淘宝性能测试白皮书

淘宝性能测试团队

版本号：V0.3

发布时间：2009-09-01

目录

引言

淘宝网自创立以来，除了对功能要求很高以外，对性能要求也越来越高。从最初的系统框架性能测试、TOP-API 接口性能测试，到现在的 web 应用性能测试，进军无线性能测试领域，淘宝性能测试在不断向前发展，横向、纵向都在不断深入、拓宽，不断创新。

经过五彩石项目对淘宝的整体应用重构之后，淘宝网形成了以四个中心为应用基础的分布式架构体系。而分布式网站的性能，很大程度上决定了网站的竞争优势。但是，一个应用的性能由多方面因素决定，这样就加大了对性能测试和性能调优的难度，也扩大了性能测试的广度，这是一个挑战。专业的测试需要专业的团队， 我们的团队也应运而生。

本性能测试白皮书旨在以理论指导实践， 以实践修正理论，将会从以下几个方面介绍和分析淘宝的性能测试：性能测试指标、淘宝性能测试模型、性能测试策略、性能测试类型、性能测试执行方法、性能监控和性能分析、性能测试通过标准，以及性能测试流程和文件模版。同时，也是让更多的人更好的了解淘宝性能测试和性能调优，参与性能测试，共同将淘宝网做的更大、更强、更稳定，并期望淘宝的性能测试成为引领电子商务性能测试业界的标准。

性能测试指标

Vuser虚拟用户

Virtual user，模拟真实业务逻辑步骤的虚拟用户，虚拟用户模拟的操作步骤都被记录在虚拟用户脚本里。Vuser 脚本用于描述 Vuser 在场景中执行的操作。

Transaction事务

事务是性能测试脚本的一个重要特性。要度量服务器的性能，需要定义事务，每个事务都包含事务开始和事务结束标记。事务用来衡量脚本中一行代码或多行代码的执行所耗费的时间。可以将事务开始放置在脚本中某行或者多行代码的前面，将事务结束放置在该行或者多行代码的后面，在该脚本的虚拟用户运行时，这个事务将衡量该行或者多行代码的执行花费了多长时间。

TPS每秒事务数

(Transaction Per Second)每秒钟系统能够处理的交易或事务的数量，它是衡量系统处理能力的重要指标。TPS 是 LoadRunner 中重要的性能参数指标。

PV Page View

PV 是 Page View 的缩写。用户通过浏览器访问页面，对应用服务器产生的每一次请求，记为一个 PV。淘宝性能测试环境下，将这个概念做了延伸，系统真实处理的一个请求，视为一个 PV。即，PV 的概念也适用于接口。

Peak PV 高峰Page View

即 PV 峰值，指一天中 PV 数达到的最高峰。

Concurrency并发

并发分为狭义和广义两类。

狭义的并发，即所有的用户在同一时刻做同一件事情或操作，这种操作一般针对同一类型的业务;或者所有用户进行完全一样的操作,目的是测试数据库和程序对并发操作的处理。

广义的并发，即多个用户对系统发出了请求或者进行了操作，但是这些请求或操作可以是不同的。对整个系统而言，仍然有很多用户同时进行操作。

狭义并发强调对系统的请求操作是完全相同的，多适用于性能测试、负载测试、压力测试、稳定性测试场景；广义并发不限制对系统的请求操作，多适用于混合场景、稳定性测试场景。

Scenario场景

性能测试过程中为了模拟真实用户的业务处理过程,在 Loadrunner 中构建的基于事务、脚本、虚拟用户、运行设置、运行计划、监控、分析等的一系列动作的集合，称之为性能测试场景。场景中包含了待执行脚本、脚本组、并发用户数、负载生成器、测试目标、测试执行时的配置条件等。

Response Time响应时间

响应时间是指从客户端发一个请求开始计时,到客户端接收到从服务器端返回的响应结果结束所经历的时间，响应时间由请求发送时间、网络传输时间和服务器处理时间三部分组成。

Think Time思考时间

模拟正式用户在实际操作时的停顿间隔时间。

从业务的角度来讲，思考时间指的是用户在进行操作时，每个请求之间的间隔时间。

在测试脚本中，思考时间体现为脚本中两个请求语句之间的间隔时间。

CPU资源

CPU 资源是指性能测试场景运行的这个时间段内，应用服务系统的 CPU 资源占用率。CPU 资源是判断系统处理能力以及应用运行是否稳定的重要参数。应用系统可以包括应用服务器、web 服务器、数据库服务器等。

Load负载

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数。如果一个进程满足以下条件则其就会位于运行队列中：

- 它没有在等待 I/O 操作的结果
- 它没有主动进入等待状态（也就是没有调用'wait'）
- 没有被停止（例如：等待终止）

Std. Deviation标准差

该标准差根据数理统计的概念得来，标准差越小，说明波动越小，系统越稳定，反之，标准差越大，说明波动越大，系统越不稳定。包括响应时间标准差和 TPS 标准差等。

性能测试模型

PV计算模型

为了让性能测试的 PV 计算更接近生产线真实情况，通过数学建模的方式，对现有性能测试 PV 的计算公式进行验证、修订。

现有的 PV 计算公式是：

每台服务器每秒平均 PV 量=((总 PV*80%)/(24*60*60*40%))/服务器数量

刨除常数，其中关键的参数就是 80%和 40%，公式可以整理成：

每台服务器每秒平均 PV 量=2*(总 PV)/* (24*60*60) /服务器数量

以上的算法是根据以往的数据抽象得出，现在的目的就是经过再次研究最近的真实数据，得出目前更符合实际情况的算法，或者验证之前的算法仍然适用。

首先搜集已有数据，通过http://monitor.taobao.com搜集已有数据，如图 1-1 所示：

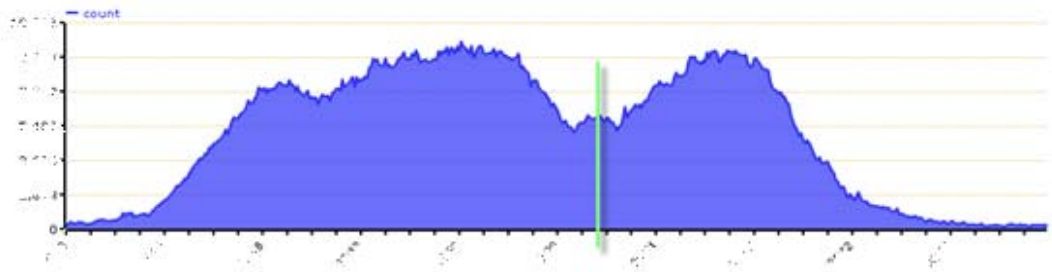

图 1-1

任何一天的分布图都与上图类似，故将这种分布视为整个淘宝网的浏览量分布。

其次进行统计数据：采取随机抽样的方式，选择了某些天的分布来做样本，再每半小时采点，考察其值的走势，找出其每一天的最大值，抽样的每个时刻的值可以转化为所占最大值的比例。得出图 1-2 的分布：

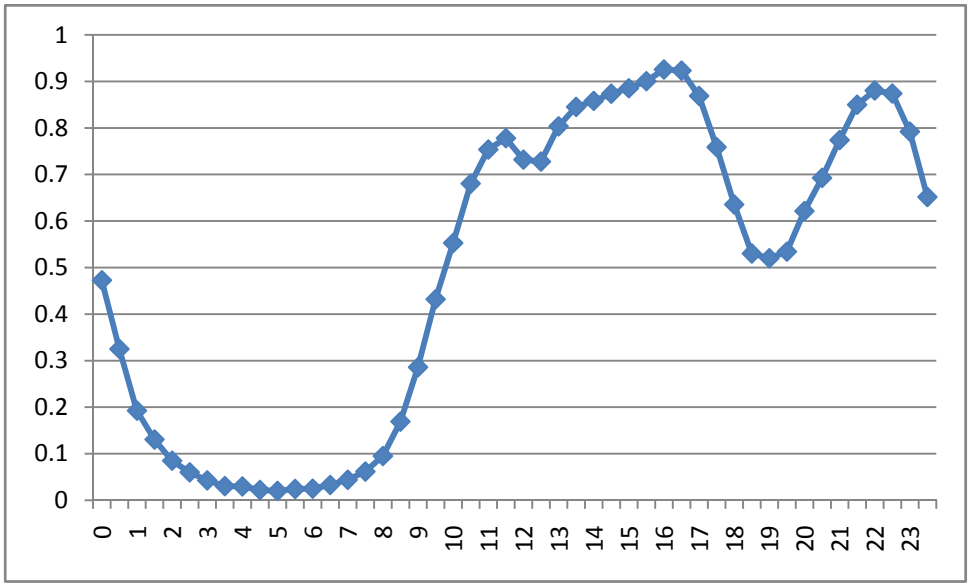

图 1-2

接下来采用拟合分布函数：方法是从现有的统计分布模型进行改进这时需要对分布图做调整，将一天的分布起止时间分别调整为有规律的两段。这样做的目的是可以对分布更易于把握。调整后的分布，如图 1-3 所示：

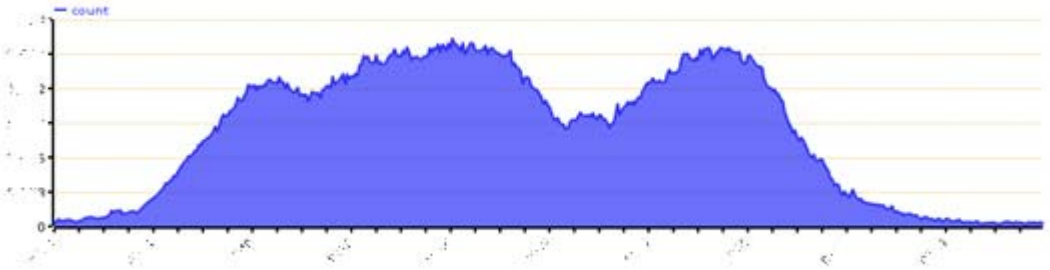

图 1-3

同理，可以使用上面的方法，进行半小时采样，得出的结果如图 1-4 所示：

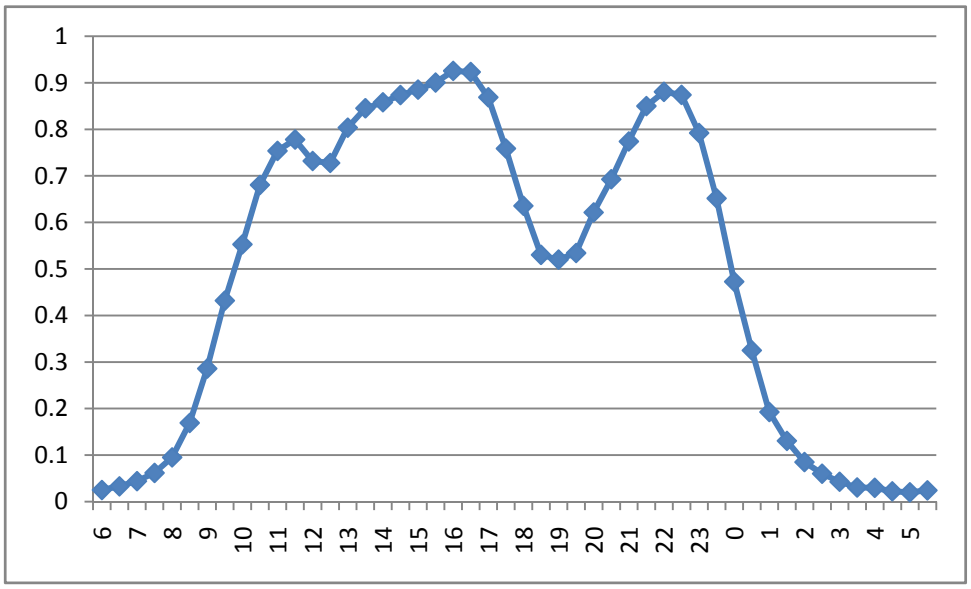

图 1-4

图 1-4 可以视为大体上符合某些统计分布，将上面的分布图如下划分成两部分，得到图 1-5：

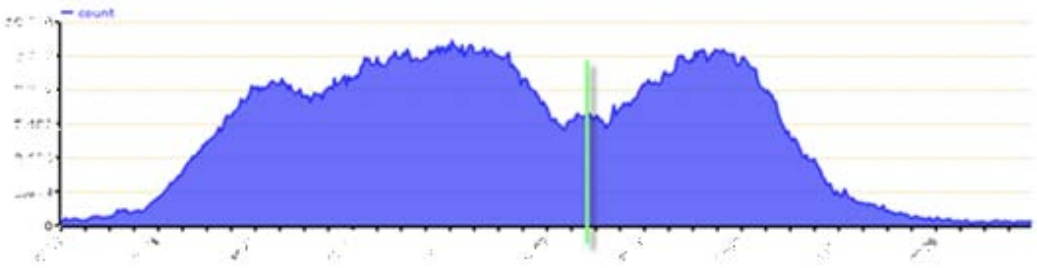

图 1-5

绿色的直线左边部分视为卡方分布，绿色直线的右边视为 t 分布。将已有数据利用 Mathematica6.0 软件拟合函数，可以得到下面的两张图 1-6 和 1-7：

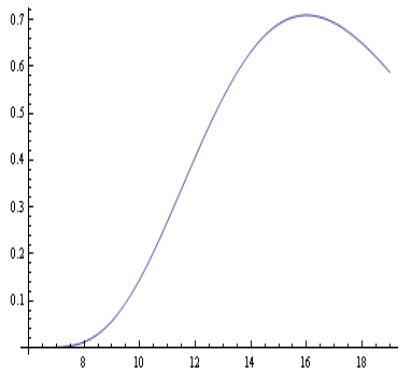

图 1-6

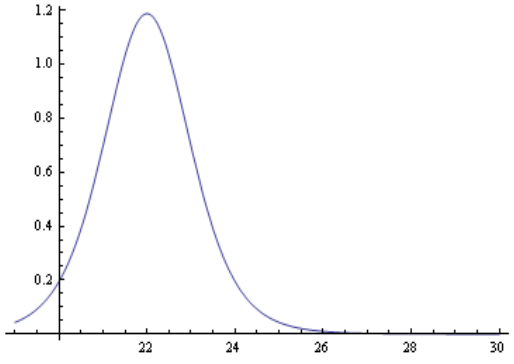

图 1-7

通过定积分求整个分布图的面积，然后求出最高值附近范围内的定积分，可以求得占据了 80%的 pv 量的总时间。根据这个数据，得出计算 pv 的公式变成：

每台服务器每秒平均 PV 量= ((80%*总 PV)/(24*60*60*(9/24)))/服务器数量

即 每台服务器每秒平均 PV 量=2.14*(总 PV)/* (24*60*60) /服务器数量

进而还可以得出最高峰的 pv 量是 1.29 倍的平均 pv 值。

PV->TPS转换模型

为了让 PV 在性能测试环境下可量化，根据 PV 的概念，通过以下方式将其转换成 TPS。

1. 性能测试脚本中，只保留与性能点相关的内容，异步处理的，保留多个请求；
2. 在执行场景中，不模拟浏览器缓存，确保每次请求都到达应用服务器；
3. 在执行场景中，每次迭代，都模拟一个新用户，而且清除用户缓存信息，确保每个用户每次发送请求都是全新的。

TPS波动模型

TPS 是性能测试过程中的重要参考指标，在淘宝的性能测试设计、执行、分析、评估等

各个阶段，TPS 都被看作是非常重要的一个环节；同时在性能测试是否通过的衡量参数指标里，它也有着举足轻重的影响。

众所周知，性能测试会依赖于特殊的硬件、软件、应用服务、网络资源，所以在性能场景执行期间，TPS 可能会表现为非常稳定，或者偶有振动，抑或遵循一定的上升或下降趋势。

TPS 表现轨迹可以总结为两大类：

一类是 TPS 有明显的大幅波动，不稳定。例如 TPS 轨迹缓慢下降，缓慢上升后骤降，呈瀑布型，呈矩形，分时间段有规律的波动，无规律的波动等。这些 TPS 的波动轨迹反映出被测试的性能点存在性能瓶颈，需要性能测试工程师与开发工程师查找性能瓶颈的原因。

另一类是 TPS 轨迹比较平稳，但是也存在波动现象。该类波动不明显，很难直接确定是否存在性能瓶颈。如果单场景性能测试、稳定性测试结果显示 TPS 达到了期望值，是否就确定 TPS 满足了期望呢？由于程序执行过程中服务器会有活动，造成 TPS 不可能是一条完整的直线，那么多大的 TPS 波动范围认为是可以被接受的呢？

Loadrunner TPS 分析图中涉及到了 4 个重要的参数，最大值、平均值、最小值和标准差值；平均值和标准差是衡量 TPS 是否稳定的重要因子。在此，我们强调一下 TPS 平均值和 TPS 标准差的概念。

TPS 平均值是在场景执行过程中，被测系统在指定时间段内的平均每秒处理的事务数量。TPS 标准差是根据数理统计的概念得来，反映被测系统的波动情况，标准差越小，说明波动越小，系统越稳定，反之，标准差越大，说明波动越大，系统越不稳定。既然 HP Mercury 提供了这么好的数据给用户，那我们就要充分运用它。

目前淘宝性能测试，Java Web 和 Java 接口占了绝大部分，Java Web 和 Java 接口在系统处理能力上有非常大的区别，如果将 Java Web 和 Java 接口分别做内部类比，由于是不同的应用，每个性能点的 TPS 平均值、TPS 标准差也是差距非常大；因此，如果定义一个 TPS 或者 TPS 标准差作为一个基线的话，不能应用到当前淘宝各类应用服务系统的性能测试中；这也就说明了如果纯粹拿一个 TPS 平均值或者 TPS 标准差，不能完全说明 TPS 是否稳定。

因此，淘宝性能测试团队提出一个 TPS 波动范围的概念，定义为 TPS 标准差除以 TPS 平均值，如公式 1-1 所示：

$$t\ (\text{TPS波动范围}) = \frac{\text{TPS标准差}}{\text{TPS平均值}} \times 100\%$$

公式 1-1

由此来观察 TPS 标准差与 TPS 的比值 t 有多大，如果这个比值 t 超过了一定的范围，就确认这个性能点的 TPS 不够稳定，也间接证明被测系统响应波动大而不满足性能期望。

性能测试团队搜集了以往 20 个典型项目中的 75 个性能场景，针对 JAVA Web 与 JAVA 接口的性能测试和稳定性测试，做了统计分析，得出可接受的 TPS 波动范围。如表 1-1 所示：

被测系统种类	测试是否通过	测试类型	超时概率	采样时间(s)	可接受波动范围
Java 接口	是	性能测试	低于万分之一	1	5%±3%
Java 接口	是	稳定性测试	低于万分之一	60	5%±3%
Java Web	是	性能测试	低于万分之一	1	4%±2%
Java Web	是	稳定性测试	低于万分之一	60	4%±2%

表 1-1

建立 TPS 波动范围模型的出发点是为淘宝性能测试以及项目组建立一个可参考的标准，

更为有效的控制、降低项目上线后的风险。

有了这个统计模型，在性能测试衡量 TPS 是否稳定的时候，就有了一个参考依据，如果 TPS 的波动范围高于表 1-1 所示，性能测试工程师就要配合项目组开发人员查找定位性能瓶颈，并做进一步的性能优化。

注意：这个 TPS 波动模型是会随着淘宝各应用性能表现而相应变更的。当然模型是基于搜集来的数据、分析、论证得来的，也不排除一些特殊的例子，在具体的性能测试应用中，会做出更为专业的评估。

性能测试策略

性能测试确立了成型的整套策略，从性能测试环境，到性能测试数据、执行、调优、性能评估，都已经摸索出操作性很强的策略。具体策略如下：

1. 模拟生产线真实的硬件环境。
2. 服务器置于同一机房，最大限度避免网络问题。
3. 以 PV 为切入点，通过模型将其转换成性能测试可量化的 TPS。
4. 性能测试数据分为基础数据和业务数据两部分，索引和 SQL 都会被测试到。
5. 日志等级设置成 warn，避免大量打印 log 对性能测试结果的影响。
6. 屏蔽 ESI 缓存，模拟最坏的情况。
7. 先单场景，后混合场景，确保每个性能瓶颈都得到调优。
8. 拆分问题，隔离分析，定位性能瓶颈。
9. 根据性能测试通过标准，来判断被测性能点通过与否。
10. 针对当前无法解决的性能瓶颈，录入 QC 域进行跟踪，并请专家进行风险评估。

性能测试类型

性能测试压力变化模型

随着单位时间流量的不断增长，被测系统的压力不断增大，服务器资源会不断被消耗，TPS 值会因为这些因素而发生变化，而且符合一定的规律。淘宝网性能测试压力变化模型如图 1-8 所示：

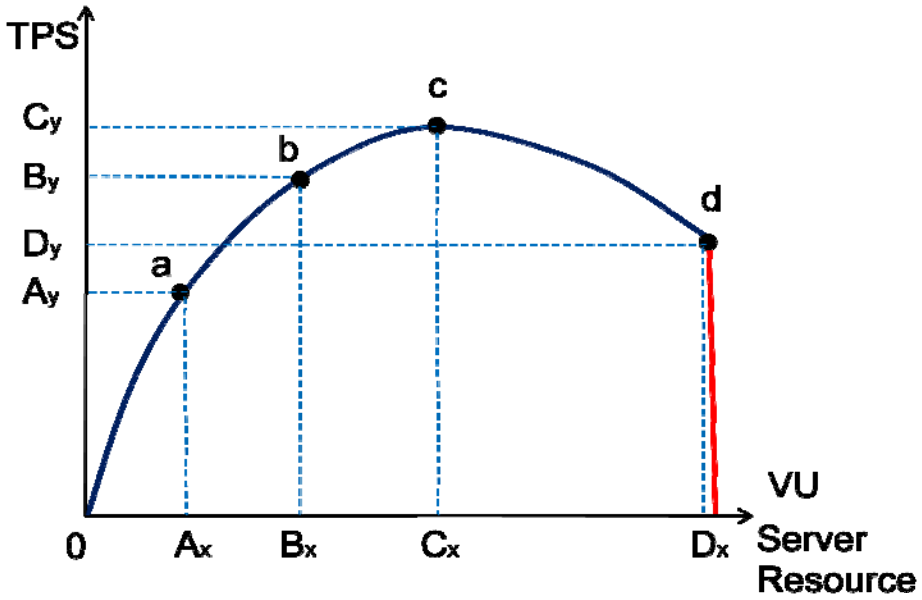

图 1-8

图中

a 点：性能期望值

b 点：高于期望，系统资源处于临界点

c 点：高于期望，拐点

d 点：超过负载，系统崩溃

性能测试类型

由上述压力变化模型，将淘宝网性能测试分成狭义的 4 种类型：

1. 性能测试
2. 负载测试
3. 压力测试
4. 稳定性测试

性能测试

a 点到 b 点之间的系统性能

定义：狭义的性能测试，是指以性能预期目标为前提，对系统不断施加压力，验证系统在资源可接受范围内，是否能达到性能预期。

负载测试

b 点的系统性能

定义：狭义的负载测试，是指对系统不断地增加压力或增加一定压力下的持续时间，直到系统的某项或多项性能指标达到极限，例如某种资源已经达到饱和状态等。

压力测试

b 点到 d 点之间

定义：狭义的压力测试，是指超过安全负载的情况下，对系统不断施加压力，是通过确定一个系统的瓶颈或不能接收用户请求的性能点，来获得系统能提供的最大服务级别的测试。

稳定性测试

a 点到 b 点之间

定义：狭义的稳定性测试，是指被测试系统在特定硬件、软件、网络环境条件下，给系统加载一定业务压力，使系统运行一段较长时间，以此检测系统是否稳定，一般稳定性测试时间为 n*12 小时。

性能测试执行方法

性能测试通常采用先单场景，后混合场景的执行方法。

单场景

针对单个性能测试点，构建一个性能测试场景，而进行的性能测试。单场景适用于性能测试、负载测试、压力测试、稳定性测试。

混合场景

为了尽量模拟生产线上运行的业务压力或用户使用场景，测试系统的整体性能是否满足性能需求，把经过一定规则筛选的性能测试点，按照合乎实际逻辑的虚拟用户请求、并发，组合成一个混合场景。

混合场景的特征，通常包含两个或者两个以上的脚本组，执行时间较长。混合场景通常在稳定性测试、负载测试中使用。

性能监控

性能测试需要使用不同的工具，结合系统日志，监控服务器、应用等方面的多项指标。以下阐述监控指标、监控工具和监控步骤。

监控指标

性能测试通常需要监控的指标包括：

1. 服务器 Linux（包括 CPU、Memory、Load、I/O）。
2. 数据库：1.Mysql 2.Oracle（缓存命中、索引、单条 SQL 性能、数据库线程数、数据池连接数）。
3. 中间件：1.Jboss 2. Apache（包括线程数、连接数、日志）。

4. 网络： 吞吐量、吞吐率。
5. 应用： jvm 内存、日志、Full GC 频率。
6. 监控工具（LoadRunner）：用户执行情况、场景状态、事务响应时间、TPS 等。
7. 测试机资源：CPU、Memory、网络、磁盘空间。

监控工具

性能测试通常采用下列工具进行监控：

1. Profiler

一个记录 log 的类，阿里巴巴集团自主开发，嵌入到应用代码中使用。

2. Jstat

监控 java 进程 GC 情况，判断 GC 是否正常。

3. JConsole

监控 java 内存、java CPU 使用率、线程执行情况等，需要在 JVM 参数中进行配置。

4. JMap

监控 java 程序是否有内存泄漏，需要配合 eclipse 插件或者 MemoryAnalyzer 来使用。

5. JProfiler

全面监控每个节点的 CPU 使用率、内存使用率、响应时间累计值、线程执行情况等，需要在 JVM 参数中进行配置。

6. Nmon

全面监控 linux 系统资源使用情况，包括 CPU、内存、I/O 等，可独立于应用监控。

7. Valgrind

监控 C/C++程序是否存在内存泄漏，基于 linux 环境。

8. Vmmap 和 ApplicationVerifier

监控 C/C++程序是否存在内存泄漏，基于 windows 环境。

监控步骤

1. 确定需要监控的目标。
2. 确定监控和分析所需信息（可以采用 CheckList 模版法，列出所需要监控的指标，信息）。
3. 确定监控所使用的工具（根据性能点的类型，以及需要关注的性能指标来确定）。
4. 收集监控所得数据（可以采用日志监控＋辅助工具法，收集所需监控数据）。
5. 分析所采集的数据，定位性能瓶颈。
6. 性能调优。
7. 循环以上步骤，直到调优完毕。

性能分析

淘宝网已经跨入分布式时代，分布式的到来，也为性能测试增加了难度。性能测试分析原则和分析方法，显得尤为重要。以下阐述性能测试分析原则、分析信息来源、分析标准和分析工具。

分析原则

在分布式架构下，性能瓶颈分析也变得相对困难。根据不同的应用系统，不同的测试目标，不同的性能关注点，根据性能指标的表现，采用“拆分问题，隔离分析”的方法进行分析，即逐步定位、从外到内、从表及里、逐层分解、隔离排除。

淘宝性能分析，可按以下顺序：

中间件瓶颈（apache/jboss 参数配置、数据库参数配置）-> 应用服务的 debug log -> 应用服务的 filter log -> 本应用的性能瓶颈（SQL 语句、索引、业务逻辑、线程池设置、算法）-> 服务提供者的性能瓶颈 -> 相关联的底层存储应用的性能瓶颈

注：以上是比较通用的分析过程，具体性能测试查找瓶颈过程中，需要具体问题具体分析。

分析信息来源

1. 监控工具所采集的信息。包括 TPS、响应时间、用户并发数、JVM 内存、Full GC 频率等。
2. 应用服务器的日志。包括错误日志、超时日志等。
3. 项目配合人员所提供的信息。包括 DBA 提供的数据库监控信息、开发人员提供的代码逻辑信息。

分析标准

通过性能指标的表现形式，分析性能是否稳定。比如：

1. 响应时间是否符合性能预期，表现是否稳定。
2. 应用日志中，超时的概率，是否在可接受的范围之内。
3. TPS 维持在多大的范围内，是否有波形出现，标准差有多少，是否符合预期。
4. 服务器 CPU、内存、load 是否在合理的范围内，等等。

分析工具

对于部分性能指标，可借助自动分析工具，统计出数据的总体趋势：

1. LoadRunner analysis 分析
   LoadRunner analysis 是 loadrunner 的一个部件，用于将运行过程中所采集到的数据生成报表，主要用于采集 TPS、响应时间、服务器资源使用情况等变化趋势。
2. Memory Analyzer 分析
   Memory Analyzer 工具可以解析 Jmap dump 出来的内存信息，查找是否有内存泄漏。
3. nmon_analyser 分析
   nmon 工具可以采集服务器的资源信息。列出 CPU、MEM、网络、I/O 等资源指标的使用情况。

性能测试通过标准

性能测试从需求、设计、准备、执行到分析，最后需要判断性能测试是否通过。性能测试工程师最终需要考虑很多因素，判断的标准相应的也会有多个维度。

此处，引入超时阀值和超时概率两个概念。超时阀值，是指在框架模板中定义的超时时间，当服务器响应超过该定义值时，被测程序仍然响应请求，同时 Profiler 会打印超时日志。超时概率，是指超时次数除以总请求次数所得的值。

由上述概念可以得出，此处的超时不等于请求失败。

1. 性能测试通过标准

性能测试通过标准包括服务端性能、前端性能和用户体验性能，通过标准如表 1-2 所示：

类别	判断维度	不通过	通过	备注
服务端性能	超时概率	大于万分之一	小于万分之一	1、性能测试团队根据通过标准，判定被测性能点不通过，而 PM、PDM 认为可以通过时，需要进行沟通。沟通以会议形式，由[illegible]来评判。 2、页面大小包括 js/css/图片等资源。 3、最佳情况下，页面下载时间小于 [illegible]s。
	错误概率	大于万分之一	小于万分之一	
	TPS	小于期望高峰值	大于期望高峰值	
	TPS 波动范围	见 TPS 波动模型	见 TPS 波动模型	
	CPU 利用率	大于 75%	小于 75%	
	Load	平均每核 CPU 的 load 大于 1	平均每核 CPU 的 load 小于 1	
	JVM 内存使用率	大于 80%	小于 80%	
	Full GC 频率	平均小于半小时 1 次	平均大于半小时 1 次	
前端性能	YSlow 评定	YSlow 评定为 C 级以下	YSlow 评定为 C 级，或 C 级以上	
用户体验性能	页面大小	页面大于 [illegible]	页面小于 [illegible]	
	页面响应	[illegible]M 带宽，大于 [illegible]s	1M 带宽，小于 [illegible]	

表 1-2

2. Profiler 超时阀值设定

淘宝性能测试团队联合架构师和各共享中心负责人，定义了不同应用的 Profiler 超时阀值。目前已经界定出的超时阀值如表 1-3 所示：

超时阀值定义	页面 http 响应时间	页面 http 响应时间	[illegible]ms
	UIC 响应时间	单条接口（缓存）	[illegible]ms
		批量接口（缓存）	[illegible]ms
		UserPromotedService 接口	[illegible]ms
	IC 响应时间	提交操作的接口	[illegible]ms
		单条查询类的接口	[illegible]ms
		批量查询类的接口	[illegible]ms
		有缓存的接口	[illegible]ms
	TC 响应时间	不与支付宝交互的接口	[illegible]ms
		与支付宝交互的接口（mock 支付宝）	[illegible]ms
	SC 响应时间	一般的接口	[illegible]ms
		相册相关接口	[illegible]ms

表 1-3

性能测试通过标准，是判断性能测试通过与否的最全面的参考依据。随着淘宝性能测试的发展，这些标准也会随之相应改动，要求会越来越严格。

性能测试流程

性能测试是一个系统工程，涉及到 PDM、PM、性能测试工程师、DBA、SCM、OPS 等多个角色。从性能测试申请，到性能测试设计、性能测试环境、性能测试数据、性能测试执行、性能测试调优，都离不开这些角色的相互配合，共同协助。因此，淘宝网确立了一整套性能测试流程，采取团队合作的方式，降低沟通成本，提高工作效率。

简化的性能测试流程图如图 1-9 所示：

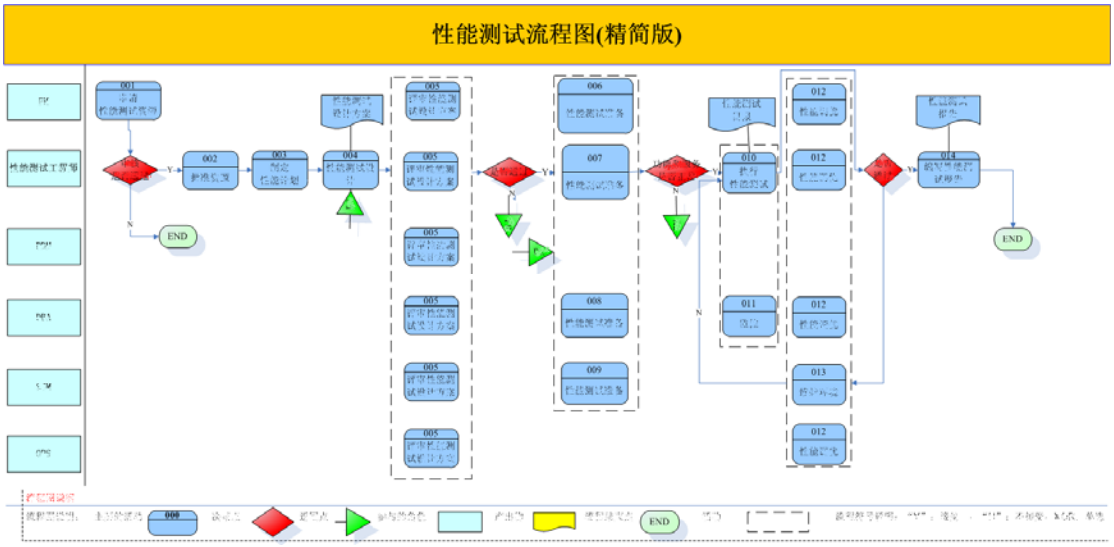

图 1-9

性能测试文件模版

为了配合性能测试流程，更好的开展测试工作。性能测试开发多个模版，用于指导和规范文档编写，提高效率。当前使用的模版和规范包括：

1. 性能测试资源申请模版
2. 性能测试设计方案模版
3. 性能测试报告模版
4. 性能测试日报模版
5. 淘宝性能测试脚本和场景制作规范

结论

通过以上的介绍，详细描述了性能测试指标的概念，性能测试 PV 模型，PV->TPS 转换模型、TPS 波动模型，性能测试策略，性能测试类型，性能测试执行方法，性能监控，性能分析，性能测试通过标准，性能测试流程和文件模板。

希望本白皮书能够更好的指导性能测试、以及性能测试协作团队更好的完成性能测试工作，达到最佳的性能测试目的，降低沟通成本，提高效率。让更多的淘宝同学们理解性能测试、协助性能测试，更好的参与到性能测试中来。

参考文献

【1】 Load Runner 8.1 User Manual. Mercury Interactive , 2006
【2】 Fundamentals of Load Runner 8.1. Mercury Interactive , 2005
【3】 Mercury Load Runner 8.1 教程. Mercury Interactive , 2006
【4】 于涌 软件性能测试与 Load Runner 实战. 北京：人民邮电出版社出版
【5】 姜艳，于波 Load Runner 性能测试应用. 北京：电子工业出版社
【6】 陈绍英，夏海涛，金成姬. Web 性能测试实践【M】. 北京：电子工业出版社
【7】 Mathematica 5 在大学数学中的应用，丁大正，电子工业出版社

版本更新说明

本次 V0.3 版，在原 V0.1 版本基础上做了新增、修订和丰富。具体如下：

新增：TPS 波动模型、性能测试通过标准。

修订：性能测试指标中的 Transaction、Concurrency、Scenario 的定义；

丰富：性能测试执行方法中的单场景、混合场景的描述；

V0.3 旨在完善性能测试白皮书的整个框架。主要将不可或缺的 TPS 波动模型和性能测试通过标准确立并增加进来。性能测试白皮书的体系已经形成，后续，我们将根据收集到的建议进行完善、丰富。

作者介绍

淘宝性能测试团队成员，将性能测试工作实践升华、提炼之后，编写成白皮书。自从白皮书的概念提出以来，各成员都参与了研究、编写、修订工作。在此，感谢每位成员的辛苦付出。作者介绍如下：

- 丘虚

  总体界定了白皮书的范围，规划编写资源、提供宏观的修订意见。

- 悟石

  协调整体编写和修订工作。主要负责编写 PV->TPS 转换模型、性能测试策略、性能测试类型、性能测试通过标准、性能测试流程和性能测试文件模版，参与编写性能监控、性能分析、TPS 波动模型。

- 耿电

  提出白皮书概念。主要负责编写性能测试指标、TPS 波动模型、性能测试执行方法，参与编写性能分析。

- 元壮

  数学建模。主要负责编写 PV 计算模型、建模，参与 TPS 波动模型建模。

- 云帅

  主要负责编写性能监控和性能分析，参与编写 TPS 波动模型。

- 词馨

  负责 review 白皮书，提出章节调整和修正建议。

在介绍作者的同时，更要特别感谢为白皮书 review 的测试部门的其它同学们。鉴于本次为 0.3 版本，不免有错误或不全之处，欢迎读阅的同学们给出宝贵的建议，以便持续改进和完善。