文章编号：1672-2892(2012)02-0231-06

面向 JavaEE Web 应用系统的性能诊断策略

薛振伟

(中国工程物理研究院 计算机应用研究所，四川 绵阳 621900)

摘 要： JavaEE Web 应用系统的复杂性极容易导致性能问题，其中服务器端是性能问题产生的主要根源。针对服务器环节，基于“由外而内，由表及里，层层深入”的基本原则，提出一种系统化的性能诊断思路和流程，从用户视角、资源视角和组件视角逐层深入，分别从用户事务、系统资源、Web 资源和页面组件 4 个层次阐述性能诊断的策略，为同类系统的性能诊断工作提供参考。

关键词： 性能诊断；Web 应用；JavaEE 平台

中图分类号： TN915；TP306.2 **文献标识码：** A

Strategies of performance diagnosis for JavaEE Web application system

XUE Zhen-wei

(Southwest Computing Center, Mianyang Sichuan 621900, China)

Abstract: The complexities of JavaEE Web application system easily lead to many performance problems which are mainly caused by servers. The systematic diagnosis process and solutions for servers are presented based on the basic principal described by “From outside to inside, from surface to core, breakdown step by step”. Three angles of view such as user experiences, resources and components are presented to drilldown the problems step by step. Then the strategies of performance diagnosis are discussed from four aspects including user transactions, system resources, Web resources and page components. Some good experiences are put forward for similar projects of performance diagnosis.

Key words: performance diagnosis; Web application; JavaEE

基于 JavaEE 多层体系架构的 B/S 模式 Web 应用软件因其硬件要求较低、良好的跨平台性和可扩展性等优点，逐步成为管理信息系统技术方案的主流代表。但大多数关键 JavaEE Web 应用系统均具有如下特点：运行环境、体系结构、实现技术、安全措施、业务关系、用户角色均复杂，用户量大、数据容量大和性能要求高等。这种高复杂性和高要求给项目实施带来了诸多风险，其中一个重要的方面就是“性能”问题，主要表现为响应时间、吞吐量、执行效率、资源占用、稳定性及可靠性等方面的问题。基于这种复杂性，系统的任何一个环节出现瓶颈，都会导致系统出现性能问题，这也给性能诊断带来了挑战[1-3]。

总之，任何复杂的系统都可大体分为 3 部分：客户端、网络和服务器[4]，但鉴于 JavaEE Web 应用的特点，客户端出现问题的几率较低。因此主要问题集中在系统的瓶颈是出现在网络环节，还是服务器环节？在正常的工作过程中，首先需要确定网络是否存在瓶颈，此时可以选用专业的网络测试软硬件工具进行测试，此处不再赘述。多数情况下，系统的瓶颈出在服务器环节，此时需要进一步层层推进，判查相应环节，直到最后找出造成性能问题的根本原因。本文主要针对服务器环节，讨论性能诊断的原则和流程，并从用户事务、系统资源、Web 资源和页面组件 4 个层次分别提出性能诊断的策略，为同类系统的性能诊断工作提供参考。

1 性能诊断原则及流程

在进行性能诊断时一个普遍遵循的原则是“由外而内，由表及里，层层深入”[4]，这也符合对问题的认识和挖掘的普遍规律，即从表象开始，逐层深入，不断排除各类因素，最后找到问题的真正原因。

收稿日期： 2010-09-02；**修回日期：** 2011-09-22

基金项目： 中国工程物理研究院科学技术发展基金资助项目(2009B0403048)

在开展 JavaEE Web 应用的诊断之前,须将性能测试各阶段的测试结果和测试数据(包括监控数据)全部收集起来，为进一步性能诊断做好准备。接着再对系统是否出现性能问题进行初级判断,即从原始数据中查看系统的响应时间是多少,判断它是否满足用户对性能的期望，如果不满足，就要继续判断系统在哪个环节出现了瓶颈。本文给出了 1 个通用的性能诊断流程，如图 1 所示。

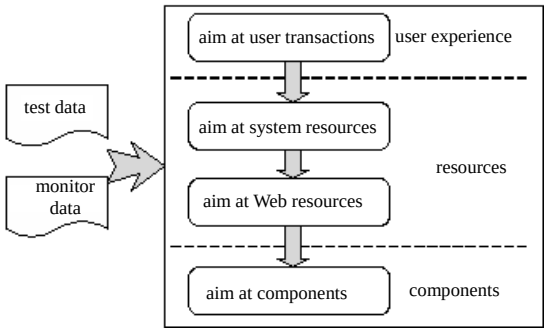


Fig.1 Performance diagnostic process
图 1 性能诊断流程

首先，以与最终用户体验直接相关的交易响应时间为起点，面向用户事务的表现进行判断，尽可能锁定几个突出的问题，基本确立下一步的诊断方向。然后，通过面向资源瓶颈的诊断，确定性能下降是否属系统资源瓶颈造成的。继而通过面向 Web 资源的诊断，进一步验证上 2 步的中间判断结果。最后，通过面向页面响应时间的诊断，找到响应时间比较长的页面，并进一步判断到底是页面的哪些组件(Servlet,JSP(Javaserver Page),EJB(Enterprise Java Bean))或其他)引起的延迟，发生延迟时在服务器上和网络传输上耗费的时间分布如何等等。

同时，在流程之外，测试与诊断工具的选择也非常重要，因为一般性能诊断工作在很大程度上依靠工具来进行。在选择工具时，要考虑工具本身的可靠性、测试能力和资源管理能力[5]，商业工具(如 HP LoadRunner、Compuware Vantage 系列等)的综合指数和工作效率比较高，但是并不能解决所有的诊断问题，这就需要在开展性能诊断工作时有针对性的选择。

2 性能诊断策略

2.1 面向用户事务的诊断

与最终用户体验直接相关的是用户事务的响应时间表现。用户事务主要针对业务而言，1 个用户事务通常由 1 个或一系列的用户操作组成。分析用户事务通常从不同的角度来对其执行结果进行分析。

在 LoadRunner 中有 2 种和事务相关的概念：Action 和 Transaction[6]。Action 是用户的一系列操作的组合，Transaction 是用户某一具体的动作。Action 通常会包含一系列功能相关的 Transaction。在测试过程中，事务往往有 3 种执行结果：1) 通过(Pass)：按照预定计划执行了全部虚拟用户脚本；2) 失败(Fail)：虚拟用户脚本执行过程中发生了错误；3) 停止(Stop)：由于测试时间结束等原因而停止执行用户脚本。

在 LoadRunner 中提供了 8 张与事务分析相关的事务分析图[6]，从这 8 张图来进行用户事务级的初级诊断。

1) 事务综述图(Transaction Summary)

对事务进行综合分析是性能分析的第 1 步，通过分析测试时间内用户事务的成功与失败情况，可以直接判断出系统是否运行正常。当发现事务运行不正常时，才需要进一步的分析和诊断。在查看当前测试场景执行后的事务综述图时，如果发现失败的事务非常多，则说明系统出现了瓶颈或程序在执行过程中发生了问题。然后分析是哪些业务失败的较多，从而更深入地分析该事务的细节。

2) 事务性能摘要图(Transaction Performance Summary)

通过该图直接判断响应时间是否符合用户的需求。要重点关注事务的平均和最大执行时间，如果其范围不在用户可以接受的范围内，则需要进行原因分析。

3) 事务平均响应时间分析图(Average Transaction Response Time)

通过分析该图，可以分析测试场景运行期间应用系统随时间发展的性能走向，从而确定是否存在问题，为进一步的分析提供依据。如果随着测试时间的增加，系统处理事务的速度开始逐渐变慢，则说明系统随着运行时间的增加整体性能将会有下降的趋势。

4) 每秒通过事务总数分析图(Total Transaction per Second)

该图关注服务器整体处理事务的情况，如果系统性能稳定，在同等压力下，该图应该接近直线，而不是逐渐倾斜。如果发现系统每秒通过的事务总量越来越少，则说明整体性能在下降。下降的原因有很多，例如内存泄露、程序中的缺陷、资源瓶颈等都可能导致该指标下降。因此发现性能问题时，要综合其他测试数据(包括资源监控数据)共同进行分析。

5) 每秒通过事务数分析图(Transaction per Second)

通过该图可以确定系统在任何给定时刻的每一个事务的实际负载。与 Total Transaction per Second 相比，该

图具体到每一个事务，可具体分析每一个事务的详细数据。一般情况下，是先在查看 Total Transaction per Second 之后，再有针对性地查看每个具体的事务的 TPS 图来进行验证和深入诊断。如果随着测试的进展，发现应用系统每秒通过的事务数在减少时，说明系统的处理能力在降低。

6) 事务响应时间与负载分析图(Transaction Response Time Under Load)

通过该图可以看出在任一时间点事务响应时间与用户数之间的关系,从而掌握系统在用户并发方面的性能数据，为扩展应用系统提供参考。由于该图用于查看虚拟用户负载对执行时间的总体影响，因此在分析具有渐变负载的测试场景时更加有用。

7) 事务响应时间百分比图(Transaction Response Time(Percentile))

该图是工具通过一些统计分析方法间接得到的图表,可以分析在给定事务响应时间范围内能够执行的事务百分比。分析该图要从整体出发,如果大多数事务的响应时间均在可接受的范围内,只有极少事务的响应时间较长,则整个系统仍然符合要求。

8) 事务响应时间分布情况分布图(Transaction Response Time Distribution)

通过该图可以直接看出虚拟用户的事务响应时间分布情况，了解测试过程中不同响应时间的事务数量。如果系统预先定义了相关事务可以接受的最小和最大事务响应时间,则可以使用此图确定服务器性能是否在可以接受的范围内。

2.2 面向系统资源的诊断

在面向用户事务的诊断中，初步判断出应用系统是否出现了性能问题，以及哪些用户事务的时间表现不能满足需求。当发现问题后，建议立即对操作系统资源状况进行诊断，以确定在测试中是否存在系统资源瓶颈。

对系统资源瓶颈的诊断基于在测试过程中对所有服务器操作系统资源的监控数据。无论是 Windows 平台还是 Unix(Linux)平台，操作系统都提供了自带的系统监视手段。当然，也可以使用第三方的监控工具，如 Quest Spotlight 系列、LoadRunner Monitor 等。

对于服务器而言，一般情况下主要关注 CPU、内存、磁盘 I/O 和网络 4 个方面[6]。

1) CPU

如果 Processor Queue Length 大于 2，而 CPU 利用率(%Processor Time)一直很低，则存在处理器阻塞。使用%Processor Time 指标查看 CPU 饱和状况，该值显示所有 CPU 的线程处理时间。如果 1 个或多个处理器的相应数值持续超过 90%,则表示此测试的负载对于目前的硬件过于沉重,CPU 有可能是瓶颈点。还需要监控%Interrupt Time 指标，如果该计数器持续大于 15%，而%Processor Time 也持续超过 90%，可以断定 CPU 负荷过重，CPU 是系统瓶颈点。

2) 内存

物理内存的可用数(Available Mbytes)至少要有 10% 的物理内存值。如果持续过低则说明总的内存可能不足，或者某程序始终占用而没有释放内存，系统可能存在严重内存泄露问题。如果 Pages/sec 持续高于几百，则有可能需要增加内存以减少换页的需求,但此时还需进一步研究页交换活动。因为过多的页交换使用大量的硬盘空间，在运行使用内存映射文件的程序时，同样会引起页交换过多的情况，此时可能导致页交换内存瓶颈，也可能导致页交换的磁盘瓶颈问题。因此必须把 Pages/sec 指标与 Disk 指标结合起来判断，在内存判断环节引入 Disk 计数器指标，如 Physical Disk 的%Disk Time 指标和 Avg.Disk Queue Length 指标。如果 Page Reads/sec 指标的值很低，但同时%Disk Time 和 Avg.Disk Queue Length 的值却很高，则可确定为磁盘瓶颈。但如果 Avg.Disk Queue Length 增加的同时 Page Reads/sec 指标并未降低，则确定为内存不足。

内存泄露的原因较多，不能单靠一类指标来确定是某一个方面的原因。建议监视 Memory 的 Available Bytes 和 Committed Bytes 指标以查看内存行为，同时监视可能存在内存泄露的进程(Process)的 Private Bytes、Working Set 和 Handle Count 指标，如果 Process/Private bytes 和 Process/working set 这 2 个计数器值升高，但是 Available Mbytes 降低则可能存在内存泄漏。如果怀疑是内核模式进程导致了泄漏,则还应该监视 Memory 的 Pool Nonpaged Bytes 和 Pool Nonpaged Allocs 指标，以及 Process 的 Pool Nonpaged Bytes 等内容。如果池被填满，则可能发生内存泄露。综上所述，由于涉及的因素较多，内存问题的诊断比较复杂，这里建议在分析过程中要比对，多换算，不能通过一两个指标就简单地下结论，要有充分的依据或数据才能确定是否存在“内存泄露”问题。

3) 磁盘 I/O 瓶颈

如果 Disk Time 指标和 Avg.Disk Queue Length 指标都很高，而 Memory 的 Page Reads/sec 却很低，则确定为存在磁盘瓶颈。Disk Queue Length 指标显示磁盘中未完成的请求数量。如果队列长度始终大于 3，则表示磁盘或

者内存有问题。还需要引入内存指标检查具体问题在哪里。如果%Disk Time 比较大，而 Current Disk Queue Length 大于 2，则表示需要提高系统磁盘处理性能，以提升系统整体业务处理能力。

4) 网络瓶颈

Bytes Total/sec 为发送和接收字节的速率，包括帧字符在内。Current Bandwidth 为以“bit/s”估计的网络接口的当前带宽。判断网络连接速度是否有瓶颈，可以将这 2 个指标进行比较，用 Bytes Total/sec 的值和 Current Bandwidth 的值相除，结果应该小于 50%。

Current Late Send Rate 指标是评估网络瓶颈时最重要的性能指标。当发送包的时间比预定发送时间至少晚 0.5 s 时，它就会报告 1 次发送延迟。当然发送延迟可能由以下 3 个原因造成：缺少可用的带宽来满足所有的客户端请求；缺少处理器周期来按时完成所有预定操作；从原始数据源读取而产生的延迟。其中第 2 和第 3 个原因不是直接与网络瓶颈有关，此时为了确定是否是网络瓶颈造成的发送延迟，需一步进行比对，还要判读是网络瓶颈还是缓冲问题。

2.3 面向 Web 资源的诊断

通过对服务器端系统资源状况的诊断，可基本确定服务器端是否存在资源瓶颈。在此前提下，从 Web 资源的角度进一步分析，从服务器的 Web 指标表现来分析服务器的性能问题。

首先，HTTP 1.x 协议定义了 1 组标准的 HTTP 服务器状态代码，对系统执行状态进行描述，通过很多监测工具都可以捕捉到这些消息。通过它们可以了解系统的请求、在不同传输环节的状态等内容，从而有利于发现系统的性能瓶颈。最常用的 HTTP 状态代码有 4 类：HTTP 2××，表示客户端请求被成功接收、理解和接受；HTTP 3××，表示用户代理要想完成请求，还需要发出进一步的操作；HTTP 4××，表示客户端发生了错误；HTTP 5××，表示服务器端发现自己出现了错误，不能继续执行请求。

在 LoadRunner 中提供了 4 个 Web 资源分析相关的分析图[6]，可按照下述顺序进行分析和诊断：

1) 点击率图(Hits per Second)

通过分析该图随着时间的变化，可以判断系统是否稳定。如果在图中显示系统每秒接受的点击次数在下降，表明服务器的响应速度在变慢，需要更进一步分析。

2) 吞吐率图(Throughput)

可以依据服务器吞吐量来评估虚拟用户产生的负载量，也可以看出服务器在数据流量方面的处理能力以及是否存在瓶颈。“吞吐率”图与“点击率”图形状基本类似，区别是“点击率”图是每秒服务器处理的 HTTP 申请数，“吞吐率”图是客户端每秒从服务器获得的总数据量。

3) 每秒 HTTP 响应数图(HTTP Response per Second)

通过分析该图，可以判断服务器在压力状态下的整体运行情况。状态代码最直接地反映了 HTTP 请求和响应的具体状态，通过对结果进行分组分析，可以定位生成错误的组件。

4) 每秒连接数图(Connections per Second)

通过该图可以看出服务器的处理情况。新连接数应该是每秒点击次数的一小部分，就服务器、路由器和网络资源消耗来说，新的 TCP/IP 连接非常昂贵。在理想状态下，很多的 HTTP 请求都应该使用同一个连接，而不是每个请求都新打开一个连接。

2.4 面向页面组件的诊断

Web 资源诊断是更深入一步的分析，结合用户事务分析结果，基本可以定位是否是应用程序发生了问题。如果是 Web 应用程序发生了问题，则需要对页面组件进行更深入的分解，以定位究竟是应用程序的哪一部分发生了问题。

在多数情况下，页面组件从发出请求到最终用户可以分为 7 个阶段，分别是：客户端时间、DNS(Domain Name Service)处理、建立连接、发送请求、等待服务器处理、接收服务器反馈和客户端缓存读取，其工作原理和时间分解如图 2 所示。

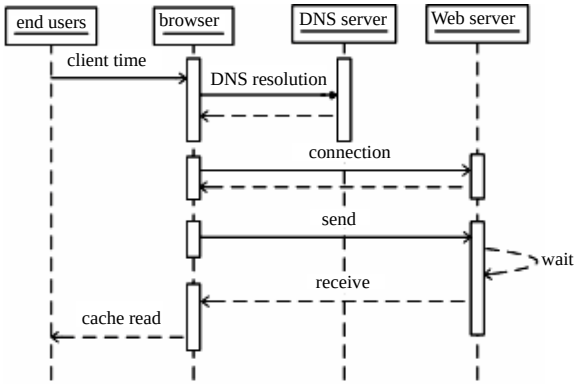


Fig.2 Component time breakdown
图 2 组件时间分析

在特殊情况下，如果使用 FTP(File Transfer Protocol)

传输或者 SSL(Secure Socket Layer)加密等，则还要加上 FTP 验证和 SSL 握手的时间。

在 LoadRunner 中提供了 7 个关键网页细分图[3]，显示每个页面及其组件的相关下载时间和大小，主要用于评估页面内容是否影响事务响应时间。通过与不同的事务图关联，可以深入分析 Web 系统中下载慢或中断连接等问题的原因，确定系统性能问题出处。建议按照下述顺序进行分析和诊断：

1) 页面分解总图(Web Page Breakdown)

通过总体趋势曲线图形获得各个组件的信息，了解哪些页面的下载时间比较长。如果存在占用下载时间明显高于其他占用时间，且随着测试时间的延长呈递增趋势的情况，可初步判断这些页面可能存在问题。

2) 页面下载时间细分(Download Time Breakdown)

选择可能存在问题的事务产生的页面，查看“Download Time Breakdown”图形。通过进一步细分，可以看出分解的各个组件大小，以及各个组件的下载时间。此时关键看页面的下载时间，暂时排除第 1 次缓冲时间。尤其要关注页面大小与下载时间严重不成比例的组件，例如，正常情况下下载 10 KB 的页面需 0.02 s，50 KB 大小的页面应该在 0.1 s 左右，但是如果分解得出这个 50 KB 的页面下载时间为 0.5 s 或者更多，就可以初步判断该页面存在问题。

3) 页面下载大小图(Downloaded Components Size Graph)

针对上一步中初步确定的有问题的页面，通过查看该图，确定该页面中各种组件的大小及所占的比例关系。该图以饼图的形式直观展现组件所占的比例，确定尺寸比例最大的组件为考查的最关键因素。

4) 页面组件下载时间(Page Component Time Breakdown)

针对上一步中确定的尺寸比例最大的组件，通过进一步查看该图，可得到被测页面中最关注组件具体的下载时间信息的分解信息。该图以直方图的形式呈现，按照图 2 的时间分解方式进行进一步分解，可以看出该组件不同的时间分布，以确定是哪种时间模式存在问题。

5) 基于时间的页面组件细分(Component Breakdown(Over Time))

选择该图，以图形曲线的形式查看各组件在场景运行中的下载时间。通过查看所关注组件的曲线是否与主曲线轨迹基本重合，以确定该组件在所处页面中的时间表现上是否起主导作用。

6) 下载时间细分(Download Time Breakdown(Over Time))

选择该图，可以看出在场景运行中组件用在传输各个环节所占用的时间。如果各种时间分配比较平均，没有明显特别高或特别低的地方，属正常页面组件。但是如果发现网络传输占用的时间比服务器处理占用的时间要高出很多，即可以确定问题产生的原因是由网络传输引起的，反之亦然。

7) 第一次缓冲时间细分图(Time to First Buffer Breakdown)

在前面已经确定了存在问题的页面，但还需要确定问题到底是由服务器引起还是网络引起的。通过查看该图，可以确认问题的更确切的原因。

通过以上的步骤已经能够将问题锁定在尽可能小的范围内，但是如果要更深入地了解到底是由哪个更具体的模块引起的，就需要借助相关的(一般都是由应用软件自带的监控工具或者其他专用监控工具)工具来诊断，遵从以上的步骤，可一步步接近问题源。

3　结论

本文针对 JavaEE Web 应用系统在层次化的执行环境当中的服务器端性能问题，提出一种系统化的性能诊断思路和流程，从用户事务、系统资源、Web 资源和页面组件 4 个层次阐述性能诊断的策略。以用户交易响应时间为出发点，面向用户事务的表现进行判断，尽可能锁定几个突出的问题，基本确立下一步的诊断方向；然后，通过面向资源瓶颈的诊断，确定性能下降是否属系统资源瓶颈造成的；继而通过面向 Web 资源的诊断，进一步验证上 2 步的中间判断结果；最后，通过面向页面响应时间的诊断，找到响应时间比较长的页面，并进一步判断是页面哪些组件引发的延迟。

在具体的工程实践中，性能诊断更多地依靠工程人员的经验积累和敏锐的观察视角，因此性能诊断知识的积累非常重要[4]，需要不断地补充和完善。另外，随着技术的不断发展，JavaEE 框架本身和实现产品也不断推陈出新，在具体的 JavaEE Web 工程项目中往往应用了很多极具特色的新技术和第三方插件或产品，这些因素的加入更增加了性能诊断的难度[7-8]。此时仅从文中所述的 4 个层次是不够的，还需要更多针对性的测试工具和测试策略，遵循由表及里的大原则，逐层深入，从而找到性能的真正瓶颈。

参考文献:

[1] 陶以政,吴志杰,唐定勇,等. J2EE 构件化软件支撑平台研究与应用开发[J]. 信息与电子工程, 2009,7(3):247-251. (TAO Yizheng,WU Zhijie,TANG Dingyong,et al. Component software support platform based on Java 2 Enterprise Edition[J]. Information and Electronic Engineering, 2009,7(3):247-251.)

[2] Meier J D,Carlos Farre,Prashant Bansode,et al. Performance Testing Guidance for Web Applications[M]. Beijing:China Machine Press, 2008.

[3] Pressman R S,David Lowe. Web Engineering:A Practitioner's Approach[M]. Beijing:China Machine Press, 2010.

[4] 陈绍英,夏海涛,金成姬. Web 性能测试实战[M]. 北京:电子工业出版社, 2006. (CHEN Shaoying,XIA Haitao,JIN Cheng ji. Web Performance Test Model and Application[M]. Beijing:Publishing House of Electronics Industry, 2006.)

[5] 赵冲冲,白晓颖,王钊. 性能测试工具能力评估框架研究[J]. 计算机科学, 2006,33(3):244-248. (ZHAO Chongchong, BAI Xiaoying,WANG Zhao. A Framework for Evaluating the Capabilities of Performance Testing Tools[J]. Computer Science, 2006,33(3):244-248.)

[6] 姜艳,于波. LoadRunner 性能测试应用[M]. 北京:电子工业出版社, 2009. (JIANG Yan,YU Bo. Application of LoadRunner Performance Testing[M]. Beijing:Publishing House of Electronics Industry, 2009.)

[7] Connie Smith U,Lloyd Williams G. Performance Solutions:A Practical Guide to Creating Responsive, Scalable Software[M]. Beijing:China Machine Press, 2003.

[8] Ian Molyneaux. The Art of Application Performance Testing[M]. Beijing:China Machine Press, 2010.

作者简介

薛振伟(1978-),男,河南省濮阳市人,高级工程师,主要从事软件测试、性能工程等方面的研究. email:mailxzw@163.com.

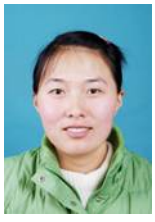
(上接第 212 页)

[6] Kay S M. Fundamentals of Statistical Signal Processing, vol.II: Detection Theory[M]. [S.l.]:Prentice Hall PTR, 1998.

[7] 杨祥龙,江波,吴为麟,等. 小信号检测中的自适应随机共振检测技术[J]. 信号处理, 2003,19(2):182-184. (YANG Xianglong,JIANG Bo,WU Weilin,et al. Adaptive Stochastic Resonance Technology in detecting Weak Signal[J]. Signal Processing, 2003,19(2):182-184.)

[8] 刘利姣,黄光明,杜茜,等. 大参数随机共振的两种方法及数值仿真[J]. 信息与电子工程, 2007,5(3):182-185. (LIU Lijiao,HUANG Guangming,DU Qian,et al. Two Methods of the Large Parameter Signal Stochastic Resonance and Numerical Simulation[J]. Information and Electronic Engineering, 2007,5(3):182-185.)

作者简介:



蔡卫菊(1981-),女,湖北荆州人,硕士,讲师,主要研究方向为信号处理.email:caicai322@126.com.