软件性能与性能测试教程 第 2 部分

中国软件评测中心　陈 兵

本系列教程对软件性能与性能测试这个主题进行了较为全面的介绍和分析。教程分为三个部分：在第 1 部分，讲解了软件性能的基本知识，介绍了常用的软件性能指标，在分析影响软件性能因素的同时，以实例清晰讲解了如何提高软件的性能及性能的可扩展性等问题；在第 2 部分，首先给出了性能测试的基础知识，然后侧重于从性能工程的角度提出开展性能测试工作的流程，和进行性能测试工作的策略，以及如何分析性能测试数据；在第 3 部分，从开发的（而不是第 3 方测试）角度，以实例为引导，一步一步地分析讲解在开发环境下进行性能测试的策略、方法和手段（采用的工具）。

本系列教程可以用作软件开发人员、软件测试人员、软件项目经理、软件质量人员和需要了解软件性能测试的各级软件管理人员的工作参考手册，也可作为有志于软件开发和软件性能测试领域人员的参考资料。...

1. 软件性能测试

各种软件在系统实施过程中，需要满足客户的一些特殊要求。如果软件系统没有经过测试和优化，软件系统将无法满足用户的需求，还会给软件在实际应用中带来很大的风险。一些公司缺乏必要的测试手段和工具知识导致测试不彻底，其中性能测试是整个开发、测试过程中一个重要方面。

（1）什么是性能测试

性能测试用来保证产品发布后系统的性能满足用户需求。性能测试在软件质量保证中起重要的作用。通俗地说，性能测试主要是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行的一种测试。性能测试包含三种类型的测试：

- 负载测试

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、内存使用等如何决定系统的性能，例如稳定性和响应等。　负载测试通常描述一种特定类型的压力测试，即增加用户数量以对应用程序进行压力测试。

通俗的说，这种测试方法就是模拟真实环境下的用户活动，在特定的运行条件下验证系统的能力状况。在项目中，通常是测试现有负载和预期负载的负载压力测试，进行对比分析。

- 压力测试

压力测试通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大的服务级别的测试。通俗地讲，压力测试是为了发现在什么条件下您的应用程序的性能会变得不可接受。在压力测试中，可以采取两种不同的压力情况--用户量压力测试或数据量压力测试。

- 疲劳强度测试

通常是采用系统稳定运行情况下能够支持的最大并发用户数或 者日常运行用户数，持续执行一段时间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理最大工作量强度性能的过程。

（2）性能测试的目的

为什么要进行性能测试呢？其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。主要包括以下几个方面

1）评估系统的能力，测试中得到的负荷和响应时间数据可以被用于验证所计划的模型的能力，并帮助作出决策。

2）识别体系中的弱点：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。

3）系统调优：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。

4）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露引起的失败，揭示程序中的隐含的问题或冲突。

5） 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。

（3）压力测试能够发现缺陷

进行压力测试，您有希望找到很多种用其他测试方法更难发现的错误。主要有两种错误类型：

- **内存泄漏（Memory leak）：**一种极难检测的现象（或者说一般在单元级的开发和测试过程中往往会忽视的问题，现在有许多商品化或开源工具可以用于检测内存泄露问题，使用起来也很简单）。内存泄漏经常发生在已发行的产品中，原因很简单，使用简单的功能测试，几乎发现不了内存泄漏问题，因为在产品完成之前测试没对产品进行足够多的使用。内存泄漏通常要求操作要重复非常多的次数以使内存消耗达到能引起注意的程度。尽管与其它编程语言（如 C/C++）相比，Java 程序更难引入内存泄漏错误，但只要程序仍保持着对对象的引用，该对象仍有可能被实例化并且它占用的内存永远不会被释放。
- **并发与同步（Concurrency and Synchronization）：**性能测试在查找并发性问题上非常出众，这是因为在任何一个测试生命周期中，它都应用了许多不同的代码路径和定时条件。一般的规则是，性能测试运行的时间越长，涉及并应用的代码路径组合和定时条件就越多。当然，这也的确使得这些问题很难再现（错误可以在 5 分钟或 5 天后发生）。死锁、线程泄漏以及任何一般的同步问题通常只能在性能测试阶段被检测出来。这些类型的问题很难通过执行单元测试来发现。开发人员不会一直考虑他或她的代码将与其他地方的代码（在执行单元测试时这些代码可能还没写出来，内存泄漏问题也类似）进行交互。

（4）性能测试阶段

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相应的工具（在第 3 部分讲述）进行测试和评估；

通常，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。不过如果许多单元模块的性能不佳，集成起来的系统性能会好吗？答案是很明白的–一般都会更家糟糕。因此要从开发的早期阶段就关注和实施性能测试。

2. 性能测试的步骤

应用系统的性能测试通常有如下步骤：

（1）制定目标和分析系统

1）制定目标

- 确定客户需求和期望

在需求分析和文档的支持下，需要对软件系统上的用户业务使用情况进行分析，提出我们所关注的性能测试需求，并告知业务人员。让业务人员来判断我们的性能需求是否能满足客户的真实要求。

例如在与用户和系统分析人员进行了充分的讨论后，确定系统的响应时间，这是用户最重视的性能体验。确定响应时间的基本原则是：

2/5/10 原则：

- 2：2 秒钟用户会觉得是一个很好的体验。
- 5：5 秒钟用户可能会觉得差了一点，还行，比较好
- 10：10 秒钟是用户所能承受的最大极限。

可以根据不同的网络环境，确定用户所能承受的响应时间极限定（如 12～15 秒）。

- 确定实际业务需求

主要是确定用户的业务请求分布等：主要业务请求、平均日交易量、年交易量、峰值交易量等等。

下面是一个性能应用请求分布的示例（表 1）：

主要 业务请求	每单位日平均交易量	单位数	平均日 交易量	日峰值 交易量	年交易量 (250 工作日)	备注
A 业务	165	6	990	1287	247500	
B 业务	120	6	720	936	180000	
……						

- 确定系统需求

这里的主要工作是分析系统的性能需求、确定合理性能目标。可以从以下几个方面进行考虑。

- 普通数据容量负载时，在不同的用户数时的并发业务响应时间、随机业务响应时间；

- 大数据容量负载时，在不同的用户数时的并发业务响应时间、随机业务响应时间；
- 未来若干年数据容量负载时，在不同的用户数时的并发业务响应时间、随机业务响应时间；

下面是一个分析系统的性能需求、确定合理性能目标的示例：

- 并发业务响应时间

N 个用户在登录系统做业务前,设置集合点,待并发用户到齐后,同时做预定的业务，参见表 2。

表 2：并发业务响应时间

单位数	人数	并发用户数	业务	响应时间
6	480	10	A 业务	<2 秒
		200	A 业务	<5 秒
12	1920	10	A 业务	<2 秒
		300	A 业务	<5 秒
24	4800	10	A 业务	<2 秒
		300	A 业务	<5 秒
		600	A 业务	<8 秒
注释：A 业务数据容量是 20 万条记录				

- 随机业务响应时间

N 人随机登录系统,部分用户随机做 A 业务,部分用户随机做 B 业务、C 业务，参见表 3。

表 3：随机业务响应时间

单位数	人数	随机用户数	业务	响应时间	峰值响应时间
6	480	100	A 业务		
		200	B 业务	<3 秒	<5 秒
12	1920	300	A 业务		
		400	B 业务	<5 秒	<8 秒
		500	C 业务		
24	4800	500	A 业务		
		800	B 业务	<5 秒	<8 秒
		1200	C 业务		
注释：A 业务数据容量是 20 万条记录、B、C 业务数据容量均为 10 条记录。					

2）分析系统

- 系统类别

系统类别：分清系统类别是我们掌握什么样的技术的前提，掌握相应技术做性能测试才可能成功。例如：系统类别是 bs 结构,需要掌握 http 协议，java，C#，html 等技术 。或者是 cs 结构，可能要了解操作系统，winsock，com 等。所以甄别系统类别对于我们来说很重要。

- 系统构成

硬件设置，操作系统设置是性能测试的制约条件，一般性能测试都是利用测试工具模仿大量的实际用户操作，系统在超负荷情形下运作。不同的系统构成性能测试就会得到不同的结果。

- 系统功能

系统功能指系统提供的不同子系统，例如办公管理系统中的公文子系统，会议子系统等，系统功能是性能测试中要模拟的环节，了解这些是必要的。

（2）制定性能测试计划

主要任务是规划性能测试所需的测试环境、测试软件、相关技术与测试工具的选择，测试的人员组织，测试目标、测试日程等；

1）测试环境

测试环境主要是描述进行性能测试时所使用的环境，测试环境示例如表 4 所示：

表 4：测试环境

服务器	机型	数量	CPU	内存	存储	网络连接	操作系统	系统软件	应用系统	备注
数据库服务器	IBM RS6000 650	2	8	16G	SAN	1000M	AIX	Oracle 9.2	数据库	
应用服务器	IBM RS6000 630	1	2	4G	SAN	1000M	AIX	WebLogic 8.1	应用系统	

2）测试软件

描述需要测试的应用软件，其特点、采用的技术、架构等。

3）相关技术与测试工具的选择

要在对各种性能测试工具和性能测试的任务（是在单元阶段进行的性能测试还是在系统已经集成后进行的性能测试，如在单元阶段进行性能测试，那么没有必要采用 LoadRunner 等昂贵的商品化软件，效果也未必好，在后面我们会讲述这个问题，第 3 部分也会介绍适合这个阶段使用的工具）进行评估的基础上，选择符合现有软件架构和开发阶段的性能测试工具。性能测试是通过工具，模拟大量用户操作，对系统增加负载。所以需要掌握一定的工具知识才能进行性能测试。大家都知道性能测试工具一般通过 winsock,http 等协议纪录用户操作。而协议选择是基于软件的系统架构实现（web 一般选择 http 协议,cs 选择 winsock 协议），不同的性能测试工具，脚本语言也不同，比如 rational robot 中 vu 脚本用类 c 语言实现。

开展性能测试需要对各种性能测试工具进行评估，因为每一种性能测试工具都有自身的特点，只有经过工具评估，才能选择符合现有软件架构和开发阶段的性能测试工具。确定测试工具后，需要组织测试人员进行工具的学习，培训相关技术。参见表 5。主流黑盒测试工具列表参见表 6。

表 5

用途	工具	生产厂商/自产	版本

表 6

工具名	公司名	官方站点
WAS	M$	http://www.micro$oft.com
LoadRunner	Mercury	http://www.mercuryinteractive.com
Astra Quicktest	Mercury	http://www.mercuryinteractive.com
Qaload	Compuware	http://www.empirix.com
TeamTest:SiteLoad	IBM Rational	http://www.rational.com
Webload	Radview	http://www.radview.com
Silkperformer	Segue	http://www.segue.com
e-Load	Empirix	http://www.empirix.com
OpenSTA	OpenSTA	http://www.opensta.com

4）测试的人员组织

不同的阶段、不同项目、不同的组织方式，使得参与性能测试的人员可能会有比较大的差异。从测试角色来看需求分析人员、案例设计人员、测试执行人员、测试分析人员；从人员组成的角度，应该有开发人员、测试人员、用户代表、系统人员等

针对性能测试不同的关注点所对应的相关人员参见表 7：

表 7

关注问题	相关人员
确定性能需求的解决方法	项目经理、业务人员、DBA、系统部人员、性能测试工程师
确立性能测试目标的原则	项目经理
不同阶段的性能测试目标	项目经理 r
性能测试方案的确立	性能测试工程师
用例和场景设计	性能测试工程师
设定需要监控的资源	性能测试工程师
性能测试的应用领域	项目经理、性能测试工程师
分析影响性能因素的步骤	设计人员、DBA、性能测试工程师
开发角度性能问题的原因	开发人员、设计人员、DBA、leader
部署阶段	项目经理
维护阶段	项目经理
性能测试的策略	性能测试工程师
各阶段所要进行的性能测试	设计人员、开发人员、项目经理、性能测试工程师
系统的稳定性度量	项目经理、性能测试工程师
性能测试的基本概念	性能测试工程师
响应时间的分解	性能测试工程师
在性能测试中需要注意的问题	性能测试工程师

具体按角色分配人力资源时，可参见表 8 适当地删除或添加角色项。

表 8

角色	所推荐的最少资源（所分配的专职角色数量）	具体职责或注释

5）测试目标

描述本次测试任务的目标，可以从以下几个方面进行考虑：

①确立性能测试目标的原则

- 以“需求”为本

考虑系统需不需要作性能测试，性能测试的内容和范围。不是每个系统都需要做性能测试的

- 测试目标确定的经济性考虑
  - 投入到性能测试的人员是多少？
  - 具备可以确定性能测试需求，制定性能测试方案的人员是多少？可以执行性能测试的人员是多少？
  - 这些人员需要投入多长时间？
  - 所要开发系统的运行环境和设备，这些设备的配置对于性能测试的影响，比如说：tomcat4.1 的应用服务器，它的配置文件缺省的 jvm 的使用空间是 64M，一个机器的内存为 1G，我们将 jvm 的使用空间设置为 512M 对性能测试的影响。
  - 内部的人员无法满足性能测试的要求，通过外聘，采用外聘的方式，公司所能承受的成本是多高。
- 基于风险的测试目标确定
  - 系统如果不做性能测试，会有多大的风险，如果在性能指标上达不到用户的要求会有多大的风险。需要进行评估。
  - 如果做性能测试会有多大的风险，性能测试的投入会有多大，会有多大的风险需要进行评估。

②确定性能测试目标的方法

我们要确定系统的吞吐量和并发用户数的设计目标可以采用以下三种方式：

- 确定在某个特定时间端内，估计系统会有多少用户同时访问
- 在某个特定的时间端内，正在访问系统的用户的典型操作是什么？哪个页面的访问量最大？
- 在某个特定的时间端内，系统需要处理多少种用户场景

这些数据可以在系统服务器的日志文件、TSP 监视数据种找到，也可以通过监视数据库的活动情况来获得。

③不同阶段的性能测试目标

- 设计阶段的性能测试目标

设计阶段的性能测试目标主要为考察系统是否满足预期的性能要求。

- 开发阶段的性能测试目标
  - 将开发阶段的性能测试目标作为对系统进行调优的参考：考虑在每个开发阶段，性能是否能够达到标准，考虑当前阶段的性能瓶颈，及其性能瓶颈出现的原因是在于数据库访问（SQL 语句或者存储过程写的不够好）还是其他的原因。
  - 用性能测试手段发现系统存在的问题：通过模拟真实场景，发现在现场测试中可能存在的问题，比如说：用户数的突然增加，导致的应用程序崩溃，服务器崩溃的问题。
- 部署阶段的性能测试目标

提供部署方案的参考，确定合适的硬件设备，虽然更高的设备可以获取商业上的利益，但应考虑客户的具体情况。

- 系统维护阶段的性能测试目标

考察系统的可扩展性：从系统的视角考虑,在用户数扩大，在业务量增大的情况下，是一个怎样的表现。

6）测试日程

整个测试活动在日程上的安排，参见表 9 的日程安排。

表 9：测试日程示例

测试阶段	测试任务	工作量估计(人日)	人员分配	开始时间	截止时间
第一阶段	方案设计、案例设计	2	卢哲学	2005-08-16	2005-08-17
	方案设计、案例设计 review	2	王安	2005-08-22	2005-08-23
……					

（3）设计测试用例

设计测试用例是在了解软件业务流程的基础上。设计测试用例的原则是受

最小的影响提供最多的测试信息，设计测试用例的目标是一次尽可能的包含多个测试要素。这些测试用例必须是测试工具可以实现的，不同的测试场景将测试不同的功能。因为性能测试不同于功能测试的测试用例，尽可能把性能测试用例设计的复杂，才有可能发现软件的性能瓶颈。因此对于设计用例至少要考虑到以下3个问题：

- 业务复杂，功能众多，如何一次尽可能地包含多个测试要素
- 脚本如何开发
- 场景数据出来后，如何分析

1）场景设计重点

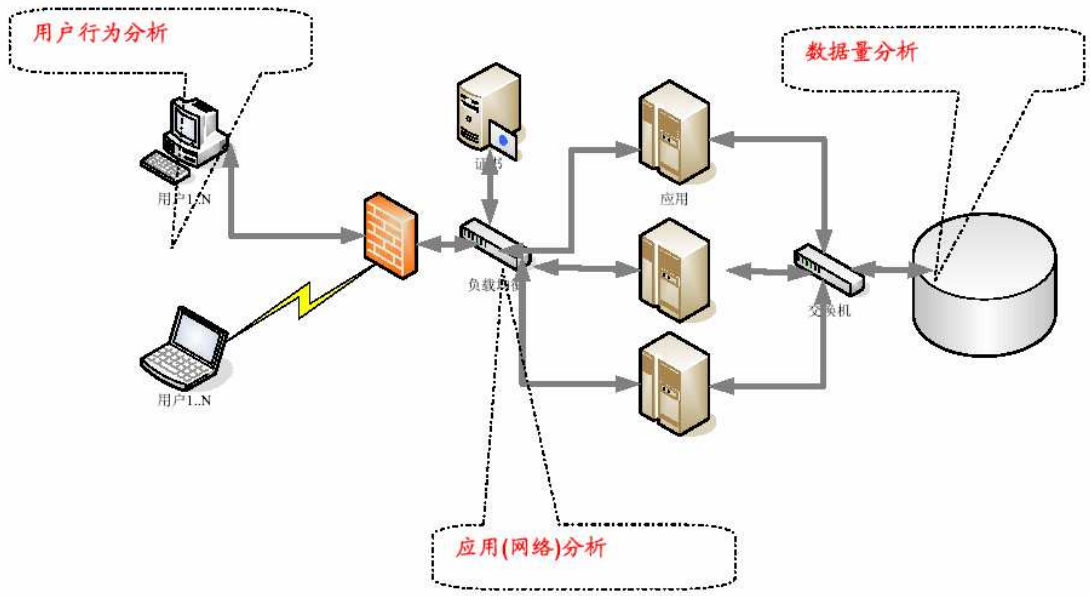

图 1　性能测试场景设计重点

参见图 1，场景设计的重点主要在以下几个方面：

- 场景设计源自需求，而且是实实在在的实际需求，一方面要避免遗漏必要的性能需求，另外一方面也要剔除那些不切实际，看上去“很美”的需求（“幽灵需求”）。
- 系统大致的并发用户数，这些数据可以在系统服务器的日志文件中找到，也可以通过监视数据库的活动情况来获得。
- 用户在时间上的分布，峰值与平均并发。可以在系统服务器的日志文件、TSP 监视数据种找到，也可以通过监视数据库的活动情况来获得

- 用户在空间上的分布，不同模块与用户分布。
- 系统在稳定性上的要求，系统长时间运行不会出现错误的能力，此部分需与业务人员讨论。
- 数据库数据量大小、增长方式、访问方式，此部分需与设计人员、DBA等人员进行讨论。

表 10 给出了一个场景设计的例子：

表 10

序号	分类	项　　目	数据	单位
1	统计数据及经验数据	A：总用户数	1000000	个
2		B：激活用户比例，每天访问用户数占总用户数的比例	50%	
3		C：每个激活用户邮件数	100	封
4		D：每个用户每天收到的信件数	8	封
5		E：每个用户每天发出的信件数	6	封
6		F：系统高峰时间（小时）	4	小时
7		G：高峰时间内收发的邮件数占一天总邮件数	50%	
8		H：每个用户每天收发邮件次数	6	
9		I：每封邮件大小平均为（K）	30	KByte
10		J1：据统计，使用 WEBMAIL 的用户百分比	60%	
		J2：使用邮件客户端软件的用户百分比	38%	
		J3：使用 IMAP 的用户百分比	2%	
		K：平均每通过 web 访问一封信,大约要访问页面数为：	4	个
		L：假定每个页面大小约为	30	KByte
		M：通过本系统向外转送百分比	75%	
		N：发送给本系统的邮件百分比	25%	
		O：系统峰值时 CPU 利用率	60%	

2）用例设计摸板

可以分为以下几种方式来组织进行设计

- 预期性能测试用例

通常系统在设计前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，针对每个指标都要统写多个测试用例来验证是否达到要求，根据测试结果来改进系统的性能。预期性能指标通成以单用户为主。参见表 11

表 11

测试目的 前置条件			
测试需求	测试过程说明	期望的性能（平均值）	实际性能（平均值）
功能 1	场景 1 场景 2 场景 3		

备注：

- 用户并发测试用例

用户并发测试是性能测试最主要的部分，主要是通过增加用户数量来加重系统负担，以检验测试对象能接收的最大用户数来确定功能是否达到要求。裁减表 12。

表 12

测试目的 前提条件				
测试需求	输入（并发用户数）	用户通过率	期望性能（平均值）	实际性能（平均值）
功能 1	50 100 200			
功能 2	50 100 200			

备注：

- 大数据量测试用例

大数据量测试使测试对象处理大量的数据，以确定是否达到了将使软件发生故障的极限。大数据量测试还将确定测试对象在给定时间内能够持续处理的最大负载或工作量，参见表 13。

表 13

测试目的 前提条件				
测试需求	输入（最大数据量）	事务成功率	期望性能（平均值）	实际性能（平均值）
功能 1	10000 第条记录 15000 第条记录 20000 第条记录			
功能 2	10000 第条记录 15000 第条记录 20000 第条记录			
…				

备注：

- 疲劳强度测试用例

强度测试也是性能测试是的一种，实施和执行此类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。强度测试还可用于确定测试对象能够处理的最大工作量，参见表 14

表 14。

测试目的 测试说明			
前提条件		连续运行 8 小时，设置添加 10 用户并发	
测试需求	输入/动作	输出/响应	是否正常运行
功能 1	2 小时		

	4 小时	
	6 小时	
	8 小时	
功能 1	2 小时	
	4 小时	
	6 小时	
	8 小时	

- 负载测试测试用例

负载测试也是性能测试中的一种。在这种测试中，将使测试对象承担不同的工作量，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。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，参见表 15。

表 15

测试目的 前提条件			
测试需求	输入	期望输出	是否正常运行

备注

3）根据场景编写脚本

采用 Jmeter 模拟前端，用户向应用系统发生业务请求。或者，采用 LoadRunner 录制用户与应用系统之间发生交互过程的脚本。

脚本开发示例—业务上下关联

```
//href='/wm/mail/read.htmlsessionid=0fe692e3f08bc1f6ef924188e70a2c137&uid=10&msgid=10&mbox=ur.u3'>高效、安全、稳定--To: User3</a>
web_reg_save_param("MailList", "LB/ic=href=\'/wm/mail/read.html?sessionid=", "RB=\'>", "Ord=All", LAST);
跟据上一个业务拼下一个URL
// check how much mails in the list
        MailCnt=atoi(lr_eval_string("{MailList_count}"));
        if (MailCnt > 2)
                for (cnt=1; cnt<4; cnt++) {
                        sprintf(mail_url, "URL=http://10.0.68.200/wm/mail/read.html?sessionid={MailList_%d}", cnt);
                        strcpy(mail_id_url, lr_eval_string(mail_url));
//                      lr_message(mail_id_url);

                        lr_start_transaction("HTTP_READ_MAIL");
                }
```

一个测试脚本常常包含多个事务，即使多个进程同时运行同一个脚本，也难以保证脚本内的某个事务同时运行，这将影响对这个事务的响应时间的测试。为了解决这个问题，需要设置并发点，先运行到并发点的进程将等待，当所有进程都运行到并发点时，进行释放，使所有的进程同时运行同一个事务，这样就可以测定与实际比较接近的响应时间，可以设置多个并发点，对多个事务的响应时间进行测试。

可以通过进程的挂起和唤醒操作实现。当到并发点时，挂起进程，当所有进程到达并发点时，释放进程。在发现时，常常也设置超时设置，当到达并发点的进程时，在等待一定时间后，可以自己唤醒。即使个别进程由于别的原因不能运行到并发点，还要保证其余运行到并发点的进程可以继续运行。

（4）执行测试用例

通过性能测试工具运行测试用例。同一环境下作的性能测试得到的测试结果是不准确的，所以在运行这些测试用例的时候，需要用不同的测试环境，不同的机器配置上运行。在执行测试用例时，应掌握以下原则：

- 测试前，要确认系统级的关键参数已经基本配置正确（例如：数据库、WEB 容器、线程池、JDBC 连接池、对象池、JVM、操作系统、应用系

统等配置）；

- 测试前，要确保测试脚本的业务功能运行正确。
- 测试前，清空所有应用日志、调高错误日志的输出级别（Error 级），必要时在每次测试前重启应用服务和数据库应用服务；
- 调整系统参数时，一次只调整一个，不要同时调整多个，并记录调整前后的系统变化。
- 优先测试基线案例。

以压力测试为例，当准备好测试案例与测试脚本后，就可以运行测试程序，进行压力测试。有时，还需要根据运行情况，增加或减少并发的进程。在测试运行时，要对一些结果进行自动记录，它们一般都是一些与时间有关的数据。运行压力测试时的另一个任务就是监测系统资源，可以使用一些工具来监测并记录资源使用情况。

监测的对象有：网络阻塞情况；主机 CPU 使用情况；内存使用情况；缓存使用情况；数据库系统中的数据锁；回滚段；重做日志缓冲区等。

监测的结果包括图象与数据文件，并且图象可以实时显示，也可以在压力测试运行结束后分析。可以使用操作系统、数据库系统软件附带的监测工具。

通过资源监测的结果很容易找到系统的瓶颈，并对产生瓶颈的资源进行调整、优化。

客户端与服务器端的区分，有些工具可测试客户端和服务器端的资源，如 compuware，有些不能。监测系统资源参见图 2、图 3、图 4。

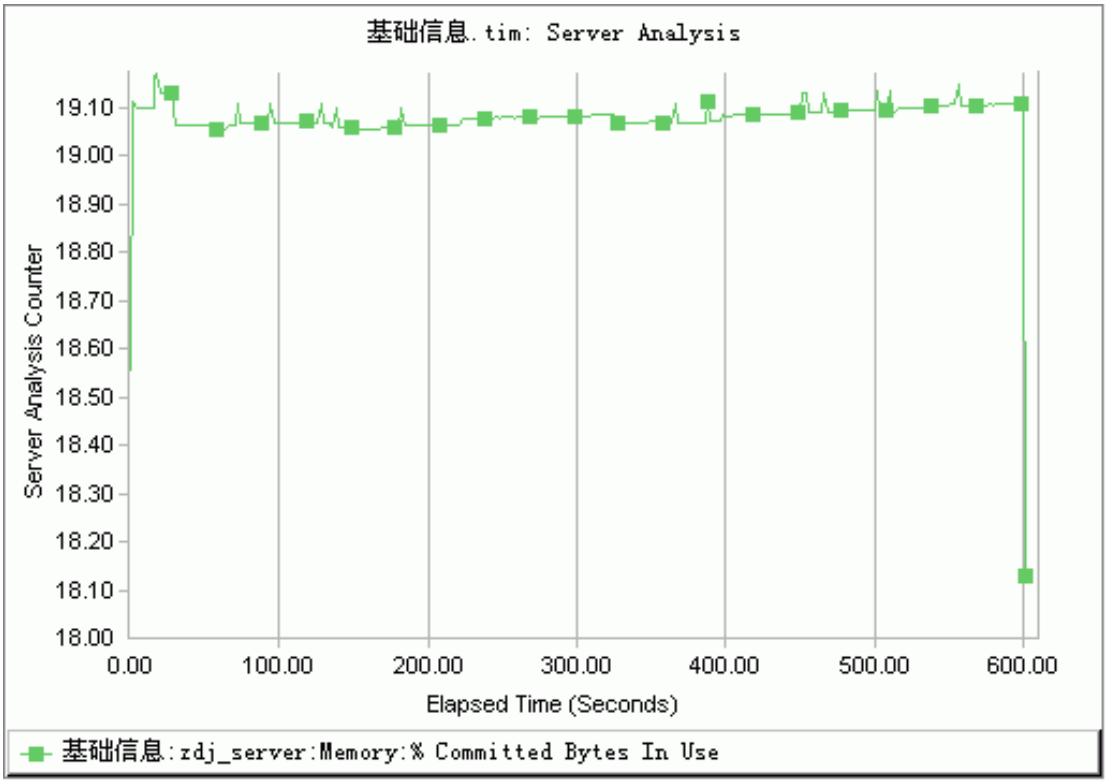

图 2　内存使用百分比

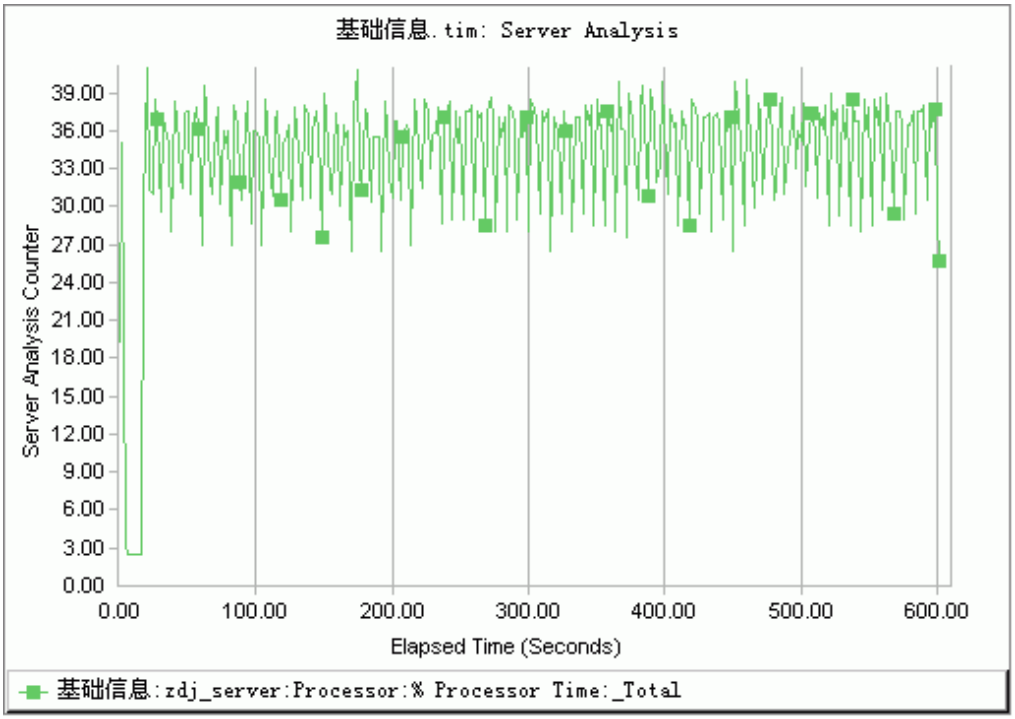

图 3　处理器占用情况百分比

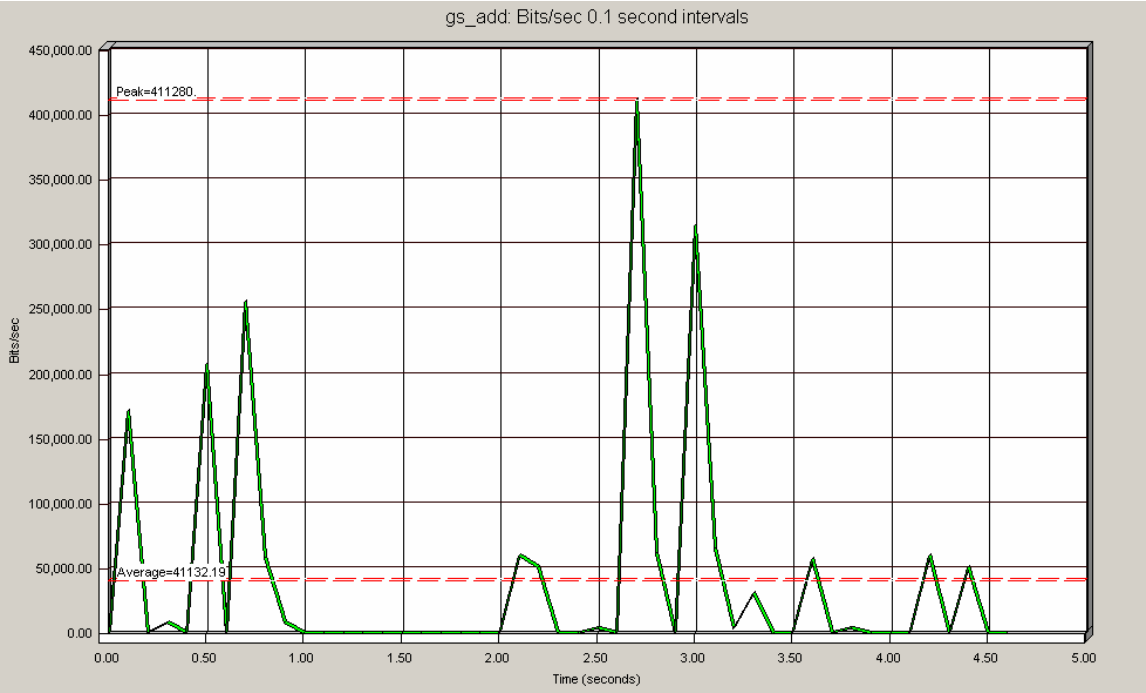

图 4　网络传输速度 bits/sec

（5）分析测试结果

分析测试结果在整个测试过程中最重要的过程之一，通过分析可以发现应用程序的各种功能性缺陷。当运行完某个测试脚本后，会产生一个测试报告，从这个测试报告中我们能发现应用程序的性能缺陷，能看到实际结果和期望结果之间的差异，以及在测试过程中产生的各类检测结果等。

1）性能测试结果分析原则

在进行性能分析时，我们应当遵循以下的原则：

- 把事实与推测分开，总是用实际的证据来证明你的推测；
- 在没有足够证据之前，不对程序进行优化；
- 优先验证简单的假设；
- 日志文件中没有错误不代表真的没有错误；
- 从系统到应用、从外到内进行层层剥离，缩小范围。
  - ◆ 确认是系统级问题还是应用级问题；
  - ◆ 确认是否外部系统问题（如密码鉴权问题、EJB 问题等）；
  - ◆ 确认是应用程序问题还是数据库问题。
- 范围缩小后，再分割成多个小单元，对每个小单元进行轮番压力测试，

来证明或者否定是那个单元引起性能问题。例如在查找瓶颈时按以下顺序，由易到难。

服务器硬件瓶颈-〉网络瓶颈（对局域网，可以不考虑）-〉服务器操作系统瓶颈（参数配置）-〉中间件瓶颈（参数配置，数据库，web 服务器等）-〉应用瓶颈（SQL 语句、数据库设计、业务逻辑、算法等）

注：以上过程并不是每个分析中都需要的，要根据测试目的和要求来确定分析的深度。对一些要求低的，我们分析到应用系统在将来大的负载压力（并发用户数、数据量）下，系统的硬件瓶颈在哪儿就够了。

2）常见性能问题成因

参见表 15

表 15

性能问题	描述	问题特征	成因
线性内存泄漏	每单位(每事务、每用户等)泄漏造成内存随着时间或负载线性增长。这会随着时间或负载增长降低系统性能。只有重启才有可能恢复。	随着时间越来越慢 随着负载越来越慢	虽然可能有多种外部原因，但最典型的是与资源泄漏有关(例如，没有清理不能被 GC 回收的大对象，引起大对象不断积累)。
指数方式内存泄漏	双倍增长的内存泄漏造成系统内存消耗表现为时间的指数曲线	随着时间越来越慢 随着负载越来越慢	通常是由于向集合(Vector，HashMap) 中加入永远不删除的元素造成的。
糟糕的编码：无限循环	线程在 while(true) 语句以及类似的语句里阻塞。	可以预见的锁定	需要对循环进行大刀阔斧的删剪。
资源泄漏	JDBC 语句，CICS 事务网关连接，以及类似的东西被泄漏了，造成桥接层和后端系统的影响。	随着时间越来越慢 可以预见的锁定 突然混乱	通常情况下，这是由于遗漏了 finally 块，或者更简单点，就是忘记用 close() 关闭外部资源的对象所造成的。
外部瓶颈问题	后端或者其他外部系统（如鉴权）越来越慢，同样减缓了应用服务器和应用程序	持续缓慢 随着负载越来越慢	咨询专家（负责的第三方或者系统管理员），获取解决外部瓶颈问题的方法。
外部系统	应用程序通过太大或太多的请求滥用后端系统。	持续缓慢 随着负载越来越慢	清除冗余的工作请求 ，批量处理相似的工作请求，把大的请求分解成若干个更小的请求，调整工作请求或后端系统(例如，公共查询关键字的索引)等。
糟糕的编码：CPU 密集的组	一些糟糕的代码进行交互处理时，就挂起了 CPU，把吞吐速度减慢	持续缓慢 随着负载越来越慢	典型的解决方案就是数据高速缓存。

件	到爬行的速度。		
中间层问题	实现得很糟糕的桥接层(如 JDBC 驱动程序、数据库连接池管理)，由于对数据和请求不断的排列、解除排列，从而把所有通过它的流量减慢到爬行速度。这个毛病在早期阶段很容易与外部瓶颈混淆。	持续缓慢 随着负载越来越慢	检查桥接层和外部系统的版本兼容性。如果有可能，评估不同的桥接供应商。如果重新规划架构，有可能完全不需要桥接。
内部资源瓶颈：过度使用或分配不足	内部资源(线程池、对象池)变得稀缺。是在正确使用的情况下加大负载时出现过度使用还是因为泄漏？	随着负载越来越慢 零星的挂起或异常错误	分配不足：根据预期的最大负载提高池的最大尺寸。过度使用：请参阅外部系统的过度使用。
不停止的重试	这包括对失败请求连续的(或者在极端情况下无休止的)重试。	可以预见的锁定 突然混乱	可能是后端系统完全当机，或者网络连接中断。
线程：阻塞点	线程遇到同步阻塞，造成交通阻塞。	随着负载越来越慢 零星的挂起或异常错误 可以预见的锁定 突然混乱	可能同步是不必要的(只要重新设计)，或者比较外在的锁定策略(例如，读/写锁)也许会有帮助。
线程：死锁/活动锁	这是基本的“获得顺序”的问题。	突然混乱	“获得顺序”的算法不合理。

3）监控指标数据分析

① 最大并发用户数：

应用系统在当前环境（硬件环境、网络环境、软件环境（参数配置））下能承受的最大并发用户数。

在方案运行中，如果出现了大于 3 个用户的业务操作失败，或出现了服务器 shutdown 的情况，则说明在当前环境下，系统承受不了当前并发用户的负载压力，那么最大并发用户数就是前一个没有出现这种现象的并发用户数。

如果测得的最大并发用户数到达了性能要求，且各服务器资源情况良好，业务操作响应时间也达到了用户要求，那么 OK。否则，再根据各服务器的资源情况和业务操作响应时间进一步分析原因所在。

② 业务操作响应时间：

分析方案运行情况应从平均事务响应时间图和事务性能摘要图开始。使用“事务性能摘要”图，可以确定在方案执行期间响应时间过长的事务。

细分事务并分析每个页面组件的性能。查看过长的事务响应时间是由哪些

页面组件引起的？问题是否与网络或服务器有关？

如果服务器耗时过长，请使用相应的服务器图确定有问题的服务器度量并查明服务器性能下降的原因。如果网络耗时过长，请使用“网络监视器”图确定导致性能瓶颈的网络问题

③ 服务器资源监控指标

内存：

UNIX 资源监控中指标内存页交换速率（Paging rate），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。

Windows 资源监控中，如果 Process\Private Bytes 计数器和 Process\Working Set 计数器的值在长时间内持续升高，同时 Memory\Available bytes 计数器的值持续降低，则很可能存在内存泄漏。

内存资源成为系统性能的瓶颈的征兆：

- 很高的换页率(high pageout rate);
- 进程进入不活动状态;
- 交换区所有磁盘的活动次数可高;
- 可高的全局系统 CPU 利用率;
- 内存不够出错(out of memory errors)

处理器：

UNIX 资源监控（Windows 操作系统同理）中指标 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。如果服务器专用于 SQL Server,可接受的最大上限是 80-85% 合理使用的范围在 60%至 70%。

Windows 资源监控中，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：

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)

磁盘 I/O：

UNIX 资源监控（Windows 操作系统同理）中指标磁盘交换率（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。

Windows 资源监控中，如果 Disk Time 和 Avg.Disk Queue Length 的值很高，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 ：

- 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)

数据库服务器：

SQL Server 数据库：

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。

Oracle 数据库：

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’ ;

4 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)’

注：上述 SQL Server 和 Oracle 数据库分析，只是一些简单、基本的分析，特别是 Oracle 数据库的分析和优化，是一门专门的技术，进一步的分析可查相关资料。

在分析出影响系统的性能之后还要对系统或应用程序进行调优，由于性能优化对原有系统进行了相应的修改，所以必须进行相应的回归测试，以检查修改的正确性和有效性。这两个都不是本教程的内容，将另文叙述。

3. 附件--性能测试参数说明

Memory: 内存使用情况可能是系统性能中最重要的因素。如果系统“页交换”频繁，说明内存不足。“页交换”是使用称为“页面”的单位，将固定大小的代码和数据块从 RAM 移动到磁盘的过程，其目的是为了释放内存空间。尽管某些页交换使 Windows 2000 能够使用比实际更多的内存，也是可以接受的，但频繁的页交换将降低系统性能。减少页交换将显著提高系统响应速度。要监视内存不足的状况，请从以下的对象计数器开始：

Available Mbytes:可用物理内存数. 如果 Available Mbytes 的值很小（4 MB 或更小），则说明计算机上总的内存可能不足，或某程序没有释放内存。

page/sec: 表明由于硬件页面错误而从磁盘取出的页面数，或由于页面错误而写入磁盘以释放工作集空间的页面数。一般如果 pages/sec 持续高于几百，那么您应该进一步研究页交换活动。有可能需要增加内存，以减少换页的需求（你可以把这个数字乘以 4k 就得到由此引起的硬盘数据流量）。Pages/sec 的值很大不一定表明内存有问题，而可能是运行使用内存映射文件的程序所致。

page read/sec:页的硬故障，page/sec 的子集，为了解析对内存的引用，必须读取页文件的次数。阈值为>5. 越低越好。大数值表示磁盘读而不是缓存读。

由于过多的页交换要使用大量的硬盘空间，因此有可能将导致将页交换内存不足与导致页交换的磁盘瓶径混淆。因此，在研究内存不足不太明显的页交换的原因时，您必须跟踪如下的磁盘使用情况计数器和内存计数器：

Physical Disk\ % Disk Time

Physical Disk\ Avg.Disk Queue Length

例如，包括 Page Reads/sec 和% Disk Time 及 Avg.Disk Queue Length。如果页面读取操作速率很低，同时% Disk Time 和 Avg.Disk Queue Length 的值很高，则可能有磁盘瓶径。但是，如果队列长度增加的同时页面读取速率并未降低，则内存不足。

要确定过多的页交换对磁盘活动的影响，请将 Physical Disk\ Avg.Disk sec/Transfer 和 Memory\ Pages/sec 计数器的值增大数倍。如果这些计数器的计数结果超过了 0.1，那么页交换将花费百分之十以上的磁盘访问时间。如果长时间发生这种情况，那么您可能需要更多的内存。

Page Faults/sec:每秒软性页面失效的数目（包括有些可以直接在内存中满足而有些需要从硬盘读取）较 page/sec 只表明数据不能在内存的指定工作集中立即使用。

Cache Bytes：文件系统缓存（File System Cache），默认情况下为 50%的可用物理内存。如 IIS5.0 运行内存不够时，它会自动整理缓存。需要关注该计数器的趋势变化。

如果您怀疑有内存泄露，请监视 Memory\ Available Bytes 和 Memory\ Committed Bytes，以观察内存行为，并监视您认为可能在泄露内存的进程的 Process\Private Bytes、Process\Working Set 和 Process\Handle Count。如果您怀疑是内核模式进程导致了泄露，则还应该监视 Memory\Pool Nonpaged Bytes、Memory\ Pool Nonpaged Allocs 和 Process(process_name)\ Pool Nonpaged Bytes。

Pages per second :每秒钟检索的页数。该数字应少于每秒一页。

Process：

%Processor Time: 被处理器消耗的处理器时间数量。如果服务器专用于sql server,可接受的最大上限是80-85%。

Page Faults/sec:将进程产生的页故障与系统产生的相比较，以判断这个进程对系统页故障产生的影响。

Work set: 处理线程最近使用的内存页，反映了每一个进程使用的内存页的数量。如果服务器有足够的空闲内存，页就会被留在工作集中，当自由内存少于一个特定的阈值时，页就会被清除出工作集。

Inetinfo:Private Bytes:此进程所分配的无法与其它进程共享的当前字节数量。如果系统性能随着时间而降低,则此计数器可以是内存泄漏的最佳指示器。

Processor：监视“处理器”和“系统”对象计数器可以提供关于处理器使用的有价值的信息，帮助您决定是否存在瓶颈。

%Processor Time: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。

%User Time:表示耗费 CPU 的数据库操作，如排序，执行 aggregate functions 等。如果该值很高，可考虑增加索引，尽量使用简单的表联接，水平分割大表格等方法来降低该值。

%Privileged Time：（CPU 内核时间）是在特权模式下处理线程执行代码所花时间的百分比。如果该参数值和"Physical Disk"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。另外设置 Tempdb in RAM，减低"max async IO"，"max lazy writer IO"等措施都会降低该值。

此外，跟踪计算机的服务器工作队列当前长度的 Server Work Queues\Queue Length 计数器会显示出处理器瓶颈。队列长度持续大于 4 则表示可能出现处理器拥塞。此计数器是特定时间的值，而不是一段时间的平均值。

% DPC Time:越低越好。在多处理器系统中，如果这个值大于 50%并且 Processor:% Processor Time 非常高，加入一个网卡可能会提高性能，提供的网络已经不饱和。

Thread：

ContextSwitches/sec: (实例化 inetinfo 和 dllhost 进程) 如果你决定要增加线程字节池的大小，你应该监视这三个计数器（包括上面的一个）。增加线程数可能会增加上下文切换次数，这样性能不会上升反而会下降。如果十个实例的上下文切换值非常高，就应该减小线程字节池的大小。

Physical Disk:

%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。如果三个计数器都比较大，那么硬盘不是瓶颈。如果只有%Disk Time 比较大，另外两个都比较适中，硬盘可能会是瓶颈。在记录该计数器之前，请在 Windows 2000 的命令行窗口中运行 diskperf -yD。若数值持续超过 80%，则可能是内存泄漏。

Avg.Disk Queue Length: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。该值应不超过磁盘数的 1.5~2 倍。要提高性能，可增加磁盘。注意：一个 Raid Disk 实际有多个磁盘。

Average Disk Read/Write Queue Length:指读取(写入)请求(列队)的平均数。

Disk Reads(Writes)/s: 物理磁盘上每秒钟磁盘读、写的次数。两者相加，应小于磁盘设备最大容量。

Average Disksec/Read: 指以秒计算的在此盘上读取数据的所需平均时间。

Average Disk sec/Transfer:指以秒计算的在此盘上写入数据的所需平均时间。

Network Interface：

Bytes Total/sec :为发送和接收字节的速率，包括帧字符在内。判断网络连接速度是否是瓶颈，可以用该计数器的值和目前网络的带宽比较

SQLServer 性能计数器：

Access Methods(访问方法) 用于监视访问数据库中的逻辑页的方法。

Full Scans/sec(全表扫描/秒) 每秒不受限的完全扫描数。可以是基本表扫描或全索引扫描。如果这个计数器显示的值比 1 或 2 高，应该分析你的查询以确定是否确实需要全表扫描，以及 S Q L 查询是否可以被优化。

Page splits/sec(页分割/秒)由于数据更新操作引起的每秒页分割的数量。

Buffer Manager(缓冲器管理器)：监视 Microsoft® SQL Server? 如何使用： 内存存储数据页、内部数据结构和过程高速缓存；计数器在 SQL Server 从磁盘读取数据库页和将数据库页写入磁盘时监视物理 I/O。监视 SQL Server 所使用的内存和计数器有助于确定：是否由于缺少可用物理内存存储高速缓存中经常访问的数据而导致瓶颈存在。如果是这样，SQL Server 必须从磁盘检索数据。

是否可通过添加更多内存或使更多内存可用于数据高速缓存或 SQL Server 内部结构来提高查询性能。

SQL Server 需要从磁盘读取数据的频率。与其它操作相比，例如内存访问，物理 I/O 会耗费大量时间。尽可能减少物理 I/O 可以提高查询性能。

Page Reads/sec：每秒发出的物理数据库页读取数。这一统计信息显示的是在所有数据库间的物理页读取总数。由于物理 I/O 的开销大，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，使开销减到最小。

Page Writes/sec (.写的页/秒) 每秒执行的物理数据库写的页数。

Buffer Cache Hit Ratio. 在“缓冲池”（Buffer Cache/Buffer Pool）中没有被读过的页占整个缓冲池中所有页的比率。可在高速缓存中找到而不需要从磁盘中读取的页的百分比。这一比率是高速缓存命中总数除以自 SQL Server 实例启动后对高速缓存的查找总数。经过很长时间后，这一比率的变化很小。由于从高速缓存中读数据比从磁盘中读数据的开销要小得多，一般希望这一数值高一些。通常，可以通过增加 SQL Server 可用的内存数量来提高高速缓存命中率。计数器值依应用程序而定，但比率最好为 90% 或更高。增加内存直到这一数值持续高于 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。

Lazy Writes/sec(惰性写/秒)惰性写进程每秒写的缓冲区的数量。值最好为 0。

Cache Manager(高速缓存管理器) 对象提供计数器，用于监视 Microsoft® SQL Server? 如何使用内存存储对象，如存储过程、特殊和准备好的 Transact-SQL 语句以及触发器。

Cache Hit Ratio(高速缓存命中率，所有 Cache”的命中率。在 SQL Server 中，Cache 可以包括 Log Cache，Buffer Cache 以及 Procedure Cache，是一个总体的比率。）高速缓存命中次数和查找次数的比率。对于查看 SQL Server 高速缓存对于你的系统如何有效，这是一个非常好的计数器。如果这个值很低，持续低于 80%，就需要增加更多的内存。

Latches(闩) 用于监视称为闩锁的内部 SQL Server 资源锁。监视闩锁以明确用户活动和资源使用情况，有助于查明性能瓶颈。

Average Latch Wait Ti m e （ m s ）(平均闩等待时间(毫秒)) 一个 SQL Server 线程必须等待一个闩的平均时间，以毫秒为单位。如果这个值很高，你可能正经历严重的竞争问题。

Latch Waits/sec (闩等待/秒) 在闩上每秒的等待数量。如果这个值很高，表明你正经历对资源的大量竞争。

Locks(锁) 提供有关个别资源类型上的 SQL Server 锁的信息。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。例如，如果一个排它(X) 锁被一个事务加在某一表的某一行上，在这个锁被释放前，其它事务都不可以修改这一行。尽可能少使用锁可提高并发性，从而改善性能。可以同时监视 Locks 对象的多个实例，每个实例代表一个资源类型上的一个锁。

Number of Deadlocks/sec(死锁的数量/秒) 导致死锁的锁请求的数量

Average Wait Time(ms) (平均等待时间(毫秒)) 线程等待某种类型的锁的平均等待时间

Lock Requests/sec(锁请求/秒) 每秒钟某种类型的锁请求的数量。

Memory manager:用于监视总体的服务器内存使用情况，以估计用户活动和资源使用，有助于查明性能瓶颈。监视 SQL Server 实例所使用的内存有助于确定：

是否由于缺少可用物理内存存储高速缓存中经常访问的数据而导致瓶颈存在。如果是这样，SQL Server 必须从磁盘检索数据。

是否可以通过添加更多内存或使更多内存可用于数据高速缓存或 SQL Server 内部结构来提高查询性能。

Lock blocks:服务器上锁定块的数量，锁是在页、行或者表这样的资源上。不希望看到一个增长的值。

Total server memory：sql server 服务器当前正在使用的动态内存总量.

监视 IIS 需要的一些计数器

Internet Information Services Global:
File Cache Hits %、File CacheFlushes、File Cache Hits

File Cache Hits %是全部缓存请求中缓存命中次数所占的比例，反映了 IIS 的文件缓存设置的工作情况。对于一个大部分是静态网页组成的网站，该值应该保持在 80%左右。而 File Cache Hits 是文件缓存命中的具体值，File CacheFlushes 是自服务器启动之后文件缓存刷新次数，如果刷新太慢，会浪费内存；如果刷新太快，缓存中的对象会太频繁的丢弃生成，起不到缓存的作用。通过比较 File Cache Hits 和 File Cache Flushes 可得出缓存命中率对缓存清空率的比率。通过观察它两个的值，可以得到一个适当的刷新值（参考 IIS 的设置 ObjectTTL 、MemCacheSize 、MaxCacheFileSize）

Web Service:

Bytes Total/sec:显示 Web 服务器发送和接受的总字节数。低数值表明该 IIS 正在以较低的速度进行数据传输。

Connection Refused：数值越低越好。高数值表明网络适配器或处理器存在瓶颈。

Not Found Errors：显示由于被请求文件无法找到而无法由服务器满足的请求数（HTTP 状态代码 404）