有 7 个状态:

1.idle,表示 CPU 闲置并等待工作分配.

2.user,表示 CPU 在运行用户的进程

3.system,表示 CPU 在执行 kernel 工作

4.nice,表示 CPU 花费在被 nice 改变过优先级的 process 上的时间(注意:被 nice 命令改变优先级的 process 仅指那些 nice 值为负的 process.花费在被 nice 命令改变优先级的任务上的时间也将被计算在系统和用户时间内,因此整个时间加起来可能会超过百分之百)

5.iowait,表示 CPU 等待 IO 操作完成的时间

6.irq,表示 CPU 开销在响应硬中断上的时间

7.softirq,表示 CPU 开销在响应软中断上的时间.

我们一般用 vmstat 看到的都是四个状态:sy,us,id,wa,通过他和 load avg 结合，基本可以知道 cpu 的状态

大部分的性能工具用百分比表示 CPU 时间.当 system 时间占用很高的时候,你可以用"oprofile"工具发现时间都花费在哪里.当 iowait 很高的时候,你需要分析你的 IO 设备,比如磁盘,网卡.

第 3 节 MEM 性能分析

6.3.1 内存管理

- **内存管理的主要工作：**

  跟踪内存的使用和可用内存的情况；

  为进程分配内存；

  管理磁盘与物理内存之间的换页(paging);

- **2)什么是虚拟内存(virtual memory)?**

virtual memory uses a disk as an extension of RAM so that the effective size of usable memory grows correspondingly. The kernel will write the contents of a currently unused block of memory to the hard disk so that the memory can be used for another purpose. When the original contents are needed again, they are read back into memory. This is all made completely transparent to the user; programs only see the larger amount of memory available and don't notice that parts of them reside on the disk from time to time. Of course, reading and writing the hard disk is slower (on the order of a thousand times slower) than using real memory, so the programs don't run as fast. The part of the hard disk that is used as virtual memory is called the swap space.

物理内存(physical memory) ：-

swap space：-

- **3)paging**

A technique by which the contents of a virtual memory address(called pages) are moved from virtual memory(the disk)into and out of the main memory where it is accessed by the CPU.

这个机制是由一个叫 vhand 的进程来完成。

当可用内存的数量小于 LOTSFREE 时，例程 pageout 将被调用来选择什么内存可以释放。它采用 two-handed clock algorithm,the reference hand turn off the reference bit of each memory page it references.If the refernce bit is still zero when the second hand gets to it,the page is freed.If the page is clean(unmodified),it is added to the freelist.If the page is dirty,it must be posted to the swap device before being put on the freelist.

lotsfree: based on physical memory (64 MB -> 863) upper bound where paging starts/stops

desfree: based on physical memory (64 MB -> 215)lower bound where paging starts/stops

minfree: based on physical memory (64 MB -> 53) threshold where deactivation starts

- **4)Page fault**

An invalid address error(trap)that occurs when the CPU requests a page from memory that has not been paged in from the disk(virtual memory)to the main memory. The page may also have been paged out from the main memory prior the request.

- **5)Process Deactivaton**

Deactivating a process so its memory pages will be flagged free or paged out rapidly by vhand.

- **6)Thrashing**

A situation in which a process is spending more time paging than processing;a high number of page faults is occuring.

- **7)Buffer Cache**

它是内存的一部分，用于加快文件存取时间；

缓存的大小可以随可用内存动态变化，但也可以通过修改内核参数而改成固定的大小；
缓存可以提高磁盘的读/写性能；
在缓存的内容可以通过 sync 进程来强制写入磁盘；
从缓存的读和写又称为逻辑读和逻辑写；

- **8)内存需求**

按用途来分，内存可以分成两部分：预留内存和动态内存。
预留内存主要用于存放：
system table
data structures
buffer cache
其中系统表和数据结构占用的数量一般很小，但缓存则可能占到很大一部分。
动态内存主要用于存放：
process text
data stack
share memory segments
其中各进程锁定的内存会影响动态内存的大小。

1.SWAP 这个指标直接反映了系统内存问题,如果很多的 SWAP 空间被使用,一般你的系统内存不足

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b    swpd    free    buff  cache   si   so    bi    bo   in   cs us sy id wa
 2  1 131560    2320    8640  53036    1    9   107    69 1137  426 10  7 74  9
 0  1 131560    2244    8640  53076  480    0   716     0 1048  207  6  1  0 93
 1  2 132476    3424    8592  53272  832  916  1356   916 1259  692 11  4  0 85
 1  0 132476    2400    8600  53280  764    0  1040    40 1288  762 14  5  0 81
```

这是一个典型的内存不足导致内存被频繁的交换到 swap 上,造成大量的 IO 操作.

2.Cache 当系统内存比应用需要的内存多的时候,系统会把数据 cache 在内存中,减少费时的 IO 操作

3.Buffer 当应用程序在做写磁盘操作时,系统会把要写的数据 buffer 起来,等到一个触发点的时候,再把 buffer 中的数据刷新到磁盘,减少应用程序等待 IO 操作的时间.

一般你在看一个运行长时间的系统时,会发现它的可用内存很少,但是 Cache 和 Buffer 很大,这是正常情况,系统会利用你的所有自由内存来 cache 数据,当它需要分配内存的时候,会把 cache 的数据放回到 disk。

6.3.2 衡量内存闲忙程度的指标

1) 整体内存忙闲指标：
buffer cache size: 缓存区在内存开销中占很大比例；
page in/out rates;
swap in/out rates;
可用内存的大小，或用得到内存的大小(available memory size):
自由内存的大小(free memory size): what is currently available,it should not be confuzed with available memory,which does not change during normal sytem operation;
swap queue length;

2) 单个进程的内存衡量指标：
一个进程占用物理内存的大小(resident set size)
一个进程占用虚拟内存的大小(virtual set size)
VM reads and writes: it can show how many physical memory management reads and writes were made to and from the disk during the chosen interval.

6.3.3 内存资源成为系统性能的瓶颈的征兆

当内存资源成为系统性能的瓶颈时，它有一些典型的症状：

很高的换页率(high pageout rate):HP-UX 是一个按需调页的操作系统，通常情况下，它只执行调入页面进入内存的操作，以让进程能够运行。只有操作系统觉得系统需要释放一些内存空间时，才会执行从内存调出页面的操作，而过高的调出页面操作说明内存缺乏；

进程进入不活动状态(process deactivation activity): 当自由的内存页面数量小于 MINFREE 时，很多进程将强制进入不活动状态，因为，any deactivation activity represents a condition in which normal paging is inadequate to handle the memory demands.

自由内存的数量很小，但活动的虚拟内存却很大(very small free memory and large active virtual memory)

交换区所有磁盘的活动次数可高(high disk activity on swap devices)

可高的全局系统 CPU 利用率(high global system CPU utilization):

很长的运行进程队列，但 CPU 的空闲时间却很多(large run queue with idle CPU)

内存不够出错(out of memory errors)

CPU 用于 vhand 和 swapper 两中守护进程的时间(CPU time to vhand and swapper)

必须注意的是，有时候我们发现 CPU 很忙，这似乎是 CPU 资源成为系统性能的瓶颈，但如果进一步分析，发现 vhand 和 swapper 守护进程占用了大量的系统 CPU 时间，很显然，这时系统性能瓶颈真正所在可能是内存。

6.3.4 哪些进程是占用内存资源的大户

下面是一些典型的占用内存资源的大户：

buffer cache

plocked process：plocked processes are those that are locked in memory and are not elogible to be paged.通常，这些进程都是一些比较重要的进程，不便调出内存。

档案库(archive libraries)：把库放入内存可以加快程序的执行，但它们将占用大量的内存；

共享内存(shared memory)

关系型数据库(relational databases)

X-终端和 X-服务器进程(X-terminals and X-servers)：通常，一个 X-终端需要额外的 2-4 兆内存；一个 X-服务器需要 400KB 以上的内存；

利用 ipcs 分析消息队列、共享内存和信号量 它的语法： ipcs [-mqs] [-abcopt] [-C core] [-N namelist]

6.3.5 对内存需求密集型系统的性能调试

基于硬件的方法

增加物理内存；使用无盘工作站替代 X-terminal；

基于软件的方法

减小内核参数 maxdsiz 的值； 减少内存锁定的使用； 杀死不必要的进程； 识别出需要大量内存的进程； 重新设计应用； 减小内核的大小； 减小系统表的大小； 减小缓存区的大小；

利用 serialize 命令来调度大进程的系统资源需求

/usr/bin/serialize applicaiton ； /usr/bin/serialize -p pid

第 4 节 I/O 性能分析

6.4.1 衡量 I/O 闲忙程度的指标

◆ 下面是一些衡量 I/O 闲忙程度的经用指标：

磁盘利用率(disk utilization) ；磁盘队列长度(disk queue length) ；磁头/逻辑卷的读/写速率(read/write rates per spindle/logical volume) ；原始 I/O(raw I/O)：主要用于数据库应用 ；交换队列的长度(swap queue length) ；缓存命中率(buffer cache hit ratio) ；网络文件系统和无盘工作站速率(NFS and diskless rates(server)) ；

I/O 资源成为系统性能的瓶颈的征兆 ：

◆ 当 I/O 成为瓶颈时，会出现下面这些典型的症状：

过高的磁盘利用率(high disk utilization)

太长的磁盘等待队列(large disk queue length)

等待磁盘 I/O 的时间所占的百分率太高(large percentage of time waiting for disk I/O)

太高的物理 I/O 速率:large physical I/O rate(not sufficient in itself)

过低的缓存命中率(low buffer cache hit ratio(not sufficient in itself))

太长的运行进程队列，但 CPU 却空闲(large run queue with idle CPU)

6.4.2 哪些活动是占用 I/O 资源的大户?

下面是一些占用大量 I/O 资源的活动：

换页(paging):paging 不仅会引起内存问题，还可能引起磁盘问题；

open,creat,and stat system calls：系统调用会引起大量的磁盘 I/O；

multiuser I/O and random I/O ； relational database ； core dumps ；

第 7 章 Windows

本节信息出自 windows 帮助文档，需要了解更多 windos 详细信息请参考 windows 帮助文档的监控性能模块。

Windows 自身的监控计数器很强大，需要认真去了解它。可以查看 windows 的帮助文档获取到所有需要的知识。

第 1 节 对象瓶颈分析

◆ **性能对象**

性能测试中，需要关注如下对象，缓存 、内存 、页面文件 、物理磁盘 、处理器 、服务器 、系统 、线程 。

◆ **瓶颈原因**

由于如下原因，对系统资源的需求可能太高足以导致资源瓶颈：

资源不足，并且需要附加或升级的组件。

资源共享工作负载不平均，需要平衡。

资源出现故障，需要替换。

程序独占特定的资源；这可能需要用另一个程序代替，让开发人员重新编写该程序，添加或升级资源，或在需求较低时运行该程序。

资源不正确，需要更改配置设置。

◆ **常用指标：**

ProcessorTime： 指服务器 CPU 占用率，一般 平均达到 70%时，服务就接近饱和；

Memory Available Mbyte ： 可用内存数，如果测试时发现内存有变化情况也要注意，如果是内存泄漏则比较严重；

Physicsdisk Time ： 物理磁盘读写时间情况；

第 2 节 设置监控方案

在执行性能测试时，需要根据实际情况选择自己需要的系统资源监控器。下表显示了对监视服务器推荐使用的最少计数器。检查特定资源时，应包含与性能对象有关的其他计数器 。

组件	监视方面	要监视的计数器
磁盘	使用	Physical Disk\ Disk Reads/sec Physical Disk\ Disk Writes/sec LogicalDisk\ % Free Space 请小心处理 % Disk Time 计数器。因为该计数器的 _Total 实例不能精确反映多磁盘系统的利用率，因此使用 % Idle Time 计数器也非常重要。注意这些计数器不能显示超过 100% 的数值。

磁盘	障碍	Physical Disk\Avg.Disk Queue Length（所有实例）
内存	使用	Memory\Available Bytes Memory\Cache Bytes
内存	障碍	Memory\Pages/sec Memory\PageReads/sec Memory\TransitionFaults/sec Memory\Pool Paged Bytes Memory\Pool Nonpaged Bytes 尽管没有明确的 Memory 对象计数器，但下面的对象对内存分析还是有用的： Paging File\%Usage 对象（所有实例） Cache\Data Map Hits% Server\Pool Paged Bytes 和 Server\Pool Nonpaged Bytes
网络	吞吐量	协议传输计数器（随网络协议不同而不同）；对于 TCP/IP： Network Interface\Bytes total/sec Network Interface\ Packets/sec Server\Bytes Total/sec 或 Server\Bytes Transmitted/sec 和 Server\Bytes Received/sec 您可能要监视在监视网络活动中描述的网络和服务器吞吐量的其他对象。
处理器	使用	Processor\% Processor Time（所有实例）
处理器	障碍	System\Processor Queue Length（所有实例） Processor\ Interrupts/sec System\Context switches/sec

到目前为止，内存不足是计算机系统中的严重性能问题最常见的原因。如果怀疑存在其他问题，请检查内存计数器以排除内存短缺问题。工作站响应速度很慢最有可能是内存和处理器问题造成的；服务器更容易受磁盘和网络问题的影响。

以下列出了要监视的资源的调节提示：

内存：

增加物理内存，使之超过所需的最小内存。

使用多个磁盘时创建多个页面文件。

决定页面文件的正确大小。建议页面文件的大小应等于系统可用 RAM 的 1.5 倍。

确保内存设置已正确配置。

在最高性能的计算机上或当系统工作负载较轻时运行需要大量内存的程序。

磁盘：

升级为更高速度的磁盘或添加磁盘。这样做时，请升级磁盘控制器和总线。

在服务器上，使用“磁盘管理”在多个物理磁盘上创建带区卷。这种解决方案增加了吞吐量，因为 I/O 命令可以同时发布。

在服务器之间分配程序。分布式文件系统 （DFS） 可以用来平衡工作负载。

将使用磁盘 I/O 很多的任务隔离在单独的物理磁盘或磁盘控制器上。

使用磁盘整理碎片程序来合并文件，以优化数据访问和磁盘空间。

如果要提高磁盘访问的效率，可以考虑安装最新的主机适配器驱动程序软件。与适配器制造商联系以获取有关信息。

处理器：

添加处理器（尤其对于多线程程序）或升级为更快的处理器。

在多处理器计算机中，管理与处理线程和中断有关的处理器相似性。

网络：

配置您的网络，使由同一组人员共享的系统位于同一子网上。

解除很少使用的网卡的绑定。

第 3 节 指标可接受值

资源	对象\计数器	建议的阈值	注释
磁盘	Physical Disk\% Free Space Logical Disk\% Free Space	15%	
磁盘	Physical Disk\\% Disk Time Logical Disk\% Disk Time	90%	
磁盘	Physical Disk\Disk Reads/sec、Physical Disk\Disk Writes/sec	取决于制造商的规格	检查磁盘的指定传送速度，以验证此速度没有超出规格。通常，Ultra Wide SCSI 磁盘每秒可以处理 50 到 70 次 I/O 操作。
磁盘	Physical Disk\Current Disk Queue Length	主轴数加 2	这是即时计数器；观察在多个间隔上的值。对于随时间变化的平均值，请使用 Physical Disk\ Avg.Disk Queue Length。
内存	Memory\Available Bytes	少于 4 MB	考察内存使用情况在需要时添加内存。
内存	Memory\Pages/sec	20	研究页交换活动。
页面文件	Paging File\% Usage	70% 以上	与 Available Bytes 和 Pages/sec 一起复查该值，了解计算机的页交换活动。
处理器	Processor\% Processor Time	85%	查找使用处理器时间高百分比的进程。升级到更快的处理器或安装其他处理器。
处理器	Processor\Interrupts /sec	取决于处理器；每秒 1000 次中断是好的起点	此计数器的值明显增加，而系统活动没有相应的增加则表明存在硬件问题。标识导致中断的网卡。可能需要安装额外的适配器或者控制器卡。
服务器	Server\Bytes Total/sec		如果所有服务器的 Bytes Total/sec 和与网络的最大传送速度大致相等，则可能需要将网络分段。
服务器	Server\Work Item Shortages	3	如果值达到该阀值，请考虑将 DWORD 项“InitWorkItems”（在启动期间分配给处理器的工作项

			数）或者 MaxWorkItems（服务器可以分配的接收缓冲区的最大数）添加到注册表（在 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters 下面）。InitWorkItems 的范围可以是从 1 到 512，同时 MaxWorkItems 的范围可以是从 1 到 65535。以 InitWorkItems 的任何值以及 MaxWorkItems 的值 4096 开始，并一直加倍这些值直到 Server\Work Item Shortages 阀值低于
服务器	服务器\分页池峰值	物理 RAM 的数量	此值是最大页面文件大小和物理内存数量的指示器。
服务器	Server Work Queues\Queue Length	4	如果值到达此阈值，则可能存在处理器瓶颈。这是即时计数器；观察在多个间隔上的值。
多个处理器	System\Processor Queue Length	2	这是即时计数器；观察在多个间隔上的值。

第 4 节 Win-Processor

指标名称	指标描述	指标范围
CPU 利用率（% Processor Time）	% Processor Time 指处理器执行非闲置线程时间的百分比。这个计数器设计成用来作为处理器活动的主要指示器。它通过在每个时间间隔中衡量处理器用于执行闲置处理线程的时间，并且用 100%减去该值得出。可将其视为范例间隔用于做有用工作的百分比。	根据应用系统情况，在 80%±5%范围内波动为宜。过低，则服务器 CPU 利用率不高；过高，则 CPU 可能成为系统的处理瓶颈。
中断率 Interrupts/sec.	每秒钟设备中断处理器的次数。在完成一个任务或需要注意时，装置会发出中断讯号给处理器。可以产生中断的装置包括系统定时器、鼠标、数据通讯联机、网络卡以及其它的外部装置。在中断过程中，一般的执行绪执行将被暂停，而且一个中断可以使处理器切换到另一个具有较高优先等级的执行绪。频率中断是频繁和周期性的，并且中断动作在背景执行。	取决于处理器，越低越好；不宜超过 1,000； 如果该值显著增加而系统活动没有相应的增加，则表明存在硬件问题，需要检查引起中断的网络适配器、磁盘或其他硬件。
系统调用率 System Call/sec.	指运行在计算机上的所有处理器调用操作系统服务例行程序的综合速率。这些例行程序执行所有在计算机上的如安排和同步活动等基本的程序，并提供对非图形	如果 Interrupts/sec 大于 System Calls/sec.，则系统中某一硬件设备产生过多的中断。

	设备、内存管理和名称空间管理的访问。	
Processor Queue Length	处理器队列的线程数量。此计数器只显示就绪线程，而不是正在运行的线程。	如果处理器队列中总是有两个以上的线程通常表示处理器堵塞。
进程切换率 Context Switches/sec	指计算机上的所有处理器全都从一个线程转换到另一个线程的综合速率。当正在运行的线程自动放弃处理器时出现上下文转换，由一个有更高优先就绪的线程占先或在用户模式和特权 (内核) 模式之间转换以使用执行或分系统服务	如果此计数器的数值较大，则表明锁定竞争很激烈，或者线程在用户和内核模式之间频繁切换。

第 5 节 Win-Memory

指标名称	指标描述	指标范围
Pages/sec Pages Input/sec Pages Output/sec Page Fault/sec	Page Faults/sec 是处理器每秒钟处理的错误页（包括软错误和硬错误）。Pages Input/sec 是为了解决硬错误页，从硬盘上读取的页数， 而 Page Reads/sec 是为了解决硬错误，从硬盘读取的次数。Pages/sec 是 Pages Input/sec 和 Pages Output/sec 的总和。 该系列指标是可以显示导致系统范围延缓类型错误的主要指示器。 当处理器向内存指定的位置请求一页（可能是数据或代码）出现错误时，这就构成一个 Page Fault。如果该页在内存的其他位置，该错误被称为软错误（ 用 Transition Fault/sec 衡量）； 如果该页必须从硬盘上重新读取时， 被称为硬错误。许多处理器可以在有大软错误的情况下继续操作。但是， 硬错误可以导致明显的拖延。	如果 Page Reads/Sec 持续保持为 5，表示可能内存不足。Page/sec 推荐 0-20。如果服务器没有足够的内存处理其工作负荷，此数值将一直很高。如果大于 80，表示有问题(太多的读写数据操作要访问磁盘，可考虑增加内存或优化读写数据的算法)。 该系列计数器的值比较低， 说明响应请求比较快， 否则可能是服务器系统内存短缺引起（也可能是缓存太大， 导致系统内存太少）。
Available Bytes	显示出当前空闲的物理内存总量，它等于分配给待机(缓存的)、空闲和零分页列表内存的总和。 空闲内存可以马上使用; 清零内存是由零值填满的内存页，用来防止后续进程获得旧进程使用的数据; 待机内存是从进程工作集(其物理内存)中删除然后进入磁盘的内存，但是该内存仍然可以收回。该指标仅显示最后一次观察到的值，不是平均值。	当这个数值变小时，Win 开始频繁地调用磁盘页面文件。如果这个数值很小，例如小于 5 MB，系统会将大部分时间消耗在操作页面文件上。 一般要保留 10%的可用内存。最低不能<4M，此值过小可能是内存不足或内存泄漏。

Committed Bytes	是指以字节表示的确认虚拟内存，是磁盘页面文件上保留空间的物理内存。	不超过物理内存的 75%

第 6 节 Win-Disk

指标名称	指标描述	指标范围
% Disk Time	指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间的百分比。	正常值<10，此值过大表示耗费太多时间来访问磁盘，可考虑增加内存、更换更快的硬盘、优化读写数据的算法。若数值持续超过 80 (此时处理器及网络连接并没有饱和)，则可能是内存泄漏。
CurrentDiskQueueLength	是在收集性能数据时磁盘上当前的请求数量。它还包括在收集时处于服务的请求。这是瞬间的快照，不是时间间隔的平均值。多轴磁盘设备能有一次处于运行状态的多重请求，但是其他同期请求正在等待服务。此计数器会反映暂时的高或低的队列长度，但是如果磁盘驱动器被迫持续运行，它有可能一直处于高的状态。	请求的延迟与此队列的长度减去磁盘的轴数成正比。为了提高性能，此差应该平均小于二。
DiskQueueLength DiskReadQueueLength DiskWriteQueueLength	指读取和写入请求(为所选磁盘在实例间隔中列队的)的平均数。	Avg.Disk Queue Length 正常值<0.5，此值过大表示磁盘 IO 太慢，要更换更快的硬盘。

第 8 章 Web server

第 1 节 应用级优化

系统优化一般主要分为应用级和代码级两种优化方式。应用级优化比较简单，主要是修改一些服务器参数，如 Apache 服务器，主要是对 Httpd.conf、Access.conf、Srm.conf 三个配置文件中的参数进行配置。

◆ **服务器配置优化**

运用更快速的磁盘、更好的网络存取机制、更快处理速度的 CPU、更大的内存，可以改进网站访问速度。

◆ **缓存机制（Cache）**

将一些静态页面或者有规律变化的动态页面放入读取速度更快的 cache 中，使用户访问不用每次读取数据库或文件系统。可以明显提高访问速度。

注意各个服务都会做自己的 cache；每个 cache 可能分为几级；可以做文件 cache、内存 cache 等等；设置 cache 时要特别注意，哪些文件需要 cache，cache 的大小如何设计。如何实现 cache 间负载均衡等等问题。

◆ **减小数据传输**

请求的响应速度，很大程度上是受网络传输限制的。在网络状况确定的情况下，减小网络传输数据量是提高响应速度的很好的方式。如何减小网络传输？使用压缩。

对传输数据进行压缩，能提高访问速度。对于 web 页面传输来说，传输量较大的几个类型是：图片、html 页面文件、js 等脚本。HTTP1.1 的浏览器能够同时支持 GZIP 文件编码，在传输前对文件进行压缩。Web 服务器如 Apache、Microsoft IIS5.0 也支持这种方案。

一般理解认为，数据量小即意味着网站传输速度快，然而具体到某个传输对象，必须考虑压缩/解压的时间因素，高度压缩文件用到带宽可能较少，但却不一定能快速提交到终端用户。

◆ **网络布局以用户为中心**

通信传输是网络要解决的最重要课题，任何网络厂商都得面对通信和路由方面的难题，要使一个网站能够“贴近”用户，就得在通信传输方案方面下功夫。可以在访问区间设置多服务器，并运用全局负载均衡设备，使用户实现对网站内容的就近访问。

具体该如何设置服务器格局，需要作为一个单独的课题去研究，这里不做介绍。

◆ **其他方式**

将高访问量的网页静态化，以避免这些高访问量对数据库进行大量的调用；

通过负载平衡的方法来水平缩放 Web 服务器的结构；

做数据库群并将它们分为读写服务器和只读服务器，对只读服务器群负载平衡；

通过增加更多的硬件资源（CPU，内存，磁盘等）纵向的缩放 Web 服务器；

增加网络的带宽；

第 2 节 服务瓶颈分析

如要对服务器的性能进行优化，首先要了解服务器的用途和影响这类服务器性能的主要因素。一旦因素被确定，就可以有效地解决性能问题。下边讨论通用服务器类型和各类服务

器性能的瓶颈。

8.2.1 Domain Controller

域控制服务器提供认证服务，实现对网络资源进行管理，包括用户、设备、计算机。维护并实施安全策略，提供一个可靠的网络环境。具有以下的功能：

1. 用户认证；2. 资源访问认证；3. 安全控制；

域控制器影响其性能的硬件系统依次为：

1 .Memory；2 .Network；3 .Processor；

8.2.2 File Server

文件服务器用来根据客户端的请求保存、查找和更新数据。因此，影响性能的主要因素是数据传输和网络系统的速度。可用内存的数量直接影响网络缓冲区和 I/O 缓存的访问速度。处理器速度对文件服务器的性能影响不大。对于大规模网络环境，还要考虑文件服务器的位置。尽量放在靠近核心交换机的高速主干网。

文件服务器影响其性能的硬件系统依次为：

1. Network；2. Memory；3. Disk；

8.2.3 Database Server

数据库服务器主要提供数据存储、查询、找回和升级。主要有 IBM DB2, Microsoft SQL Server 和 Oracle. 由于数据库服务器要处理大量频繁的随机 I/O 请求，并进行精密计算。

影响其性能的硬件系统依次为：

1. Memory；2. Disk；3. Processor；4. Network；

8.2.4 E-mail servers

邮件服务器提供电子邮件的保存和路由，并将邮件转发到指定地址. 做为邮件服务器要定期进行目录复制、邮件同步并和第三方服务器通信而产生大量的网络流量. 同时还要存储和管理邮件，磁盘子系统也变的非常重要。

邮件服务器影响其性能的硬件系统依次为：

1. Memory；2. CPU；3. Disk；4. Network；

8.2.5 Web Server

Web 服务器用来提供 Web 页面浏览和运行 server-intensive Web 请求.

如果 Web Site 内容是静态的，影响其性能的硬件系统依次为：

1. Network；2. Memory；3. CPU；

如果 Web server 是计算密集型（比如动态生成页面），影响其性能的硬件系统依次为:

1. Memory；2. Network；3. CPU；4. Disk；

第 3 节 通用指标

◆ **Web 服务器通用指标如下：**

1. Processor \ Processon time \ Tatol cpu 时间
2. Memory \ Availalle MbyteAvai 应用服务器的内存
3. Requst Quened 进入 HTTP 队列的时间；队列/每秒
4. Total request 总请求数时间
5. Avg Rps 平均每秒钟响应次数＝ 总请求时间 / 秒数
6. Avg time to last byte per terstion （mstes）平均每秒迭代次数 ；
7. Http Error 无效请求次数
8. Send 发送请求次数字节数
9. Successful Rounds：成功的请求；
10. Failed Rounds ：失败的请求；
11. Successful Hits ：成功的点击次数；
12. Failed Hits ：失败的点击次数；
13. Hits Per Second ：每秒点击次数；
14. Successful Hits Per Second ：每秒成功的点击次数；
15. Failed Hits Per Second ：每秒失败的点击次数
16. Attempted Connections ：尝试链接数；

第 4 节 WebLogic

8.4.1 影响服务器性能的参数

指标名称	指标描述
Server	mydomain->Servers->myserver->Configuration->Tuning-> Enable Native IO
Native IOEnabled	将此值设为 True（默认为 True）。config.xml 文件中<server NativeIOEnabled=“true”>表示该 Server 使用本地 I/O。如果选中，但本地操作系统没有可用的性能包，则 WLS 会自动使用非本地 I/O。
Server	server—>configuration->tuning
SocketReaders	设置在执行线程中专用做 Socket Readers 的百分比。 如果必须使用纯 java 的 socket reader，可为每个 Server 实例和客户机配置一定数量 socket reader 线程。可设置执行线程中一部分专用于读取 Socket。ThreadPoolPercentSocketReaders 属性用于设置在执行线程中专用做 Socket Readers 的百分比。因此对各种应用此值不同。 在 WebLogic Server 中设置方法：console： mydomain->Servers->myserver->Configuration->Tuning->Socket Readers 在客户机中设置方法：-Dweblogic.ThreadPoolSize=value -Dweblogic.ThreadPoolPercentSocketReaders=value
Maximum Open Sockets	最大打开 Socket 数。

Stuck Thread MaxTime	堵塞线程时间，超过这个时间没有返回的执行线程，系统将认为是堵塞线程。如果 weblogic 认为某个队列中的所有的线程全部堵塞的话，weblogic 将会增加执行线程的数量。 注意：执行线程的数量一旦增加，目前 weblogic 不会去减少他，如果增加了一些线程以后再次出现 overflow 的警告，weblogic 会继续增加执行线程的数量，一直到达到上限为止。 当执行队列中的线程停滞时会被 weblogic 自动检测。server 会根据当前停滞线程的数目来诊断当前系统的健康状况。 如果默认队列中所有线程都停滞，服务器健康状况变为“危急”。 如果 weblogic.admin.HTTP, weblogic.admin.RMI 或用户定义执行队列中线程都处于停滞，服务器状况变为“警告”。 配置线程检测条件（这是以 server 为单位的配置，而非以执行队列为单位）： /console/中 mydomain->Servers->myserver->Configuration->Tuning->Stuck Thread Max Time 线程连续工作多长时间，会被认为该线程被阻塞住。 /console/中：mydomain->Servers->myserver->Configuration->Tuning->Stuck Thread Timer Interval weblogic server 周期性扫描所有执行队列线程的时间间隔。
Stuck Thread Timer Interval	系统检查堵塞线程的时间间隔、阻塞线程计时器间隔：WebLogic Server 定期扫描线程以查看它们是否已经连续工作了 "阻塞线程最长时间" 字段中指定的时间长度的间隔时间(秒)。默认情况下，WebLogic Server 将此时间间隔设置为 600 秒。
Low Memory GC Threshold	当可用内存小于该百分比时，垃圾回收启动。
Low Memory Granularity Level	当两次检测的可用内存变化超过该百分比时，垃圾回收启动。
Low Memory Sample Size	在一次检测中的取样次数。
Low Memory Time Interval	检测间隔时间。
Accept Backlog	等待队列中最多可以有多少 TCP 连接等待处理，如果在许多客户端连接被拒绝，而在服务器端没有错误显示，说明该值设得过低。如果连接时收到 connection refused 消息，说明应提高该值，每次增加 25％ config.xml 文件中 server 元素的 AcceptBacklog 属性设定 weblogic server 实例能接受的最大连接数。 AcceptBacklog 决定了在等待队列中最多可以有多少 TCP 连接等待处理。设置方法：/console/中 mydomain->Servers->myserver->Configuration->Tuning->Accept Backlog 如果在许多客户端连接被拒绝，而在服务器端没有错误显示，说明该值设得过低。 如果连接时收到 connection refused 消息，说明应提高该值，每次增加 25％。
ExecuteQueue	console ： mydomain->Servers->myserver ->Monitoring->Monitor all Active Queues... ->Configuration->weblogic.kernel.Default->

ThreadCount	服务器初始创建的执行线程的数量，设置原则： 增大机器的最大并发线程数使处理器利用率达到最大。对于服务器端操作比较多的线程，应该减少线程计数；对于客户端操作比较多的，应该增加线程计数。并发线程数理论上等于“本地主机 CPU 个数+Stuck 线程数”，够用即可，过大会降低系统性能。 每个新的 server 实例默认有一个队列 weblogic.kernel.default 用于存储对 Web 应用及 RMI 对象的请求。该队列默认被分配了 15 个线程。 增大机器的最大并发线程数使处理器利用率达到最大。对于服务器端操作比较多的线程，应该减少线程计数；对于客户端操作比较多的，应该增加线程计数。并发线程数理论上等于“本地主机 CPU 个数+Stuck 线程数”，够用即可，过大会降低系统性能。 console： mydomain->Servers->myserver->Monitoring->General->Monitor all Active Queues... ->weblogic.kernel.Default console： mydomain->Servers->myserver ->Monitoring->Monitor all Active Queues... ->Configuration->default config.xml：<Server…>…<ExecuteQueue ThreadCount=“50”/></Server>
Queue Length	表示执行队列中可容纳的最大请求数，默认值是 65536。 在等待队列里的请求数，理想状态下是 0
QueueLength ThresholdPercent	此值表示溢出条件，在此服务器指出队列溢出之前可以达到的队列长度大小的百分比。 一个百分数，当 request 的数量达到队列长度的这个比例的时候，weblogic 会发出 overflow 的标志信息。
Threads Increase	如果 weblogic 发出 overflow 的标志信息，weblogic 会尝试增加这个数量的执行线程，以解决处理矛盾。 当检测到溢出条件时，将增加到执行队列中的线程数量。如果 CPU 和内存不是足够的高，尽量不要改变默认值“0”。因为 Weblogic 一旦增加后不会自动缩减，虽然最终可能确实起到了降低请求的作用，但在将来的运行中将影响程序的性能。
Maximum	最大执行线程数。
Maximum	最小执行线程数。为了防止创建过多的线程数量，可以通过设定最大的线程数进行控制。
Thread Priority	线程优先级：表示线程队列中线程默认使用的级别。
JDBCConnectionPool	mydomain-> JDBC Connection Pools->Configuration->Connections
Initial Capacity	初始数据库物理连接数。 表示 WebLogic Server 启动后默认在连接池中初始化多少个连接。
Max Capacity	最大数据库物理连接数。 受制于 WebLogic Server 线程数的设置和数据库进程数,游标的大小。通常我们在一个线程中使用一个连接,所以连接数并不是越多越好,为避免两边的资源消耗，建议设置连接池的最大值等于或者略小于线程数。同时为了减少新建连接的开销,将最小值和最大值设为一致。
Capacity Increment	每次数据库物理连接增加数

Statement Cache Size	宏语句设定的静态缓存，大小由 JDBC 连接池配置时指定，调整这个数值的大小，有利于提高系统的效率。 每个连接都为宏语句设一个静态的缓存，大小由 JDBC 连接池配置时指定。缓存是静态的，时刻牢记这一点非常重要。这意味着如果缓存的大小是 n 的话，则只有放在缓存中的前 n 条语句得到执行。确保昂贵的 SQL 语句享受到缓存的方法是用一个启动类将这些语句存放到缓存中。尽管缓存技术从很大程度上改进了性能，但也不能盲目使用它。如果数据库的格式有了变化，那么在不重新启动服务器的情况下，无法使缓存中的语句失效或者是用新的进行替换。当然，缓存中的语句会使数据库中的光标得以保留。
Statement Cache Type	prepared statements 缓存的策略，LRU 算法在有新的语句到来时，将最不经常被用得语句调整出缓存。FIXED 算法为先进先出的算法
TestConnectionsOnReserve	TestConnectionsOnReserve 设置为 false（缺省设置）。如果此参数设置为真（true），则在连接被分配给调用者之前，都要经过测试，这会额外要求与数据库的反复连接
Statement Cache Size	宏语句设定的静态缓存，大小由 JDBC 连接池配置时指定，调整这个数值的大小，有利于提高系统的效率
Login Delay	创建数据库物理连接时的延时时间

8.4.2 运行时模式

1．为更改运行在一个 webLogic 主机上的所有域的运行时模式，用文本编辑器打开 WL_HOME\common\bin\commEnv.cmd(Windows) 或者 WL_HOME\common\bin\commEnv.sh (UNIX)，WL_HOME 是安装 webLogic 的路径。

2. 为指定的域更改运行时模式，就用文本编辑器打开 domain-name\StartwebLogic.cmd (Windows) or domain-name\StartwebLogic.sh (UNIX)，domain-name 为创建的域的目录。

3．在这个脚本中，更改 PRODUCTION_MODE 的值，如果你要服务器运行在产品模式，指定其值为 TRUE。

4.选择 JRockit 作为虚拟机.

5.weblogic-ejb-jar.xml 影响性能参数。

元素	解释
max-beans-in-free-pool	为 Session 和 Message-Driven Beans 设置 EJB 池大小
initial-beans-in-free-pool	为 Stateless Sessions Beans 的起始状态调整池大小
max-beans-in-cache	为 Stateful Session 和 Entity Beans 设置缓冲大小
concurrency-strategy	定义数据库锁
isolation-level	设置事务隔离级别
relationship-caching	Entity Beans 的关联缓冲支持

第 5 节 WebSphere

指标名称	指标描述
JVM	**Java 虚拟机设置：**要查看此管理控制台页面，单击服务器 > 应用程序服务器 > server_name > 流程定义 > Java 虚拟机。“配置”选项卡。 **初始的堆大小：**JVM 代码可用的最大堆大小（以兆字节计）。 增加最小堆大小可以改进启动。垃圾收集发生数减少，在性能上实现了 10% 的增益。通常，增加 Java 堆大小可改进吞吐量，直到堆不在驻留在物理内存。当堆开始交换到磁盘后，Java 性能强烈受到危害。数据类型 整型 缺省值 64 用于 OS/400，50 用于所有其它平台。 **堆大小的最大值：**指定 JVM 可用的最大堆大小（以兆字节计）。 增加堆大小可改进启动。垃圾收集发生数减少，在性能上实现了 10% 的增益。通常，增加 Java 堆大小可改进吞吐量，直到堆不在驻留在物理内存。当堆开始交换到磁盘后，Java 性能强烈受到危害。因此最大堆大小需要设置得足够低，以便将堆包含在物理内存中。数据类型 整型 缺省值 0（OS/400），256（其它所有平台）。将值保持足够低，避免页面调度或内存交换到磁盘。 **几种启动模式的配置：** **-Xquickstart：**您可将此值用于较低优化级别的初始编译，而不是缺省方式，稍后，根据采样结果，您可以缺省方式重新编译初始编译级别。对早期适中速度比长时间运行吞吐量更重要的应用程序使用 quickstart。在一些调试方案中，测试装备和短运行工具，可能将启动时间增益 15-20%。-DCOPT_NQREACHDEF 可以将启动时间改进额外的 15%。 **-Xverify:none：**当使用此值时，类装入期间跳过类验证阶段。通过使用已启用 Just In Time（JIT）编译器的 -Xverify:none，启动时间改进了 10-15%。 **-Xnoclassgc：**您可使用此值禁用类垃圾收集，使类重用更可用，而且略微改进性能。缺省情况下，类垃圾收集是启用的，但是建议您启用它。您可以使用 verbose:gc 配置设置监控垃圾收集，因为其输出包含类垃圾收集统计信息。 **-Xmc:** 线程本地堆大小是专门为线程分配的堆的一部分。由于线程本地堆大小，线程在分配对象时，不需要锁定整个堆。然而，当线程本地堆满时，对象分配从需要同步的堆完成。良好的本地高速缓存大小关键在于良好的性能. **-Xml：**您可使用此值设置从本地高速缓存分配的对象大小的限制。超出限制大小的对象需要在常规堆中分配。尽可能从本地高速缓存分配对象，或本地高速缓存耗尽，因为它不会动态增长。如果您了解一些对象将变得很大，则从常规堆分配它们。 数据类型 字符串 单位 Java 命令行变量 **禁用 JIT：**指定是否禁用 JVM 代码的 Just in Time（JIT）编译器选项。 如果您禁用 JIT 编译器，吞吐量明显减少。因此，处于性能原因，保持 JIT 启用。
connection pool size, statement cache size	**Minimum Pool Size：**池中保持的连接的最小数目；有新的请求，且没有激活连接可供用时，池中连接数将增大，到最大连接数为止 **Maximum Pool Size：**池中保持的连接的最大数目；当这个数目达到，且没有激活连接供使用时，新的请求将等待 Connection Timeout 当连接数达到最大值，且激活连接都在被使用时，新的请求等待时间 **Idle Timeout ：**连接可在池中闲置的时间；超过将释放资源，到最小连接数为止 。 **Orphan Timeout ：**连接在被应用控制时，可闲置的时间；超过将返回池中 你可以根据需要来修改这些数值，以满足你的应用需要。

Thread Pool	**最小大小：**指定池中允许的最小线程数。缺省值 10 **最大大小：**指定池中允许的最大线程数。 如果 Tivoli 性能查看器显示最大百分比度量以始终保持在双精度数字，考虑增加最大大小。最大百分比度量表明使用已配置线程的时间数。如果存在多个并发客户机连接到服务器端 ORB，增加大小以最多支持 1000 个客户机。缺省值 50 建议 50（Linux 系统上 25 个） 。 **线程不活动超时：**指定在收回线程之前应该消逝的不活动的毫秒数。为 0 的值表明不等待而负值（小于 0）意味着永远等待。单位 毫秒 缺省值 3500 。 **可增长的线程池：**指定线程数是否能增加至超过为线程池配置的最大大小。数据类型 布尔，缺省值 未启用（false） 范围 有效值是允许线程分配超过最大线程大小或“未启用”。

第 9 章 Data base

常见的数据库有：MS SQL Server、Sybase、Oracle、MySQL、DB2、Informix 等，本章重点介绍关于 SQL Server、Oracle、DB2 的一些性能指标。

9.1.1 SQL Server 计数器

◆ **常用调优指标：**

- SQLServer 资源监控中指标缓存点击率（Cache Hit Ratio），该值越高越好。如果持续低于 80%，应考虑增加内存。
- 如果 Full Scans/sec（全表扫描/秒）计数器显示的值比 1 或 2 高，则应分析你的查询以确定是否确实需要全表扫描，以及 SQL 查询是否可以被优化。
- Number of Deadlocks/sec(死锁的数量/秒)：死锁对应用程序的可伸缩性非常有害，并且会导致恶劣的用户体验。该计数器的值必须为 0。
- Lock Requests/sec(锁请求/秒)，通过优化查询来减少读取次数，可以减少该计数器的值。
- User Connections (用户连接数，也就是数据库的连接数量);
- Number of deadlocks/Sec/－Total （数据库死锁）
- Memory\ Availalle Mbyte 内存监控 (可用内存)
- Physicsdisk \disk time \－Total（磁盘读写总时间）（出现瓶颈时检查读磁盘的时间长还是写磁盘的时间长）
- Butter Caile hit(数据库缓存的选取命中率)
- 数据库的命中率不能低于 92％

◆ **详细指标：**

注：以下指标取自 SQL Server 自身提供的性能计数器。

指标名称	指标描述	指标范围
1．SQL Server 中访问方法（Access Methods）对象包含的性能计数器		
全表扫描/秒 (Full Scans/sec)	指每秒全表扫描的数量。全表扫描可以是基本表扫描或全索引扫描。由于全表扫描需要耗费大量时间，因此全表扫描的频率过高的话，会影响性能。	如果该指标的值比 1 或 2 高，应该分析设计的查询以确定是否确实需要全表扫描，以及 SQL 查询是否可以被优化。
2．SQL Server 中缓冲器管理器（Buffer Manager）对象包含的性能计数器		
缓冲区高速缓存命中率 (Buffer Cache Hit Ratio %)	指在缓冲区高速缓存中找到而不需要从磁盘中读取的页的百分比。该比率是缓存命中总次数与缓存查找总次数之比。经过很长时间后，该比率的变化很小。由于从缓存中读取数据比从磁盘中读取数据的开销小得多，一般希望该比率高一些。	该指标的值最好为 90% 或更高。通常可以通过增加 SQL Server 可用的内存数量来提高该指标的值。增加内存直到这指标的值持续高于 90%，表示 90% 以上的数据请求可以从数据缓冲区中获得所需数据。
读的页/秒 (Page Reads/sec)	指每秒发出的物理数据库页读取数。该指标主要考察数据库从磁盘读取数据的频率。因为物理 I/O 会耗费大量时间，所	该指标的值应尽可能的小。可以通过使用更大的数据高速缓存、智能索引、更高效的查询或者改变数据

	以应尽可能地减少物理 I/O 以提高性能。	库设计等方法，以降低该指标的值。
写的页/秒 (Page Writes/sec)	指每秒执行的物理数据库写的页数。该指标主要考察数据库向磁盘写入数据的频率。因为物理 I/O 会耗费大量时间，所以应尽可能地减少物理 I/O 以提高性能。	该指标的值应尽可能的小。可以通过使用更大的数据高速缓存、智能索引、更高效的查询或者改变数据库设计等方法，以降低该指标的值。
惰性写/秒 (Lazy Writes/sec)	指每秒被缓冲区管理器的惰性编写器写入的缓冲区数。惰性编写器是一个系统进程，用于成批刷新脏的老化的缓冲区（包含更改的缓冲区，必须将这些更改写回磁盘，才能将缓冲区重用于其他页），并使它们可用于用户进程。	该指标的值最好为 0。
3．SQL Server 中高速缓存管理器（Cache Manager）对象包含的性能计数器		
高速缓存命中率 (Cache Hit Ratio %)	指高速缓存命中次数和查找次数的比率。在 SQL Server 中，Cache 包括 Log Cache, Buffer Cache 以及 Procedure Cache，该指标是指所有 Cache 的命中率，是一个总体的比率。	该指标的值越高越好。如果该指标的值持续低于 80%，就需要增加更多的内存。
4．SQL Server 中闩（Latches）对象包含的性能计数器		
平均闩等待时间(毫秒) (Average Latch Wait Time(ms))	指一个 SQL Server 线程必须等待一个闩的平均时间。	如果该指标的值很高，则系统可能正经历严重的资源竞争问题。
闩等待/秒 (Latch Waits/sec)	指在一个闩上每秒的平均等待数量。	如果该指标的值很高，则系统可能正经历严重的资源竞争问题。
5．SQL Server 中锁（Locks）对象包含的性能计数器		
死锁的数量/秒 (Number of Deadlocks/sec)	指每秒导致死锁的锁请求数。	锁加在 SQL Server 资源上（如在一个事务中进行的行读取或修改），以防止多个事务并发使用资源。应尽可能少使用锁以提高事务的并发性，从而改善性能。
平均等待时间(毫秒) (Average Wait Time(ms))	指线程等待某种类型的锁的平均等待时间。	同上
锁请求/秒 (Lock Requests/sec)	指每秒钟某种类型的锁请求的数量。	同上

9.1.2 Oracle 计数器

◆ **常用调优指标：**

- 如果自由内存接近于 0 而且库快存或数据字典快存的命中率小于 0.90，那么需要增加

SHARED_POOL_SIZE 的大小。

- 快存（共享 SQL 区）和数据字典快存的命中率：
  select(sum(pins-reloads))/sum(pins) from v$librarycache;
  select(sum(gets-getmisses))/sum(gets) from v$rowcache;
- 自由内存： select * from v$sgastat where name='free memory';
- 如果数据的缓存命中率小于 0.90，那么需要加大 DB_BLOCK_BUFFERS 参数的值（单位：块）。
- 缓冲区高速缓存命中率：
  select name,value from v$sysstat where name in ('db block gets',
  'consistent gets','physical reads') ;
  Hit Ratio = 1-(physical reads / (db block gets + consistent gets))
- 如果日志缓冲区申请的值较大，则应加大 LOG_BUFFER 参数的值。
- 日志缓冲区的申请情况：
  select name,value from v$sysstat where name = 'redo log space requests' ;
- 如果内存排序命中率小于 0.95，则应加大 SORT_AREA_SIZE 以避免磁盘排序 。
- 内存排序命中率：
  select round((100*b.value)/decode((a.value+b.value), 0, 1, (a.value+b.value)), 2)from v$sysstat a, v$sysstat b where a.name='sorts (disk)' and b.name='sorts (memory)'

◆ **详细指标：**

注：以下指标取自 Oracle 的性能分析工具 Statspack 所提供的性能分析指标。

指标名称	指标描述	指标范围
1．关于实例效率（Instance Efficiency Percentages）的性能指标		
缓冲区未等待率 (Buffer Nowait %)	指在缓冲区中获取 Buffer 的未等待比率。	该指标的值应接近 100%，如果该值较低，则可能要增大 buffer cache。
Redo 缓冲区未等待率 (Redo NoWait %)	指在 Redo 缓冲区获取 Buffer 的未等待比率。	该指标的值应接近 100%，如果该值较低，则有 2 种可能的情况： online redo log 没有足够的空间； 2）log 切换速度较慢。
缓冲区命中率 (Buffer Hit %)	指数据块在数据缓冲区中的命中率。	该指标的值通常应在 90%以上，否则，需要调整。如果持续小于 90%，可能要加大 db_cache_size。但有时，缓存命中率低并不意味着 cache 设置小了，可能是潜在的全表扫描降低了缓存命中率。
内存排序率 (In-memory Sort %)	指排序操作在内存中进行的比率。当查询需要排序的时候，数据库会话首先选择在内存中进行排序，当内存大小不足的时候，将使用临时表空间进行磁盘排序，但磁盘排序效率和内存排序效率相差好几个数量级。	该指标的值应接近 100%，如果指标的值较低，则表示出现了大量排序时的磁盘 I/O 操作，可考虑加大 sort_area_size 参数的值。
共享区命中率	该指标主要代表 sql 在共享区的	该指标的值通常应在 95%以上，否则

(Library Hit %)	命中率。	需要考虑加大共享池（修改 shared_pool_size 参数值），绑定变量，修改 cursor_sharing 等参数。
软解析的百分比 (Soft Parse %)	该指标是指 Oracle 对 sql 的解析过程中，软解析所占的百分比。软解析（soft parse）是指当 Oracle 接到 Client 提交的 Sql 后会首先在共享池（Shared Pool）里面去查找是否有之前已经解析好的与刚接到的这一个 Sql 完全相同的 Sql。当发现有相同的 Sql 就直接用之前解析好的结果，这就节约了解析时间以及解析时候消耗的 CPU 资源。	该指标的值通常应在 95%以上，如果低于 80%，那么就可能 sql 基本没被重用，sql 没有绑定变量，需要考虑绑定变量。
闩命中率 (Latch Hit %)	指获得 Latch 的次数与请求 Latch 的次数的比率。	该指标的值应接近 100%，如果低于 99%，可以考虑采取一定的方法来降低对 Latch 的争用。
SQL 语句执行与解析的比率 (Execute to Parse %)	指 SQL 语句执行与解析的比率。SQL 语句一次解析后执行的次数越多，该比率越高，说明 SQL 语句的重用性很好。	该指标的值应尽可能到高，如果过低，可以考虑设置 session_cached_cursors 参数。
共享池内存使用率 (Memory Usage %)	该指标是指在采集点时刻，共享池（share pool）内存被使用的比例。	这指标的值应保持在 75%~90%,如果这个值太低，就浪费内存，如果太高，会使共享池外部的组件老化，如果 SQL 语句被再次执行，则就会发生硬分析。
2．关于等待事件（Wait events）的性能指标		
文件分散读取 (db file scattered read (cs))	该等待事件通常与全表扫描有关。因为全表扫描是被放入内存中进行的进行的，通常情况下它不可能被放入连续的缓冲区中，所以就散布在缓冲区的缓存中。	如果这个等待事件比较显著，可能说明对于某些全表扫描的表，没有创建索引或没有创建合适的索引。尽管在特定条件下执行全表扫描可能比索引扫描更有效，但如果出现这种等待时，最好检查一下这些全表扫描是否必要。
文件顺序读取 (db file sequential read (cs))	该等待事件通常与单个数据块相关的读取操作有关。	如果这个等待事件比较显著，可能表示在多表连接中，表的连接顺序存在问题，或者可能不合适地使用了索引。对于大量事务处理、调整良好的系统，这一数值大多是很正常的，但在某些情况下，它可能暗示着系统中存在问题。应检查索引扫描，以保证每个扫描都是必要的，并检查多表连接的连接顺序。另外 DB_CACHE_SIZE 也

		是这些等待出现频率的决定因素。
缓冲区忙 (buffer busy (cs))	当一个会话想要访问缓存中的某个块，而这个块正在被其它会话使用时，将会产生该等待事件。这时候，其它会话可能正在从数据文件向缓存中的这个块写入信息，或正在对这个块进行修改。	出现这个等待事件的频度不应大于1%。如果这个等待事件比较显著，则需要根据等待事件发生在缓存中的哪一块（如字段头部、回退段头部块、回退段非头部块、数据块、索引块等），采取相应的优化方法。
(enqueue (cs))	enqueue 是一种保护共享资源的锁定机制。该锁定机制保护共享资源，如记录中的数据，以避免两个人在同一时间更新同一数据。enqueue 包括一个排队机制，即 FIFO(先进先出)排队机制。注意：Oracle 的 latch 机制不是 FIFO。Enqueue 等待通常指的是 ST enqueue、HW enqueue、TX4 enqueue 和 TM enqueue。	如果 enqueue 等待事件比较显著，则需要根据 enqueue 等待类型，采取相应的优化方法。
闩释放 (latch free (cs))	该等待事件意味着进程正在等待其他进程已持有的 latch。 latch 是一种低级排队机制(它们被准确地称为相互排斥机制)，用于保护系统全局区域(SGA)中共享内存结构。latch 就像是一种快速地被获取和释放的内存锁。latch 用于防止共享内存结构被多个用户同时访问。	对于常见的 Latch 等待通常的解决方法： 1）Share pool latch：在 OLTP 应用中应该更多的使用绑定变量以减少该 latch 的等待。 2）Library cache latch：同样的需要通过优化 sql 语句使用绑定变量减少该 latch 的等待。
日志文件同步 (log file sync (cs))	这个等待事件是指当一个会话完成一个事务(提交或者回滚数据）时，必须等待 LGWR 进程将会话的 redo 信息从日志缓冲区写到日志文件后，才能继续执行下去。	这个等待事件的时间过长，可能是因为 commit 太频繁或者 lgwr 进程一次写日志的时间太长（可能是因为一次 log io size 太大），可调整_log_io_size，结合 log_buffer,使得(_log_io_size*db_block_size)*n = log_buffer，这样可避免和增大 log_buffer 引起冲突，或者可以将日志文件存放在高速磁盘上

◆ **Oracle 常用计数器：**

编号	监控名称	描述
1	Logons current	当前的登录总数
2	Opened cursors current	当前打开的光标总数
3	User calls	在每次登录、解析或执行时，Oracle 会分配资源（Call State 对象）以记录相关的用户调用数据结构。在确定

		活动时，用户调用与 RPI 调用的比说明了因用户发往 Oracle 的请求类型而生成的内部工作量。
4	Opers of replaced files	由于已经不在进程文件缓存中，所以需要重新打开的文件总数

◆ **Oracle 常用自定义计数器：**

1	数据高速缓存区命中率	SELECT round(1-SUM(PHYSICAL_READS)/(SUM(DB_BLOCK_GETS) + SUM(CONSISTENT_GETS)), 4) * 100 FROM (SELECT CASE WHEN NAME='physical reads' THEN VALUE END PHYSICAL_READS,CASE WHEN NAME = 'db block gets' THEN VALUE END DB_BLOCK_GETS,CASE WHEN NAME = 'consistent gets' THEN VALUE END CONSISTENT_GETS FROM V$SYSSTAT WHERE Name IN ('physical reads','db block gets','consistent gets'))	（监控 SGA 的命中率）命中率应大于 0.90 最好
2	库快存命中率	SELECT 100*((sum(pins-reloads))/sum(pins)) from v$librarycache	该计数器返回当前库快存命中率
3	共享区库缓存区命中率	Select round(sum(pins-reloads)/sum(pins) * 100, 2) from v$librarycache	（监控 SGA 中共享缓存区的命中率）命中率应大于 0.99
4	监控 SGA 中字典缓冲区的命中率	Select round(sum(gets-getmisses-usage-fixed)/sum(gets) * 100, 2) from v$rowcache	（共享区字典缓存区命中率）命中率应大于 0.85
5	检测回滚段的争用	select round(sum(waits)/sum(gets) * 100, 2) from v$rollstat	小于 1%
6	检测回滚段收缩次数	select sum(shrinks) from v$rollstat, v$rollname where v$rollstat.usn = v$rollname.usn	
7	监控表空间的 I/O 读总数	select sum(f.phyrds) pyr from v$filestat f, dba_data_files df where f.file# = df.file_id	监控表空间的 I/O
8	监控表空间的 I/O 块读总数	select sum(f.phyblkrd) pbr from v$filestat f, dba_data_files df where f.file# = df.file_id	监控表空间的 I/O
9	监控表空间的 I/O 写总数	select sum(f.phywrts) pyw from v$filestat f, dba_data_files df where f.file# = df.file_id	监控表空间的 I/O
10	监控表空间的 I/O	select sum(f.phyblkwrt) pbw from v$filestat f, dba_data_files df where f.file# = df.file_id	监控表空间的 I/O

	块写总数		
11	监控SGA中重做日志缓存区的命中率	SELECT Decode(immediate_gets+immediate_misses,0,0, immediate_misses/(immediate_gets+immediate_misses)*100) ratio2 FROM v$latch WHERE name IN ('redo copy')	应该小于1%
12	监控内存和硬盘的排序比率	select round(sum(case when name='sorts (disk)' then value else 0 end) / sum(case when name='sorts (memory)' then value else 0 end)*100,2) from (SELECT name, value FROM v$sysstat WHERE name IN ('sorts (memory)', 'sorts (disk)'))	最好使它小于10%

9.1.3 DB2 计数器

注：以下指标取自DB2的运行状况指示器所包含的各项指标。

指标名称	指标描述	指标范围
1．表空间存储器运行状况指示器		
自动调整大小表空间利用率 (ts.ts_util_auto_Resize %)	该指标用来跟踪每个DMS表空间的存储器消耗情况，这些DMS表空间已经定义了最大大小，并且可以自动调整大小，达到最大大小时，则认为DMS表空间已满。	该指标是用消耗的最大表空间存储器所占的百分比度量的。高百分比指示表空间接近已满程度。该指标的附加信息中包括的短期增长率和长期增长率可用来确定，当前增长率是短期畸变还是与长期增长一致。附加信息中对离空间已满所余时间的计算可以预测达到最大大小所余的时间。
表空间利用率 (ts.ts_util %)	如果在表空间上没有启用自动调整大小，则可用该指标来跟踪每个DMS表空间的存储器消耗情况；反之，DB2不会评估该指标。	该指标以消耗空间的百分比来度量。高百分比指示未达到该指标的最优运行状况。该指标的附加信息中包括的短期增长率和长期增长率可用来确定，当前增长率是短期畸变还是与长期增长一致。附加信息中对离空间已满所余时间的计算可以预测达到最大大小所余的时间。
表空间容器利用率 (ts.ts_op_status %)	该指标用来跟踪未使用自动存储器的每个SMS表空间的存储器消耗情况。如果对其定义容器的任何文件系统上都没有更多空间，则认为SMS表空间已满。如果文件系统上没有可用空间可供扩展SMS容器，则表示关联表空间已满	该指标以消耗空间的百分比来度量。高百分比指示未达到该指标的最优运行状况。该指标的附加信息中包括的短期增长率和长期增长率可用来确定，当前增长率是短期畸变还是与长期增长一致。附加信息中对离空间已满所余时间的计算可以预测达到最大大小所余的时间。

	。	
2．排序运行状况指示器		
专用排序内存利用率 (db2.sort_privmem_Util %)	该指标用来跟踪专用排序内存的利用率。	如果该指标的值等于或超过 100%，则说明已达到了排序堆阀值，没有足够的堆空间可用于执行排序。“阀值后排序数”快照监视元素可在调整该指标值时作为参考。该监视元素记录了超过排序堆阀值后请求堆的排序数。
共享排序内存利用率 (db2.sort_shrmem_Util %)	该指标用来跟踪共享排序内存的利用率。	如果该指标的值等于或超过 100%，则说明已达到了排序堆阀值，没有足够的堆空间可用于执行排序。 建议使用自调整内存功能，以根据当前工作负载的需要自动分配排序内存资源。
溢出排序百分比 (db.spilled_sorts %)	该指标值是指用完排序堆后可能需要磁盘空间以供临时存储器使用的总排序数占已执行的排序总数的利率。	该指标值应为 0，因为溢出至磁盘的排序可能导致严重的性能下降。 建议使用自调整内存功能，以根据当前工作负载的需要自动分配排序内存资源。
3．日志记录运行状况指示器		
日志利用率 (db.log_util %)	该指标用来跟踪在数据库中使用的总活动日志空间量。	该指标以消耗空间的百分比来度量。高百分比指示空间消耗接近已满程度。这时可调整一些与日志有关的数据库配置参数的值。这些参数的值显示在附加信息中。
日志文件系统利用率 (db.log_fs_util %)	该指标用来跟踪事务日志所在的文件系统的充满程度。如果文件系统上没有空间，则 DB2 可能无法创建新的日志文件。	该指标以消耗空间的百分比来度量。高百分比指示文件系统中的可用空间量已接近于 0。这时可调整一些与日志有关的数据库配置参数的值。这些参数的值显示在附加信息中。
4．应用程序并发性运行状况指示器		
死锁率 (db.deadlock_rate%)	该指标用来跟踪死锁出现在数据库上的比率以及应用程序遇到争用问题的等级。	该指标值应为 0，该值越高，则争用等级就越高。
锁定列表利用率 (db.locklist_util %)	该指标用来跟踪要使用的锁定列表内存量。每个数据库有一个锁定列表，锁定列表包含由同时连接至数据库的所有应用程序挂起的锁定。这是对锁定列表内存设置的限制。一旦达到该限制，就会因为下列	该指标以消耗内存的百分比来度量，出现高百分比表示状况不佳。 建议使用自调整内存功能，以根据当前工作负载的需要自动分配排序内存资源。

	情况而使得性能下降： 锁定升级将行锁定转换为表锁定，从而降低了数据库中的共享对象的并行性； 因为应用程序等待有限数目的表锁定，所以应用程序间会出现更多死锁。因此将回滚事务。	
等待锁定的 (db.Apps_waiting_locks %)	该指标度量所有当前执行的等待锁定的应用程序所占的百分比。	高百分比可能指示应用程序遇到并行性问题，这对性能有负面影响。
5．程序包和目录高速缓存，以及工作空间运行状况指示器		
目录高速缓存命中率 (db.catcache_hitratio %)	该指标用于指示目录高速缓存对避免对磁盘上的目录的实际访问所起到的帮助作用。	高命中率指示在避免实际磁盘 I/O 访问方面很成功。
程序包高速缓存命中率 (db.pkgcache_hitratio %)	该指标用于指示程序包高速缓存对避免从系统目录重新装入静态 SQL 的程序包和段以及避免重新编译动态 SQL 语句所起到的帮助作用。	高命中率指示在避免从系统目录重新装入静态 SQL 的程序包和段以及避免重新编译动态 SQL 语句方面很成功。
共享工作空间命中率 (db.shrworkspace_hitratio %)	该指标用于指示共享 SQL 工作空间对避免初始化要执行的 SQL 语句的各段所起到的帮助作用。	高命中率指示在避免初始化要执行的 SQL 语句的各段方面很成功。
6．内存运行状况指示器		
数据库堆利用率 (db.db_heap_util %)	该指标用来跟踪基于带有标识SQLM_HEAP_DATABASE的内存池的监视器堆内存的消耗。	一旦此百分比达到最大值 100%，查询和操作可能会因为没有堆可用而失败。

第 10 章 Protocol

本章只是对网络协议做了简单的介绍，想了解更多，请参照专业书籍或查阅网络资料。这里推荐一个网站：http://www.cnpaf.net。

第 1 节 一些名词

LAN、MAN、WAN、Internetwork、网关（gateway）；

子网：指把分组从源主机传送到目的主机的路由器和通信线路的集合。它包含了与网络寻址相关的含义。

子网就是当某一段默认网段的主机数量不够用时，通过修改默认子网掩码的方法划分出不同的子网段，以增加通信的主机数量。

拓扑结构：网络的结构，个个设备路由之间的连接方式。

网关：一台机器，用来连接相互不兼容的网络，并提供硬件和软件的转换。

互联网：通过 WAN 把 LAN 连接起来的集合。

服务：在形式上是由一组据原语（操作）来描述的，包括：请求、提示、响应、证实。

是各层向它的上层提供的一组原语，尽管能够代表上一层完成的操作，但不涉及上层是如何完成的。

协议：定义了同层之间对等实体之间交换的桢、分组和报文的格式及意义的一组规则。实体使用协议来实现服务。

第 2 节 网络协议

七层协议：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。其中，前 3 层是链接起来的，4-7 层是端对端的。

✧ 物理层（physical layer）

通信在信道上传输的比特流（多少伏特代表 0 或 1）

✧ 数据链路层（data link layer）

为网络层提供无错的线路；防止重复桢；调节流量；借道（回复桢和数据桢）。

✧ 网络层（network layer）

确定如何选择路由（动态和静态）；防止分组导致的阻塞；网络记费。

✧ 传输层（transport layer）

从会话层接收数据，分割为教小单元在适当时机传递给网络层。

✧ 会话层（session layer）

允许不同机器上的用户建立会话关系，允许普通数据的传输；可以控制单向传输或双向传输；令牌管理（不能同时做同样操作）；同步（断点续传）。

✧ 表示层（presentaltion layer）

完成一些通用功能[比如各种数据代码（ASCII、Unicode）之间的转换]，表示层关心所传输的信息的语法与意义，而以下的层关心的是可靠的传输比特流。

✧ 应用层（application layer）

包含大量普遍需要的协议，比如各种终端的型号（终端的型号不同问题用网络虚拟终端

的方式来解决；文件传输（包括不同文件系统之间的传输控制）；电子邮件；远程作业输入、名录查询等。

第 3 节 TCP/IP 协议

TCP/IP（TransmissionControlProtocol/InternetProtocol 的简写，中文译名为传输控制协议/互联网络协议）协议是 Internet 最基本的协议，简单地说，就是由底层的 IP 协议和 TCP 协议组成的。TCP/IP 协议的开发工作始于 70 年代，是用于互联网的第一套协议。

10.3.1 TCP/IP 参考模型

TCP/IP 协议的开发研制人员将 Internet 分为五个层次，包括：应用层、传输层、互连层（网络层）、主机至网络。如下表：

应用层（第五层）
传输层（第四层）
互联网层（第三层）
网络接口层（第二层）
物理层（第一层）

物理层：对应于网络的基本硬件，这也是 Internet 物理构成，即我们可以看得见的硬件设备，如 PC 机、互连网服务器、网络设备等，必须对这些硬件设备的电气特性作一个规范，使这些设备都能够互相连接并兼容使用。

网络接口层：它定义了将数据组成正确帧的规程和在网络中传输帧的规程，帧是指一串数据，它是数据在网络中传输的单位。

互联网层：本层定义了互联网中传输的“信息包”格式，以及从一个用户通过一个或多个路由器到最终目标的"信息包"转发机制。

传输层：为两个用户进程之间建立、管理和拆除可靠而又有效的端到端连接。

应用层：它定义了应用程序使用互联网的规程。application layer 包含了所有的高层协议：虚拟终端协议 telnet、文件传输协议 ftp、电子邮件协议 smtp、域名系统协议 DNS（domain name service）把主机名映射为网络地址、NNTP 协议用于传递新闻、HTTP 协议用语在 WWW 上获取主页。

10.3.2 网间协议 IP

Internet 上使用的一个关键的底层协议是网际协议，通常称 IP 协议。我们利用一个共同遵守的通信协议，从而使 Internet 成为一个允许连接不同类型的计算机和不同操作系统的网络。要使两台计算机彼此之间进行通信，必须使两台计算机使用同一种"语言"。通信协议正像两台计算机交换信息所使用的共同语言，它规定了通信双方在通信中所应共同遵守的约定。

计算机的通信协议精确地定义了计算机在彼此通信过程的所有细节。例如，每台计算机发送的信息格式和含义，在什么情况下应发送规定的特殊信息，以及接收方的计算机应做出哪些应答等等。

网际协议 IP 协议提供了能适应各种各样网络硬件的灵活性，对底层网络硬件几乎没有

任何要求，任何一个网络只要可以从一个地点向另一个地点传送二进制数据，就可以使用 IP 协议加入 Internet 了。

如果希望能在 Internet 上进行交流和通信，则每台连上 Internet 的计算机都必须遵守 IP 协议。为此使用 Internet 的每台计算机都必须运行 IP 软件，以便时刻准备发送或接收信息。

IP 协议对于网络通信有着重要的意义：网络中的计算机通过安装 IP 软件，使许许多多的局域网络构成了一个庞大而又严密的通信系统。从而使 Internet 看起来好像是真实存在的，但实际上它是一种并不存在的虚拟网络，只不过是利用 IP 协议把全世界上所有愿意接入 Internet 的计算机局域网络连接起来，使得它们彼此之间都能够通信。

10.3.3 传输控制协议 TCP

尽管计算机通过安装 IP 软件，从而保证了计算机之间可以发送和接收数据，但 IP 协议还不能解决数据分组在传输过程中可能出现的问题。因此，若要解决可能出现的问题，连上 Internet 的计算机还需要安装 TCP 协议来提供可靠的并且无差错的通信服务。

TCP 协议被称作一种端对端协议。这是因为它为两台计算机之间的连接起了重要作用: 当一台计算机需要与另一台远程计算机连接时，TCP 协议会让它们建立一个连接、发送和接收数据以及终止连接。

传输控制协议 TCP 协议利用重发技术和拥塞控制机制，向应用程序提供可靠的通信连接，使它能够自动适应网上的各种变化。即使在 Internet 暂时出现堵塞的情况下，TCP 也能够保证通信的可靠。

众所周知，Internet 是一个庞大的国际性网络，网路上的拥挤和空闲时间总是交替不定的，加上传送的距离也远近不同，所以传输数据所用时间也会变化不定。TCP 协议具有自动调整"超时值"的功能，能很好地适应 Internet 上各种各样的变化，确保传输数值的正确。因此，从上面我们可以了解到：IP 协议只保证计算机能发送和接收分组数据，而 TCP 协议则可提供一个可靠的、可流控的、全双工的信息流传输服务。

综上所述，虽然 IP 和 TCP 这两个协议的功能不尽相同，也可以分开单独使用，但它们是在同一时期作为一个协议来设计的，并且在功能上也是互补的。只有两者的结合，才能保证 Internet 在复杂的环境下正常运行。凡是要连接到 Internet 的计算机，都必须同时安装和使用这两个协议，因此在实际中常把这两个协议统称作 TCP/IP 协议。

10.3.4 IP 地址及其分类

在 Internet 上连接的所有计算机，从大型机到微型计算机都是以独立的身份出现，我们称它为主机。为了实现各主机间的通信，每台主机都必须有一个唯一的网络地址。就好像每一个住宅都有唯一的门牌一样，才不至于在传输数据时出现混乱。

Internet 的网络地址是指连入 Internet 网络的计算机的地址编号。所以，在 Internet 网络中，网络地址唯一地标识一台计算机。

我们都已经知道，Internet 是由几千万台计算机互相连接而成的。而我们要确认网络上的每一台计算机，靠的就是能唯一标识该计算机的网络地址，这个地址就叫做 IP（InternetProtocol 的简写）地址，即用 Internet 协议语言表示的地址。

目前，在 Internet 里，IP 地址是一个 32 位的二进制地址，为了便于记忆，将它们分为 4 组，每组 8 位，由小数点分开，用四个字节来表示，而且，用点分开的每个字节的数值范围是 0~255，如 202.116.0.1，这种书写方法叫做点数表示法。

IP 地址可确认网络中的任何一个网络和计算机，而要识别其他网络或其中的计算机，则是根据这些 IP 地址的分类来确定的。一般将 IP 地址按节点计算机所在网络规模的大小分为 A，B，C 三类，默认的网络掩码是根据 IP 地址中的第一个字段确定的。

A 类地址

A 类地址的表示范围为：0.0.0.0~126.255.255.255，默认网络掩码为：255.0.0.0；A 类地址分配给规模特别大的网络使用。A 类网络用第一组数字表示网络本身的地址，后面三组数字作为连接于网络上的主机的地址。分配给具有大量主机（直接个人用户）而局域网络个数较少的大型网络。例如 IBM 公司的网络。

B 类地址

B 类地址的表示范围为：128.0.0.0~191.255.255.255，默认网络掩码为：255.255.0.0；B 类地址分配给一般的中型网络。B 类网络用第一、二组数字表示网络的地址，后面两组数字代表网络上的主机地址。

C 类地址

C 类地址的表示范围为：192.0.0.0~223.255.255.255，默认网络掩码为：255.255.255.0；C 类地址分配给小型网络，如一般的局域网和校园网，它可连接的主机数量是最少的，采用把所属的用户分为若干的网段进行管理。C 类网络用前三组数字表示网络的地址，最后一组数字作为网络上的主机地址。

实际上，还存在着 D 类地址和 E 类地址。但这两类地址用途比较特殊，在这里只是简单介绍一下：D 类地址称为广播地址，供特殊协议向选定的节点发送信息时用。E 类地址保留给将来使用。

连接到 Internet 上的每台计算机，不论其 IP 地址属于哪类都与网络中的其他计算机处于平等地位，因为只有 IP 地址才是区别计算机的唯一标识。所以，以上 IP 地址的分类只适用于网络分类。

在 Internet 中，一台计算机可以有一个或多个 IP 地址，就像一个人可以有多个通信地址一样，但两台或多台计算机却不能共用一个 IP 地址。如果有两台计算机的 IP 地址相同，则会引起异常现象，无论哪台计算机都将无法正常工作。

顺便提一下几类特殊的 IP 地址：

1. 广播地址目的端为给定网络上的所有主机，一般主机段为全 0
2. 单播地址目的端为指定网络上的单个主机地址
3. 组播地址目的端为同一组内的所有主机地址
4. 环回地址 127.0.0.1 在环回测试和广播测试时会使用

10.3.5 子网的划分

若公司不上 Internet,那一定不会烦恼 IPAddress 的问题,因为可以任意使用所有的 IPAddress,不管是 AClass 或是 BClass,这个时候不会想到要用 SubNet,但若是上 Internet 那 IPAddress 便弥足珍贵了,目前全球一阵 Internet 热,IPAddress 已经愈来愈少了,而所申请的 IPAddress 目前也趋保守,而且只有经申请的 IPAddress 能在 Internet 使用,但对某些公司只能申请到一个 CCLass 的 IPAddress,但又有多个点需要使用,那这时便需要使用到 Subnet,这就需要考虑子网的划分。

10.3.6 几个常用的程序

10.3.6.1 Ping

这个程序用来检测一帧数据从当前主机传送到目的主机所需要的时间。当网络运行中出现故障时，采用这个实用程序来预测故障和确定故障源是非常有效的。如果执行 ping 不成功，则可以预测故障出现在以下几个方面：网线是否连通，网络适配器配置是否正确，IP 地址是否可用等；如果执行 ping 成功而网络仍无法使用，那么问题很可能出在网络系统的软件配置方面，ping 成功只能保证当前主机与目的主机间存在一条连通的物理路径。它还提供了许多参数，如－t 使当前主机不断地向目的主机发送数据，直到使用 Ctrl－C 中断；－n 可以自己确定向目的主机发送的数据帧数等等。

10.3.6.2 Winipcfg

它用来显示主机内 IP 协议的配置信息。它采用 Windows 窗口的形式显示具体信息。这些信息包括：网络适配器的物理地址、主机的 IP 地址、子网掩码以及默认网关等，还可以查看主机的相关信息如：主机名、DNS 服务器、节点类型等。其中网络适配器的物理地址在检测网络错误时非常有用。

10.3.6.3 tracert

这个程序的功能是判定数据包到达目的主机所经过的路径、显示数据包经过的中继节点清单和到达时间。还可以使用参数－d 决定是否解析主机名。

10.3.6.4 Netstat

这个程序有助于我们了解网络的整体使用情况。它可以显示当前正在活动的网络连接的详细信息，如采用的协议类型、当前主机与远端相连主机（一个或多个）的 IP 地址以及它们之间的连接状态等。它提供的较为常用的参数是：－e 用以显示以太网的统计信息；－s 显示所有协议的使用状态，这些协议包括 TCP、UDP 和 IP，一般这两个参数都是结合在一起使用的。另外－p 可以选择特定的协议并查看其具体使用信息，－a 可以显示所有主机的端口号，－r 则显示当前主机的详细路由信息。

要运行以上这些程序，只要在 DOS 方式或 Windows 开始菜单的运行栏中以命令行的形式键入程序名即可。灵活使用这几个程序可以使你大体了解自己主机对网络的使用情况。

第 4 节 HTTP 协议

超文本传输协议 HTTP（HyperText Transfer Protocol）是专门为传输 HTML 页面的一种网络协议。

10.4.1 HTTP 概述

在 TCP/IP 体系结构中，HTTP 属于应用层协议，位于 TCP/IP 协议的顶层。浏览 Web 时，浏览器通过 HTTP 协议与 Web 服务器交换信息。这些信息（文档）类型的格式由 MIME 定义。

HTTP 协议具有以下的特点：

- ✧ HTTP 按客户/服务器模式工作；
- ✧ HTTP 支持客户（一般情况是浏览器）与服务器的通讯，相互传输数据。；
- ✧ HTTP 定义的事务处理由以下四步组成：
  客户与服务器建立连接；
  客户向服务器提出请求；
  如果请求被接受，则服务器送回响应，在响应中包括状态码和所需的文件；
  客户与服务器断开连接
- ✧ HTTP 是无状态的。也就是说，浏览器和服务器每进行一次 HTTP 操作，就建立一次连接，但任务结束就中断连接。
- ✧ HTTP 使用元信息作为头标：HTTP 对所有事务都加了头标（header）。也就是说，在主要数据前加上一块信息，称为元信息（metainformation）。它使服务器能够提供正在传送数据的有关信息。例如，传送对象是哪种类型，是用哪种语言书写的等。
- ✧ 从功能上讲，HTTP 支持四类元信息：一般信息头标、请求头标、响应头标和实体头标 。
- ✧ HTTP 支持两种请求和响应格式：HTTP 由不同的两部分组成，一是从浏览器发往服务器的请求，二是服务器对客户的响应。
- ✧ HTTP 支持两种请求和响应，即简单请求与完全请求和简单响应与完全响应。
- ✧ HTTP 是基于文本的简单协议 。

10.4.2 HTTP 的工作过程

每一个 Web 主机都有一个服务器进程来监听 TCP 端口 80，以便同前来建立连接的客户取得联系。连接建立后，客户发送一个请求，服务器返回一个响应，然后就释放连接。

例如，我们单击一个 http://www.csgz.edu.cn 超连接后，浏览器与服务器之间工作过程大致如下：

浏览器确定页面的 URL；
浏览器向 DNS（域名解析服务）查询 www.csgz.edu.cn 的 IP 地址；
DNS 返回 IP 地址 210.28.160.1；
浏览器向 210.28.160.1 的端口 80 请求建立 TCP 连接；
浏览器发送 GET /hypertext/www/TheProject.html 命令；
www.csgz.edu.cn 服务器发送 TheProject.html 文件；
TCP 连接被释放；
浏览器显示 TheProject.html 中的所有文本；
浏览器获取和显示 TheProject.html 中的全部图象。

值得注意的是，对于页面中的每一幅图象（包括图标、图形、照片等），浏览器都必须与服务器建立一次 TCP 连接，获取图象后释放连接。

HTTP 请求与响应

除建立与释放连接外，HTTP 事务处理的主要内容是客户端的请求与服务器端的响应。

HTTP 的常用请求方法：

方 法	说 明
GET	请求读取一个 Web 页面
HEAD	请求读取一个 Web 页面的头标
PUT	请求存储一个 Web 页面
POST	附加到命名资源中
DELETE	删除 Web 页面
LINK	连接两个已有资源
UNLINK	取消两个资源之间的已有连接

10.4.3 HTTP 请求

HTTP 请求的格式如下所示：

```
    ＜request-line＞
＜headers＞
＜blank line＞
[＜request-body＞]
```

在 HTTP 请求中，第一行必须是一个请求行（request line），用来说明请求类型、要访问的资源以及使用的 HTTP 版本。紧接着是一个首部（header）小节，用来说明服务器要使用的附加信息。在首部之后是一个空行，再此之后可以添加任意的其他数据[称之为主体（body）]。

在 HTTP 中，定义了大量的请求类型，不过 Ajax 开发人员关心的只有 GET 请求和 POST 请求。只要在 Web 浏览器上输入一个 URL，浏览器就将基于该 URL 向服务器发送一个 GET 请求，以告诉服务器获取并返回什么资源。对于 www.wrox.com 的 GET 请求如下所示：

```
    GET / HTTP/1.1
Host: www.wrox.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

请求行的第一部分说明了该请求是 GET 请求。该行的第二部分是一个斜杠（/），用来说明请求的是该域名的根目录。该行的最后一部分说明使用的是 HTTP 1.1 版本（另一个可选项是 1.0）。那么请求发到哪里去呢？这就是第二行的内容。

第 2 行是请求的第一个首部，HOST。首部 HOST 将指出请求的目的地。结合 HOST 和上一行中的斜杠（/），可以通知服务器请求的是 www.wrox.com/（HTTP 1.1 才需要使用首部 HOST，而原来的 1.0 版本则不需要使用）。第三行中包含的是首部 User-Agent，服务器端和客户端脚本都能够访问它，它是浏览器类型检测逻辑的重要基础。该信息由你使用的浏览器来定义（在本例中是 Firefox 1.0.1），并且在每个请求中将自动发送。最后一行是首部 Connection，通常将浏览器操作设置为 Keep-Alive（当然也可以设置为其他值，但这已经超出了本书讨论的范围）。注意，在最后一个首部之后有一个空行。即使不存在请求主体，这

个空行也是必需的。

如果要获取一个诸如 http://www.wrox.com/books 的 www.wrox.com 域内的页面，那么该请求可能类似于：

```
GET /books/ HTTP/1.1
Host: www.wrox.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

注意只有第一行的内容发生了变化，它只包含 URL 中 www.wrox.com 后面的部分。

要发送 GET 请求的参数，则必须将这些额外的信息附在 URL 本身的后面。其格式类似于：

```
URL ? name1=value1&name2=value2&..&nameN=valueN
```

该信息称之为查询字符串（query string），它将会复制在 HTTP 请求的请求行中，如下所示：

```
GET /books/?name=Professional%20Ajax HTTP/1.1
Host: www.wrox.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

注意，为了将文本“Professional Ajax”作为 URL 的参数，需要编码处理其内容，将空格替换成%20，这称为 URL 编码（URL encoding），常用于 HTTP 的许多地方（JavaScript 提供了内建的函数来处理 URL 编码和解码，这些将在本章中的后续部分中说明）。“名称—值”（name—value）对用 & 隔开。绝大部分的服务器端技术能够自动对请求主体进行解码，并为这些值的访问提供一些逻辑方式。当然，如何使用这些数据还是由服务器决定的。

浏览器发送的首部，通常比本文中所讨论的要多得多。为了简单起见，这里的例子尽可能简短。

另一方面，POST 请求在请求主体中为服务器提供了一些附加的信息。通常，当填写一个在线表单并提交它时，这些填入的数据将以 POST 请求的方式发送给服务器。

```
POST / HTTP/1.1
Host: www.wrox.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
Connection: Keep-Alive
name=Professional%20Ajax&publisher=Wiley
```

从上面可以发现， POST 请求和 GET 请求之间有一些区别。首先，请求行开始处的 GET 改为了 POST，以表示不同的请求类型。你会发现首部 Host 和 User-Agent 仍然存在，在后面有两个新行。其中首部 Content-Type 说明了请求主体的内容是如何编码的。浏览器始终以 application/ x-www-form- urlencoded 的格式编码来传送数据，这是针对简单 URL 编码的 MIME 类型。首部 Content-Length 说明了请求主体的字节数。在首部 Connection 后是一个空行，再后面就是请求主体。与大多数浏览器的 POST 请求一样，这是以简单的“名称—

值”对的形式给出的，其中 name 是 Professional Ajax，publisher 是 Wiley。你可以以同样的格式来组织 URL 的查询字符串参数。

正如前面所提到的，还有其他的 HTTP 请求类型，它们遵从的基本格式与 GET 请求和 POST 请求相同。下一步我们来看看服务器将对 HTTP 请求发送什么响应。

10.4.4 HTTP 响应

如下所示，HTTP 响应的格式与请求的格式十分类似：

```
    ＜status-line＞
＜headers＞
＜blank line＞
[＜response-body＞]
```

正如你所见，在响应中唯一真正的区别在于第一行中用状态信息代替了请求信息。状态行（status line）通过提供一个状态码来说明所请求的资源情况。以下就是一个 HTTP 响应的例子：

```
    HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 122

＜html＞
＜head＞
＜title＞Wrox Homepage＜/title＞
＜/head＞
＜body＞
＜!-- body goes here --＞
＜/body＞
＜/html＞
```

在本例中，状态行给出的 HTTP 状态代码是 200，以及消息 OK。状态行始终包含的是状态码和相应的简短消息，以避免混乱。最常用的状态码有：

◆200 (OK): 找到了该资源，并且一切正常。

◆304 (NOT MODIFIED): 该资源在上次请求之后没有任何修改。这通常用于浏览器的缓存机制。

◆ 401 (UNAUTHORIZED): 客户端无权访问该资源。这通常会使得浏览器要求用户输入用户名和密码，以登录到服务器。

◆403 (FORBIDDEN): 客户端未能获得授权。这通常是在 401 之后输入了不正确的用户名或密码。

◆404 (NOT FOUND): 在指定的位置不存在所申请的资源。

在状态行之后是一些首部。通常，服务器会返回一个名为 Data 的首部，用来说明响应生成的日期和时间（服务器通常还会返回一些关于其自身的信息，尽管并非是必需的）。接下来的两个首部大家应该熟悉，就是与 POST 请求中一样的 Content-Type 和 Content-Length。在本例中，首部 Content-Type 指定了 MIME 类型 HTML（text/html），其编码类型是 ISO-8859-1

（这是针对美国英语资源的编码标准）。响应主体所包含的就是所请求资源的 HTML 源文件（尽管还可能包含纯文本或其他资源类型的二进制数据）。浏览器将把这些数据显示给用户。

注意，这里并没有指明针对该响应的请求类型，不过这对于服务器并不重要。客户端知道每种类型的请求将返回什么类型的数据，并决定如何使用这些数据。

10.4.5 HTTP 状态码

1xx：信息响应类，表示接收到请求并且继续处理；
2xx：处理成功响应类，表示动作被成功接收、理解和接受；
3xx：重定向响应类，为了完成指定的动作，必须接受进一步处理；
4xx:：客户端错误，客户请求包含语法错误或者是不能正确执行；
5xx：服务端错误，服务器不能正确执行一个正确的请求；
详细信息，请利用网络资源查询。

第 11 章 测试工具

随着 web 应用的增多，服务器应用解决方案中以 Web 为核心的应用也越来越多， 很多公司各种应用的架构都以 web 应用为主。一般的 web 测试和以往的应用程序的测试的侧重点不完全相同，在基本功能已经通过测试后，就要进行重要的系统性能测试了。系统的性能是一个很大的概念，覆盖面非常广泛，对一个软件系统而言包括执行效率、资源占用率、稳定性、安全性、兼容性、可靠性等等，以下重点从系统的负载和压力需要采用负载测试工具进行介绍。

性能测试工具，基本工作原理是类似的，网络上流行的很多测试工具的 PDF 文档都写的都不错，这里只列出标题，不做讲解。

第 1 节 MI Loadrunner

LoadRunner 是 Mercury Interactive 公司推出的专业预测系统行为和性能的负载测试工具。是目前世界上最强大的负载测试工具之一，提供了非常强大的监视功能，能够监控各种软硬件模块。

通过以模拟上千万用户实施并发负载及实时性能监测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。

Load Runner 支持 HTTP(S)、WAP、i-Mode、 RealPlayer、LDAP、Winsock、RMI、FTP、POP3、SMTP、CORBA、COM/DCOM 以及 Tuxedo 等。在监视器部分，它支持 Windows NT/2000/XP、 SUN Solaris、HP UX、IBM AIX 和 Linux 等操作系统，支持 Apache、Web Logic 等各种 Web Server，还支持各种大型数据库。

Load Runner 支持产生各种虚拟用户，包括：GUI、数据库、Java 和远程终端仿真器（RTE）虚拟用户。Load Runner 仅用几台 Windows NT 或 UNIX 就可以仿真上千个虚拟用户，帮助最大限度的降低测试所需的硬件资源。一个 GUI 虚拟用户仅是一个代理程序，它可发送、驱动和测量一个真实客户端应用的性能。

第 2 节 Apache Jmeter

JMeter 是 Apache 组织的开放源代码项目，它是功能和性能测试的工具，100%的用 java 实现。它对 windows 和 unix、linux 都有很好的支持性，并且脚本是可以通用的。不过，它和其他 java 应用程序一样，执行过程中需要占用大量内存。

第 3 节 Rational Robot

IBM Rational Robot 是业界最顶尖的功能测试工具，它甚至可以在测试人员学习高级脚本技术之前帮助其进行成功的测试。它集成在测试人员的桌面 IBM Rational TestManager 上，在这里测试人员可以计划、组织、执行、管理和报告所有测试活动，包括手动测试报告。这种测试和管理的双重功能是自动化测试的理想开始。

Rational 在软件测试方面也有非常好的成绩。该公司推出的 Robot 工具支持 SQABasic 这种面向对象的记录语言。不过，在性能测试方面，Robot 并不是很出名。但是 Robot 提供了一种新的脚本记录语言—VU 语言，它基于传统的 C 语言，能够方便地访问 Robot 提供的环境变量。同时 Robot 还提供了很多良好定义的库函数，调用通信函数更加方便。Robot 还提供了其他许多相关测试技术，例如数据池(Datapool)、同步点等，并且通过 TestManager 可以对所有类型脚本进行管理。

从功能来说，Robot 支持众多的网络协议，例如 COM、DCOM、SOCKET、IIOP、Tuxedo 等，并且可以对协议进行过滤，选取自己关心的协议。从操作方面来说，它对用户的要求也比较高，需要用户在整个访问过程中，对客户和服务器之间的交互类型和内容比较熟悉，同时对 Robot 也必须有足够的了解。

第 4 节 Web Application Stress

Microsoft Web Application Stress Tool 是由微软的网站测试人员所开发，专门用来进行实际网站压力测试的一套工具。透过这套功能强大的压力测试工具，您可以使用少量的 Client 端计算机仿真大量用户上线对网站服务所可能造成的影响。

第 5 节 Compuware QAload

Compuware 公司的 QALoad 是一个企业级的自动化负载测试工具。QALoad 还提供一个定义、管理和执行负载测试的中心控制点-Conductor。它能够自动识别网络中可进行负载测试的机器，并在这些机器之间自动分布工作量，以避免网段超载。通过执行测试脚本。它还能管理无数的虚拟用户。从 Conductor 自动启动和配置远程用户，跨国机构甚至可以进行全球负载测试。

第 6 节 RadView Webload

webload 是 RadView 公司推出的一个性能测试和分析工具，它让 web 应用程序开发者自动执行压力测试；webload 通过模拟真实用户的操作，生成压力负载来测试 web 的性能。

用户创建的是基于 javascript 的测试脚本,称为议程 agenda,用它来模拟客户的行为,通过执行该脚本来衡量web应用程序在真实环境下的性能。webload 提供巡航控制器 cruise control 的功能,利用巡航控制器,可以预定义 web 应用程序应该满足的性能指标,然后测试系统是否满足这些需求指标;cruise control 能够自动把负载加到 web 应用程序,并将在此负荷下能够访问程序的客户数量生成报告。

webload 能够在测试会话执行期间对监测的系统性能生成实时的报告,这些测试结果通过一个易读的图形界面显示出来,并可以导出到 excel 和其他文件里。

webload 是一个性能测试和分析的工具,让 web 应用程序开发者自动执行压力测试;webload 通过模拟真实用户的操作,生成压力负载来测试 web 的性能;用户创建的是基于 javascript 的测试脚本,称为议程 agenda,用它来模拟客户的行为,通过执行该脚本来衡量 web 应用程序在真实环境下的性能。

webload 提供巡航控制器 cruise control 的功能,利用巡航控制器,可以预定义 web 应用程序应该满足的性能指标,然后测试系统是否满足这些需求指标;cruise control 能够自动把负载加 web 应用程序,并将在此负荷下能够访问程序的客户数量生成报告。

第 7 节 其他评测工具

11.7.1 Jrockit 监控工具

作为第一个专门针对 Intel 平台并得到优化的 JVM，BEA JRockit 解决了这一问题。它使 Java 应用能在成本更低、基于标准的平台上，以更高的可靠性和更佳的性能运行。与其他的 JVM 不同，BEA JRockit 旨在驱动要求极高的服务器端 Java 应用，以便为企业应用提供极高的性能、可管理性和可靠性。

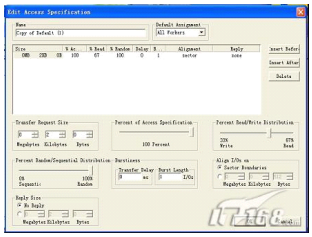
11.7.2 Netperf

Netperf 可以测试服务器网络性能，主要针对基于 TCP 或 UDP 的传输。Netperf 根据应用的不同，可以进行不同模式的网络性能测试，即批量数据传输（bulk data transfer）模式和请求/应答（request/reponse）模式。Netperf 测试结果所反映的是一个系统能够以多快的速度向另外一个系统发送数据，以及另外一个系统能够以多块的速度接收数据。

Netperf 工具以 client/server 方式工作。server 端是 netserver，用来侦听来自 client 端的连接，client 端是 netperf，用来向 server 发起网络测试。在 client 与 server 之间，首先建

立一个控制连接，传递有关测试配置的信息，以及测试的结果；在控制连接建立并传递了测试配置信息以后，client 与 server 之间会再建立一个测试连接，用来来回传递着特殊的流量模式，以测试网络的性能。

11.7.3 IOmeter



Iometer 是 Windows 系统下对存储子系统的读写性能进行测试的软件。可以显示磁盘系统的最大 IO 能力、磁盘系统的最大吞吐量、CPU 使用率、错误信息等。用户可以通过设置不同的测试的参数，有存取类型(如 sequential ,random)、读写块大小(如 64K、256K)，队列深度等，来模拟实际应用的读写环境进行测试。

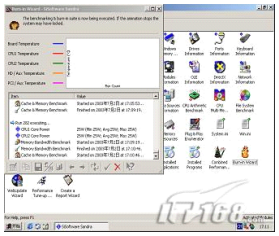
Iometer 操作简单，可以录制测试脚本，可以准确有效的反映存储系统的读写性能，为各大服务器和存储厂商所广泛采用。

11.7.4 IOzone

现在有很多的服务器系统都是采用 linux 操作系统，在 linux 平台下测试 I/O 性能可以采用 iozone。

iozone 是一个文件系统的 benchmark 工具，可以测试不同的操作系统中文件系统的读写性能。可以测试 Read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random read, pread ,mmap, aio_read, aio_write 等等不同的模式下的硬盘的性能。测试所有这些方面，生成 excel 文件，另外, iozone 还附带了用 gnuplot 画图的脚本。

11.7.5 Sisoft Sandra



SiSoft 发行的 Sandra 系列测试软件是 Windows 系统下的基准评测软件。此软件有超过三十种以上的测试项目，能够查看系统所有配件的信息，而且能够对部分配件（如 CPU、内存、硬盘等）进行打分(benchmark)，并且可以与其它型号硬件的得分进行对比。另外，该软件还有系统稳定性综合测试、性能调整向导等附加功能。

第 8 节 性能工具清单

生产厂商	工具名称	网址链接
Mercury Interactive Corporation	Loadrunner	http://www.mercury.com/us/products/
	Astra LoadTest	
IBM Rational	Rational robot	http://www-900.ibm.com/cn/software/rational/products/index.shtml
compuware corporation	QALoad	http://www.compuware.com/products/
Segue software	SilkPerformer	http://www.segue.com/products/index.asp
Empirix	e-Load	http://www.empirix.com/Empirix/Web+Test+Monitoring/Testing+SolutionsIntegrated+Web+Testing.html
RadView	WebLoad	http://www.radview.com/products/index.asp
Quest Software	Benchmark Factory	http://www.quest.com/benchmark_factory/
MicroSoft	WebApplication Stress	http://www.microsoft.com/technet/archive/itsolutions/intranet/downloads/webtutor.mspx
	Jmeter	
	OpenSTA	
	Grinder	
	httperf	
	Http-Load	