(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101505243 B
(45)授权公告日 2011.01.05

(21)申请号 200910079404.3

(22)申请日 2009.03.10

(73)专利权人 中国科学院软件研究所
地址 100190 北京市海淀区中关村南四街 4 号

(72)发明人 王伟 宋云奎 张文博 魏峻 钟华 黄涛

(74)专利代理机构 北京君尚知识产权代理事务所(普通合伙) 11200
代理人 余长江

(51)Int.Cl.
H04L 12/26(2006.01)
H04L 12/24(2006.01)
H04L 29/08(2006.01)

(56)对比文件
CN 101364185 A,2009.02.11,
CN 1763778 A,2006.04.26,
US 2002157035 A1,2002.10.24,
CN 1909551 A,2007.02.07,

审查员 胡丽丽

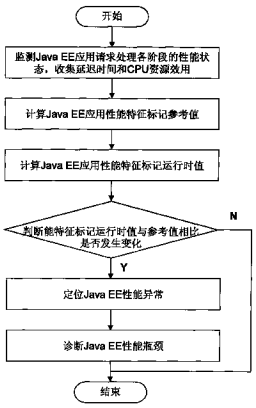
权利要求书 1 页 说明书 9 页 附图 4 页

(54)发明名称

一种 Web 应用性能异常侦测方法

(57)摘要

本发明公开了一种 Web 应用性能异常侦测方法,属于软件技术领域。本发明的方法为:1)设置 Web 应用客户端请求处理过程中各阶段的性能特征标记参考值;2)收集各请求处理阶段运行时的延迟时间和系统资源使用率,计算对应阶段的性能特征标记运行值;3)比较性能特征标记运行值与对应的性能特征标记参考值,确定出异常阶段;4)根据确定的异常阶段和该阶段的系统资源使用率,确定出现异常的系统资源。与现有技术相比,本发明方法简单、更易维护和实施,能够实现性能异常的精确捕获和定位,从而为优化业务逻辑和资源配置提供依据。



CN 101505243 B

1. 一种 Web 应用性能异常侦测方法，其步骤为：

1) 设置 Web 应用客户端请求处理各阶段的性能特征标记参考值；所述性能特征标记为服务时间；

2) 收集各请求处理阶段运行时的延迟时间和系统资源使用率，计算对应阶段的性能特征标记运行值；所述延迟时间为：请求处理阶段系统管理时间与服务时间之和，即因系统排队、调度所耗费的时间与实际执行时间之和；

3) 比较性能特征标记运行值与对应的性能特征标记参考值，确定出异常阶段；

4) 根据确定的异常阶段和该阶段的系统资源使用率，确定出现异常的系统资源。

2. 如权利要求 1 所述的方法，其特征在于在默认中间件配置下收集各请求处理阶段内监测目标的延迟时间和系统资源使用率，计算所述服务时间参考值。

3. 如权利要求 2 所述的方法，其特征在于所述服务时间参考值的计算方法为：

1) 设置监测目标的参考特征采样时间段，并将其划分为若干个采样窗口；

2) 在参考特征采样时间段内，统计每一采样窗口完成处理的请求计数，并计算该采样窗口内延迟时间和系统资源使用率；

3) 对该监测目标的监测数据进行分类，将具有相同系统资源使用率的延迟时间和请求计数归为一类；

4) 将延迟时间对应的请求计数作为权值，计算同一类的延迟时间加权平均值，得到一组系统资源使用率与延迟时间加权平均值的值对；

5) 基于经典排队论，对每一个系统资源使用率与延迟时间加权平均值的值对进行计算，得到该监测目标的一组服务时间样本；

6) 利用累计分布函数对服务时间样本进行统计，选择累计分布函数为 50％的值作为所述服务时间参考值。

4. 如权利要求 1 所述的方法，其特征在于所述服务时间运行值的计算方法为：

1) 设置监测目标的特征采样时间段，并将其划分为若干个采样窗口；

2) 在每一特征采样时间段内，统计每个采样窗口完成处理的请求计数，计算该采样窗口内延迟时间和系统资源使用率；

3) 对该监测目标的监测数据进行分类，将具有相同系统资源使用率的延迟时间和请求计数归为一类；

4) 将延迟时间对应的请求计数作为权值，计算同一类的延迟时间的加权平均值，得到一组系统资源使用率与延迟时间加权平均值的值对；

5) 基于经典排队论，对每一个系统资源使用率与延迟时间加权平均值的值对进行计算，得到该监测目标的一组服务时间样本；

6) 利用累计分布函数对服务时间样本进行统计，选择累计分布函数为 50％的值作为所述服务时间运行值。

5. 如权利要求 4 所述的方法，其特征在于采用公式 S ＝ R*(1-U) 对每一个系统资源使用率与延迟时间加权平均值的值对进行计算，得到该监测目标的一组服务时间样本；其中 S 为服务时间、R 为延迟时间、U 为资源使用率。

一种 Web 应用性能异常侦测方法

技术领域

[0001] 本发明涉及一种 Web 应用性能异常侦测方法，尤其涉及一种通过 Web 应用的请求服务时间进行性能异常侦测进而发现性能瓶颈的方法，属于软件技术领域。

背景技术

[0002] 多层架构的 Web 应用已成为当前主流的网络应用，大量关键应用（电子银行、网上支付等等）采用 Web 应用实施，系统服务质量 (QoS) 保障十分重要。目前，Web 应用的 QoS 度量主要关注 Web 应用的响应性和吞吐量等性能度量，因此 Web 应用的性能分析，包括异常侦测、问题诊断等，对于服务质量保障至关重要。但是，随着 IT 系统的规模和复杂度逐渐增加，Web 应用及中间件的动态复杂性增加了性能分析的难度和代价。一方面，对于多层架构的 Web 应用，虽然通过中间件屏蔽底层异构性简化了开发，但中间件自身的系统复杂性为性能分析引入新的挑战。如图 1 所示，中间件系统提供了多种特色服务（如线程池、调度队列、组件实例池、数据库连接池等服务），其内部架构和参数配置十分复杂，影响了性能问题的分析。同时，由于 Web 应用通常采用组件模型，客户端的一次请求可能涉及多个相互协作和依赖的组件，性能问题分析的难度也因此增加。另一方面，Web 应用的版本以及部署环境的变化等动态因素影响性能分析方法的适应性。

[0003] 上述 Web 应用及中间件的动态复杂性直接导致性能异常侦测成本的提高。首先，难以确定需要侦测的性能特征度量。其次，如何处理在线监测结果，获得性能特征，也是需要解决的问题。如果对系统实施全面的监测，一方面势必造成严重的系统额外开销，影响正常运行，另一方面如果缺乏有效的数据处理和分析方法，监测获得的大量数据将变得毫无意义。如同气象预报，需要关键的气象数据（气温、湿度、气压、风向等）与合理的气象模型才能实现天气的准确预报，性能异常侦测也需要遴选性能特征度量，并在此基础上建立合理的度量模型。

[0004] 在分布式系统的性能异常侦测和诊断方面存在较多的研究工作，可归类为基于阈值 (threshold) 的方法和基于模型的方法。

[0005] 在基于阈值的方法中，阈值的设置尤其关键，往往依赖经验和应用系统的部署环境，很难配置一个准确、固定的阈值，直接影响方法的适用性和有效性。HP(Mercury Diagnostics. http://www.mercury.com/us/products/diagnostics/)，IBM(IBM Corporation. Tivoli WebManagement Solutions，http://www-01.ibm.com/software/tivoli/)，Wily(CA Wily Introscope. http://www.ca.com/us/application-management.aspx) 等采用预设静态阈值的侦测方法，当性能指标测量值超出阈值时，监测系统发出警报，或按照设定策略执行预定动作。BMC(BMCProactiveNet Analytics. http://www.bmc.com/) 和 Netutive(NETUITIVE SI. http://www.netuitive.com/products/netuitive-si.php) 使用基于机器学习的动态阈值技术，降低了假警报的发生率，具有较高的准确性，但当环境发生变化时，要重新执行复杂和耗时的机器学习过程。

[0006] 在基于模型的方法中，一些工作 (L. Cherkasova, Y. Fu, W. Tang, A. Vahdat：

Measuringand Characterizing End-to-End Internet Service Performance.Journal ACM/IEEE Transactionson Internet Technology, (TOIT), November,2003 ;David Olshefski, Jason Nieh, Understandingthe management of cllent perceived response time, ACM SIGMETRICS Performance EvaluationReview, v.34n.1, June 2006 ;David Olshefski, Jason Nieh, Dakshi Agrawal, Using certes toinfer client response time at the web server, ACM Transactions on Computer Systems(TOCS), v.22n.1, p.49-93,February 2004[doi ＞ 10.1145/966785.966787]) 使用基于端到端 (end-to-end) 的请求延迟监测方法对多层架构的分布式系统进行监测，这些工作将客户端请求划分为客户端网络相关与服务器相关两部分，侧重分析网络延迟造成的性能问题，并没有对服务器内部的性能问题做进一步剖析。另一些工作则关注服务器端的性能问题，利用事务跟踪技术对服务器端的请求处理过程进行阶段划分和性能监测。Pinpoint(M.Chen, E.Kcman, E.Fratkin, E.Brewer, and A.Fox.Pinpoint :Problem Determination in Large, Dynamic Internet Services.In Symposium on Dependable Networks and Systems(IPDS Track),2002) 计算客户请求的 Jaccard 相似度关联，通过数据聚类分析来关联失败请求与相关组件，但该工作无法侦测资源瓶颈造成性能异常。Cohen 等人 (I.Cohen, S.Zhang, M.Goldszmidt, J.Symons, T.Kelly, A.Fox.Capturing, Indexing, Clustering, and RetrievingSystem History.In Proc.20th ACM Symposium on Operating Systems Principles,2005 ;Agarwala, S., Schwan, K. :Sysprof :Online distributed behavior diagnosis through fine-grainsystem monitoring.In :ICDCS(2006)) 利用统计学方法对收集的分布式系统底层度量指标历史数据进行建模，分析性能问题，由于缺乏应用级和中间件级的度量信息，因此无法侦测、区分应用和中间件造成的性能异常。Barham 等人 (P.Barham, A.Donnelly, R.Isaacs, R.Mortier.Using Magpie for request extraction and workload modelling December 2004 6thSymposium on Operating Systems Design and Implementation(OSDI'04),2004 ;Indicative Co.http://www.indicative.com/products/End-to-End.pdf ;Quest Software Inc.Performasure.http://java.quest.com/performasure.) 通过捕捉分布式系统中应用组件的延迟时间，在应用级和中间件级构造请求行为的统计模型。

[0007] 在度量指标方面，已有工作主要采用延迟时间，但受系统负载和部署环境变化的影响，延迟时间作为性能特征度量不具有稳定性和适应性。

发明内容

[0008] 本发明的目的在于克服现有技术中存在的问题，提供一种资源敏感的 Web 应用性能异常侦测方法。在本发明方法中，为了进行性能异常的侦测和定位，中间件处理客户端请求的过程被划分为一系列阶段。本发明方法对各阶段的处理过程进行监测，并结合资源使用情况计算各阶段的性能特征标记，得到 Web 应用的性能特征标记。本发明将性能特征标记作为判断性能异常和瓶颈的度量指标。对于多层架构的 Web 应用，性能特征标记可应用于中间件提供的各类服务（如线程池、队列、实例池、连接池等服务），也可以应用于 Web 应用的各类应用组件模型（如 JSP/Servlet 组件、EJB 组件等）。进一步，通过监测性能特征标记的变化，进行 Web 应用性能异常的捕获、定位以及性能瓶颈的诊断。

[0009] 在对中间件处理客户端请求的各阶段进行监测时，本发明方法主要包括以下两个监测度量：阶段执行的延迟时间 (Latency) 和系统资源效用 (Utility)。下面对这些度量作详细介绍：

[0010] (1) 延迟时间

[0011] 在本发明方法中，延迟时间指一个 Web 应用的请求处理阶段所经历的时间。延迟时间反映的是请求处理阶段的响应时间，是该阶段系统管理时间与服务时间 (ServiceTime) 之和，即因系统排队、调度所耗费的时间与实际执行时间之和。延迟时间受系统资源分配的影响，当系统负载较大时，系统资源的竞争将造成管理时间的变化，进而造成延迟时间的变化。

[0012] (2) 系统资源效用

[0013] 在本发明方法中，系统资源效用表示某种系统资源的使用程度，如系统的处理器 (CPU)、内存 (Memory)、磁盘 (Disks) 和网络 (Network) 等资源的使用率。

[0014] 在上述监测度量基础上，本发明通过计算性能特征标记进行 Web 应用性能异常的捕获、定位以及性能瓶颈的诊断。在本发明方法中，性能特征标记即为服务时间，它表示某一请求处理阶段内实际使用系统资源的时间，不包括因系统排队、调度等耗费的系统管理时间；对于请求处理的各阶段，阶段内包括一个监测目标，该阶段的性能特征标记值为该阶段内监测目标的性能特征标记值（本发明中为服务时间）。服务时间的大小与请求处理阶段的逻辑以及系统的资源相关，当请求处理阶段的逻辑和系统资源能力确定时，服务时间不会随系统负载的变化而变化。当请求阶段的处理逻辑或系统资源发生变化时（如应用组件更新造成业务逻辑发生变化、服务器 CPU 升级造成系统资源能力发生变化），服务时间随之发生变化。服务时间可以通过调用系统底层方法直接测量，但这种方式具有平台相关性，不易实施，并且会造成严重的系统额外负担。本发明方法基于经典排队理论，通过延迟时间和系统资源效用的测量值计算服务时间。当系统出现性能瓶颈时，系统资源效用出现异常，即资源效用达 100％，致使资源竞争，或者资源效用小于 100％并保持稳定，致使资源无法充分利用。资源效用的异常导致计算得到服务时间发生变化，本发明方法通过服务时间是否变化判断系统是否出现性能瓶颈。

[0015] 在性能特征标记基础上，一种资源敏感的 Web 应用性能异常侦测方法，包括以下各步骤：

[0016] 1) 对 Web 应用客户端请求处理的各阶段的监测目标进行监测，收集延迟时间和系统资源效用。阶段是组成中间件处理客户端请求过程的功能单元，细粒度的划分有利用性能异常的精确捕获和定位。

[0017] 2) 计算 Web 应用性能特征标记参考值。在默认中间件配置以及低负载环境下，设置一参考特征采样时间段，并将其划分为若干个采样窗口；在一个参考特征采样时间段内，对于 Web 应用请求处理的不同阶段，利用步骤 1) 收集的性能状态数据。在一个参考特征采样时间段结束时，计算性能特征标记，获得一组反映 Web 应用性能状态的特征标记参考值。例如，反映线程池性能的特征标记，反映不同应用组件性能的特征标记，反映数据库连接池性能的特征标记，反映 SQL 语句执行性能的特征标记等等。Web 应用性能特征标记组表征了该 Web 应用的客户端请求在不同处理阶段的性能特征，Web 应用性能特征标记参考值反映了 Web 应用正常运行时的性能特征。

[0018] 3) 计算 Web 应用性能特征标记运行时值，进行性能异常的捕获和定位。设置一特征采样时间段，并将其划分为若干个采样窗口；在 Web 应用运行时的每一个特征采样时间内，利用步骤 1) 收集的性能状态数据。在每一个特征采样时间段结束时，计算性能特征标记，获得一组反映该特征采样时间段的 Web 应用性能状态的特征标记运行时值。将每一组 Web 应用性能特征标记组与参考值比较，如果特征标记发生变化，则捕获到性能异常，否则不存在性能异常。在捕获到性能异常后，可以通过发生变化的特征标记的位置定位异常出现的阶段。

[0019] 4) 性能瓶颈的诊断。Web 应用性能特征标记的计算与系统资源效用相关，Web 应用性能特征标记的参考值反映 Web 应用在未出现性能瓶颈时的性能特征，当系统出现性能瓶颈时，Web 应用的性能特征标记将发生变化。以此作为诊断依据，通过分析步骤 3) 捕获和定位的性能异常，可以诊断出在哪一阶段出现性能瓶颈，同时根据步骤 1) 的监测结果，将出现性能瓶颈的阶段（异常阶段）对应的系统资源数据返回给用户，使用者根据返回的步骤 1) 中对不同系统资源进行监测的数据，则可以诊断出在某一阶段何种系统资源出现瓶颈，比如在阶段 N，使用 CPU 资源计算性能特征，那么出现异常时，可以判断阶段 N 存在 CPU 瓶颈。

[0020] 在本发明方法中，步骤 2) 和 3) 中计算 Web 应用性能特征标记的方法包括如下步骤：

[0021] (1) 在特征采样时间段（或参考特征采样时间段）的每个采样窗口内，统计每个采样窗口完成处理的请求计数，计算该采样窗口内延迟时间的平均值和系统资源效用的平均值。

[0022] (2) 在特征采样时间段（或参考特征采样时间段）结束时，对步骤 (1) 得到数据进行进一步统计整理。对于某一请求处理阶段内的监测目标的监测数据，具有相同的系统资源效用值的采样窗口，其对应的延迟时间和请求计数被归为一类。归类的目的是为了将延迟时间与资源效用相关联，同时，利用请求计数体现该采样窗口内收集得到的延迟时间是否具有代表性。

[0023] (3) 在特征采样时间段（或参考特征采样时间段）结束时，对每个系统资源效用分类中的所有延迟时间和请求计数值对进行进一步统计整理。将延迟时间对应的请求计数作为权值，计算同一类的延迟时间的加权平均值，得到一组系统资源效用与延迟时间（计算加权平均值后）的值对。

[0024] (4) 基于经典排队论，对步骤 (3) 中得到的每一个系统资源效用与延迟时间的值对进行计算，得到一组服务时间样本。

[0025] (5) 对步骤 (4) 中得到某一请求处理阶段内的监测目标的一组服务时间样本，利用累计分布函数 CDF(Cumulative Distribution Function) 进行统计，选择 CDF 为 50％的值作为服务时间的近似值，即某一请求处理阶段的监测目标的性能特征标记。在本发明方法中，与服务时间样本的平均值相比，采用 CDF 得到的近似值不受采样异常值的影响，更能反映该处理阶段的性能特征。

[0026] 与现有技术相比，本发明具有如下技术优势：

[0027] 1、将中间件处理客户端请求的过程划分为一系列阶段，通过细粒度的阶段划分实现性能异常的精确捕获和定位。

[0028] 2、对系统资源效用进行监测，结合资源使用情况，进行性能瓶颈的诊断，实现瓶颈资源的发现和定位。与基于阈值的方法相比，本发明方法与服务器硬件环境无关，不易受服务器硬件变更的影响，从而降低了人工参与的成本。与基于模型的方法相比，本发明方法没有采用复杂的统计学方法以及机器学习方法进行建模，方法简单，更易实施。

[0029] 3、本发明可为 Web 应用以及中间件的性能调优提供支持。通过对 Web 应用的不同业务组件进行侦测，帮助 Web 应用开发人员及时发现业务组件中的性能瓶颈，为进一步优化业务逻辑提供依据；通过对中间件提供的各类服务进行侦测，帮助中间件管理人员及时发现性能瓶颈，为进一步的优化配置提供依据。

附图说明

[0030] 图 1 一个简单的 Java EE 应用——网上购物系统示例图。

[0031] 图 2Java EE 应用服务器处理客户端请求 Servlet 组件的过程示例图。

[0032] 图 3Java EE 应用性能异常侦测方法流程图。

[0033] 图 4 利用监测器收集性能状态数据示例图。

[0034] 图 5 特征采样时间段内服务时间计算方法流程图。

具体实施方式

[0035] 以下结合具体实施例和附图对本发明进行详细说明。

[0036] 本发明提出的资源敏感的 Web 应用性能异常侦测方法通过收集 Web 应用以及中间件的相关监测数据，刻画性能特征标记，实现性能异常的捕获、定位以及性能瓶颈的诊断。

[0037] 作为本实施例方法的使用环境，所述 Web 应用采用一个简单的 Java EE 应用。Java EE(Java™ Platform，Enterprise Edition) 是 Sun 公司提出的开发、部署、运行和管理 Java 分布式应用的标准技术体系结构，它包括一系列应用组件模型和标准服务。本实施例采用的 Java EE 应用主要使用了 Servlet 组件模型和数据库连接服务。Servlet 是一种 Java EE Web 组件，它与客户端采用“请求 / 响应”的通信模式，当客户端请求某一 Servlet 组件时，该组件可以产生动态网页内容并作为响应返回客户端。数据库连接服务提供标准的数据库编程接口，为应用组件调用和执行 SQL 语句提供支持。

[0038] 本实施例所采用 Java EE 应用是一个简单的网上购物系统，包括商品浏览、商品订购、订单确认等功能。如图 1 所示，上述功能分别由 Search、ShoppingCart、Order 三个 Servlet 组件完成，用户通过客户端浏览器请求 Search 组件进行商品浏览和查询，通过请求 ShoppingCart 组件将商品加入购物篮，通过请求 Order 组件完成订单的确认。上述三个组件都需要利用数据库连接服务进行数据库操作，Search 组件需要进行数据库查询操作，ShoppingCart 组件和 Order 组件需要进行数据库写操作。

[0039] 作为本实施例方法的使用环境，所述中间件采用 Java EE 应用服务器。Java EE 应用服务器是开发、配置和管理 Java EE 应用的标准平台，它通过容器来支持分层体系结构。容器提供了对 Java EE 应用组件的运行时支持，其中，Servlet 组件由 Servlet 容器进行管理。Servlet 容器封装了 Web 服务器与表示层逻辑的功能，负责 Servlet 组件与客户的通信以及 Servlet 组件方法的调用。同时，Java EE 应用服务器还提供了一系列的底层服务（如数据库连接服务等）为容器提供底层功能支持。除了标准的 Java EE 容器和服务，本实施

例所采用的 Java EE 应用服务器还支持线程池、数据库连接池等资源池服务。资源池服务可以提高资源的利用率、降低资源重新创建的开销、控制资源的并发访问。由于资源管理对应用服务器的性能至关重要，所以资源池服务已成为所有应用服务器都支持的一类基本服务。

[0040] 本实施例所采用的 Java EE 应用服务器，其处理客户端请求 Servlet 组件的过程如图 2 所示，当用户通过客户端浏览器访问服务器端的某个 Servlet 组件进行数据库查询时，请求处理过程主要包括以下各阶段：

[0041] 1) 当 Java EE 应用服务器接收到客户端请求后，从线程池分配一个线程处理客户端请求；

[0042] 2) 执行客户端请求的 Servlet 组件；

[0043] 3) 从数据库连接池获得一个数据库连接；

[0044] 4) 执行 SQL 语句并返回结果；

[0045] 针对上述请求处理阶段，本实施例方法首先对各阶段的性能状态进行监测度量。接着利用度量值计算各阶段性能特征标记，并将性能特征标记作为判断性能异常和瓶颈的度量指标。对于本实施例采用的简单 Java EE 应用和 Java EE 应用服务器，性能特征标记分别应用于 Java EE 应用的三个应用组件 (Search、ShoppingCart 和 Order) 以及 Java EE 应用服务器提供的资源池服务（线程池服务和数据库连接池服务）监测目标上。最后，通过对比性能特征标记的参考值与运行时值，判断上述目标是否出现性能异常，并对性能瓶颈进行诊断。

[0046] 本实施例方法流程如图 3 所示：

[0047] 1、对 Java EE 应用的客户端请求处理的各阶段进行监测，收集延迟时间和 CPU 资源效用。由于本实施例采用的简单 Java EE 应用的关键系统资源是 CPU 资源，因此本实施例方法选择对 CPU 资源效用进行监测。对于其它 Java EE 应用，可以选择其对应的关键资源进行监测，如（内存、磁盘等）。如图 4 所示，本实施例方法通过延迟时间监测器和 CPU 资源监测器对 JavaEE 应用服务器处理客户端请求的各阶段进行监测，JavaEE 应用性能异常侦测服务器对各监测器获得的数据进行收集和处理。其中，延迟时间监测器对各阶段执行的延迟时间进行监测，CPU 资源监测器对系统 CPU 资源效用进行监测。Java EE 应用服务器产品提供了线程池、Java EE 组件、数据库连接池等处理阶段的延迟时间的监测接口，延迟时间监测器通过周期性地访问接口收集延迟时间。在本实施例方法中，CPU 资源效用即为 CPU 的使用率。操作系统提供了 CPU 资源使用率的访问接口，CPU 资源监测器通过周期性地访问接口收集 CPU 资源使用率。

[0048] 2、在 Java EE 应用服务器默认配置以及低负载环境下，对于 Java EE 应用请求处理的各阶段的监测目标，整理监测器收集到的监测数据，计算服务时间，得到一组 Java EE 应用性能特征标记的参考值，参考值反映了 Java EE 应用正常运行时的性能特征。其中，在一个特征采样时间段内，计算服务时间的步骤如图 5 所示，包括以下各步骤：

[0049] (1) 一个特征采样时间段可以划分为若干采样窗口，在特征采样时间段内，统计每个窗口内完成处理的请求计数，并计算该采样窗口的平均延迟时间（以下简称延迟时间）和平均 CPU 资源使用率（以下简称 CPU 资源使用率）。

[0050] (2) 在特征采样时间段结束时，对步骤 (1) 得到数据进行进一步统计整理。首先对

CPU 资源效用进行分类，表示为 $\{U_1 = 1\%, U_2 = 2\%, \ldots, U_i = i\%, U_{100} = 100\%\}$，其中 U 表示 CPU 资源使用率，i 表示类别。对于某一阶段内的某目标的监测数据，具有相同 CPU 资源效用值的延迟时间和请求计数被归为一类。例如，对于应用组件处理阶段内的 Order 组件的监测数据，在某一采样窗口内，存在 N 次请求处理，延迟时间为 R，且 CPU 资源使用率为 10%，则 (N^{order}, R^{order}) 作为值对被归为类别 i = 10 中。

[0051] (3) 在特征采样时间段结束时，对每个 CPU 资源效用分类中的所有 (N,R) 值对进行进一步统计整理。将延迟时间对应的请求计数 N 作为权值，计算同一类的延迟时间的加权平均值，得到一组 CPU 资源效用与延迟时间（计算加权平均值后）的值对 $(U_i^{order}, R_i^{order})$。

[0052] (4) 利用经典排队论，对步骤 (3) 中得到的每一个 CPU 资源效用与延迟时间的值对进行计算，得到一组服务时间样本。对于在 Java EE 应用服务器中各阶段的请求处理任务可以用一个简单的队列系统描述，存在等式 (1)：

[0053] $R = S+S*Q$ (1)

[0054] 其中 S 代表系统中某一个任务的平均服务时间，Q 代表平均队列长度，R 代表任务在系统中的平均延迟时间。设 T 代表系统平均吞吐量，根据 Little's 法则 (E.D.Lazowska, J.Zahorjan, G.S.Graham and K.C.Sevcik.Quantitative system performance :computer systemanalysis using queueing network models.Prentice-Hall, Inc.Upper Saddle River, NJ, USA, 1984)，$Q = T*R$，可以得到等式 (2)：

[0055] $R = S+S*(T*R)$ (2)

[0056] 进一步根据 Utilization 法则 (E.D.Lazowska, J.Zahorjan, G.S.Graham and K.C.Sevcik.Quantitative system performance :computer system analysis using queueing networkmodels.Prentice-Hall, Inc.Upper Saddle River, NJ, USA, 1984)，资源效用等于吞吐量乘以服务时间，即 $U = T*S$，可推导出等式 (3)：

[0057] $Q = U/(1-U)$ (3)

[0058] 等式 (3) 说明了队列系统队列中平均任务数与服务器效用之间的关系。对于在 Java EE 应用服务器中执行的不同阶段的请求处理任务，由于典型的计算机操作系统在处理多个任务时使用分时规则，因此存在等式 (4)：

[0059] $R = S*N$ (4)

[0060] 其中 N 代表并发执行的任务属，且 $N = Q+1$，用等式 (3) 的结果替换 Q，得到计算服务时间的等式 (5)：

[0061] $S = R*(1-U)$ (5)

[0062] 由此得到服务时间 S、延迟时间 R 以及资源效用 U 之间的关系。随着负载的增加，当系统出现性能瓶颈时，系统资源效用出现异常，即资源效用达 100%，致使资源竞争，或者资源效用小于 100%，保持稳定，致使资源无法充分利用。资源效用的上述异常引起管理时间增长，进而引起延迟时间增长，最终导致由等式 (5) 计算的服务时间增长。

[0063] 利用等式 (5) 对步骤 (3) 中得到的每一个 CPU 资源效用与延迟时间的值对进行计算，得到一组服务时间样本 $\{S_1, S_2, \ldots, S_i, \ldots, S_{100}\}$，其中 i 表示 CPU 资源效用类别。

[0064] (5) 利用 CDF 处理步骤 (4) 中得到某一请求处理阶段内的监测目标的一组服务时间样本，选择 CDF 为 50% 的值作为服务时间的近似值，该近似值即为某一请求处理阶段的监测目标的性能特征标记。例如应用组件处理阶段内的 Order 组件的性能特征标记。在本

实施例中，利用上述性能特征标记计算方法可以获得一组反映该 Java EE 应用性能状态的特征标记，该组特征标记表示某一类型的请求处理过程中各个阶段的性能状态，如表 1 所示。

[0065] 表 1Java EE 应用性能特征标记

[0066]

请求处理阶段	监测目标	性能特征标记
线程池	Java EE应用服务器线程池	$S^{threadpool}$
应用组件	Search组件	S^{search}
	ShoppingCart组件	$S^{shoppingcart}$
	Order组件	S^{order}
数据库连接池	Java EE应用服务器数据库连接池	$S^{dbconnectionpool}$
SQL语句执行	Search组件的SQL语句	$S^{searchsql}$
	ShoppingCart组件的SQL语句	$S^{shoppingcartsql}$
	Order组件的SQL语句	$S^{ordersql}$

[0067] 3、性能异常的捕获和定位。在 Java EE 应用运行时，通过步骤 2) 中所述服务时间计算方法，得到 Java EE 应用性能特征标记的运行时值。将每一个特征采样时间段获得的 Java EE 应用性能特征标记组与参考值比较，如果特征标记的运行时值发生变化，则捕获到性能异常，同时可以定位异常出现的阶段，否则不存在性能异常。在本实施例方法中，将特征标记的运行时值与参考值的相对偏差是否大于 10％作为判断运行时值是否发生变化的判断条件，可以通过调整该相对偏差的设置来获得不同的侦测灵敏度。本实施例分别人为设置了三个性能瓶颈，包括 Java EE 应用服务器线程池容量过小、Java EE 应用服务器数据库连接池容量过小以及数据库服务器 CPU 性能过低。表 2 显示了在分别设置上述三种性能瓶颈后获得的性能特征标记，其中：在线程池过小情况下，$S^{threadpool}$ 出现异常；在数据库连接池过小情况下，$S^{dbconnectionpool}$ 出现异常；在数据库服务器 CPU 性能过低情况下，$S^{searchsql}$、$S^{shoppingcartsql}$、$S^{ordersql}$ 均出现异常。

[0068] 表 2Java EE 应用性能特征标记参考值与运行时值

[0069]

性能特征标记	参考值	线程池过小	数据库连接池过小	数据库服务器 CPU 性能过低
$S^{threadpool}$	5ms	820ms	5ms	5ms
S^{search}	59ms	61ms	62ms	58ms

$S^{shoppingcart}$	72ms	75ms	73ms	69ms
S^{order}	81ms	86ms	85ms	80ms
$S^{dbconnectionpool}$	9ms	8ms	1272ms	10ms
$S^{searchsql}$	139ms	143ms	138ms	590ms
$S^{shoppingcartsql}$	439ms	442ms	425ms	2089ms
$S^{ordersql}$	832ms	843ms	823ms	3626ms

[0070] 4、性能瓶颈的诊断。Java EE 应用性能特征标记的计算与 CPU 资源效用相关，Java EE 应用性能特征标记的参考值反映 Java EE 应用在 CPU 资源未出现瓶颈时的性能特征，当 CPU 资源出现瓶颈时，Java EE 应用的性能特征标记将发生变化。例如，在三种人为设置的性能瓶颈情况下，对表 2 中的相应的三组性能特征标记进行分析：对于线程池过小的情况，$S^{threadpool}$ 出现异常，而其它性能特征标记未出现异常时，则表明线程池出现瓶颈，进而可推测根本原因是由于线程池设置过小造成；对于数据库连接池过小的情况，$S^{dbconnectionpool}$ 出现异常，而其它性能特征标记未出现异常时，则表明数据库连接池出现瓶颈，进而可推测根本原因是由于数据库连接池设置过小造成；对于数据库服务器 CPU 性能过低的情况，$S^{searchsql}$、$S^{shoppingcartsql}$ 和 $S^{ordersql}$ 出现异常时，则表明数据库服务器的 CPU 资源出现瓶颈。


网上购物系统
Servlet组件：Search
Servlet组件：ShoppingCart
Servlet组件：Order

图 1

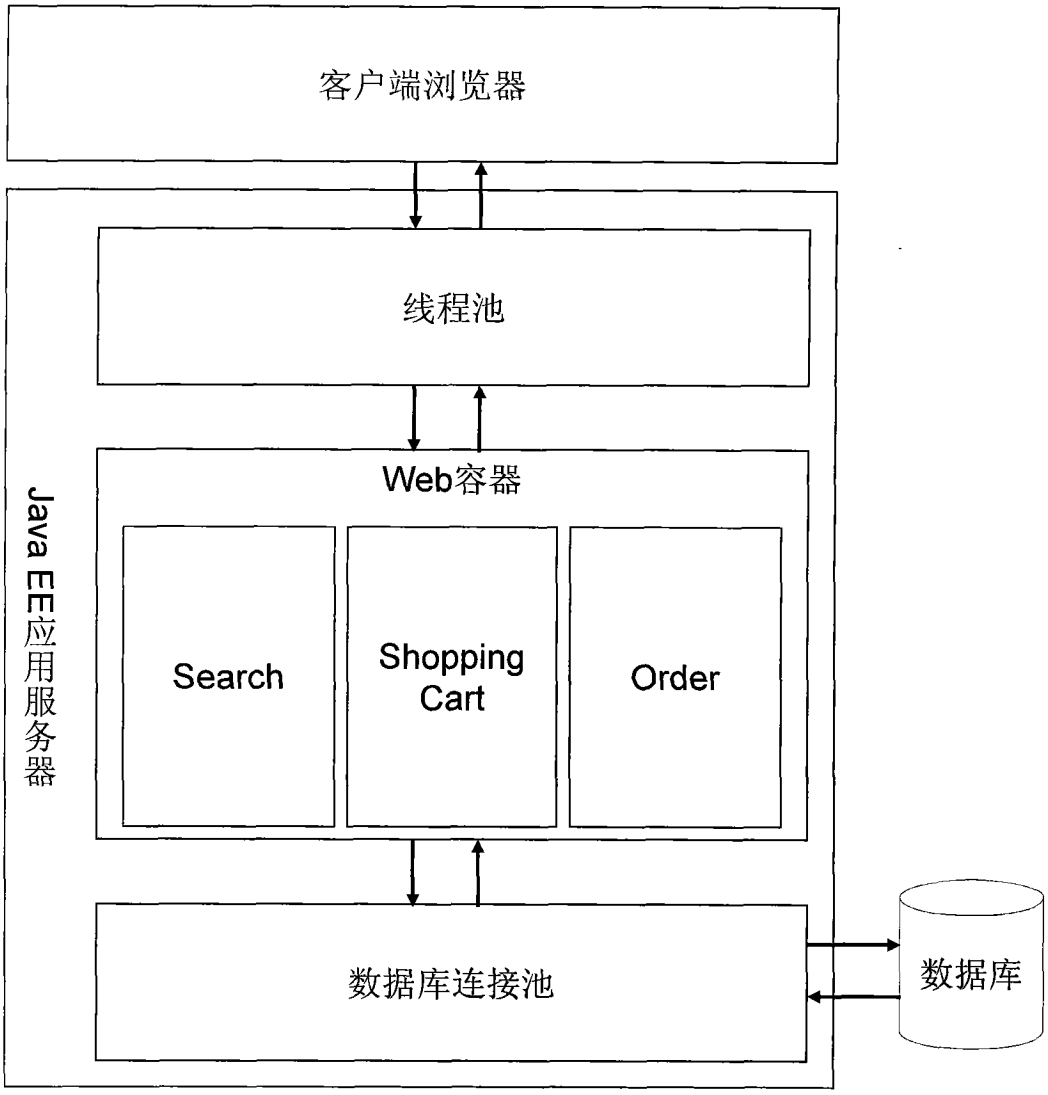

客户端浏览器
线程池
Web容器
Search
Shopping Cart
Order
Java EE应用服务器
数据库连接池
数据库

图 2

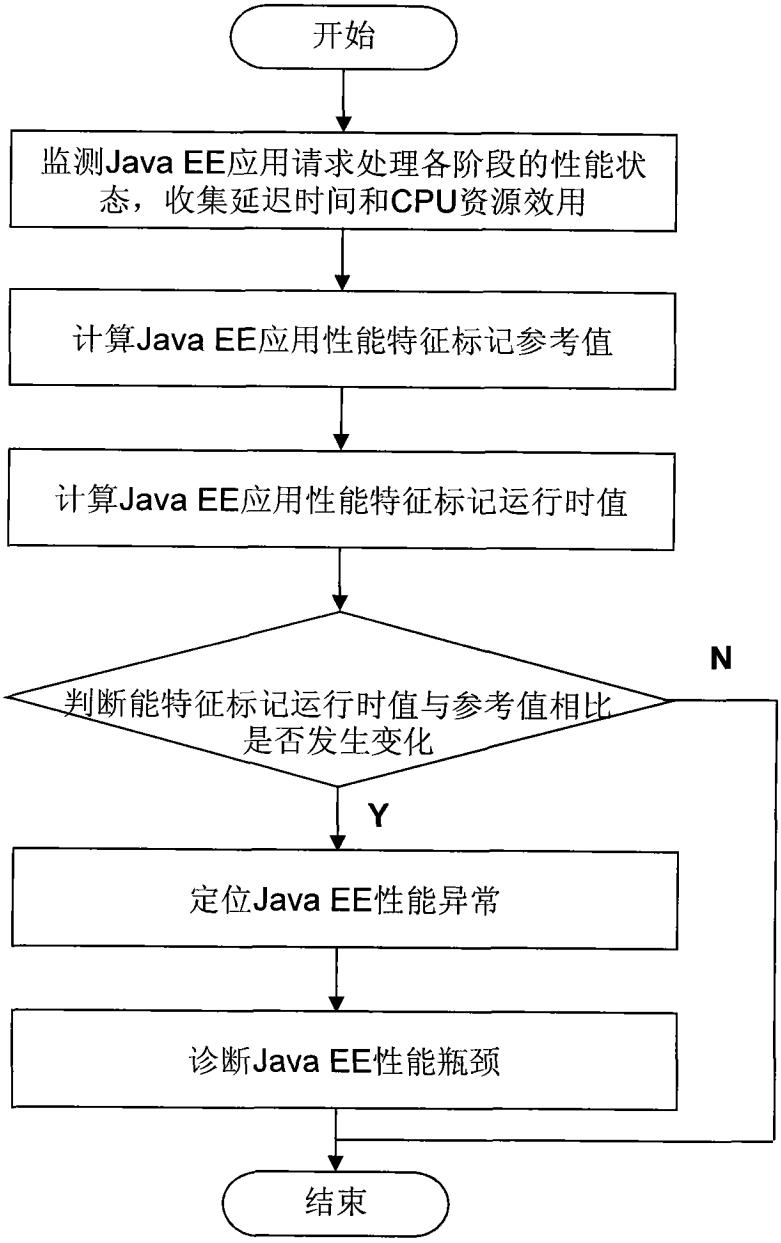

开始
监测Java EE应用请求处理各阶段的性能状态，收集延迟时间和CPU资源效用
计算Java EE应用性能特征标记参考值
计算Java EE应用性能特征标记运行时值
判断能特征标记运行时值与参考值相比是否发生变化
N
Y
定位Java EE性能异常
诊断Java EE性能瓶颈
结束

图 3

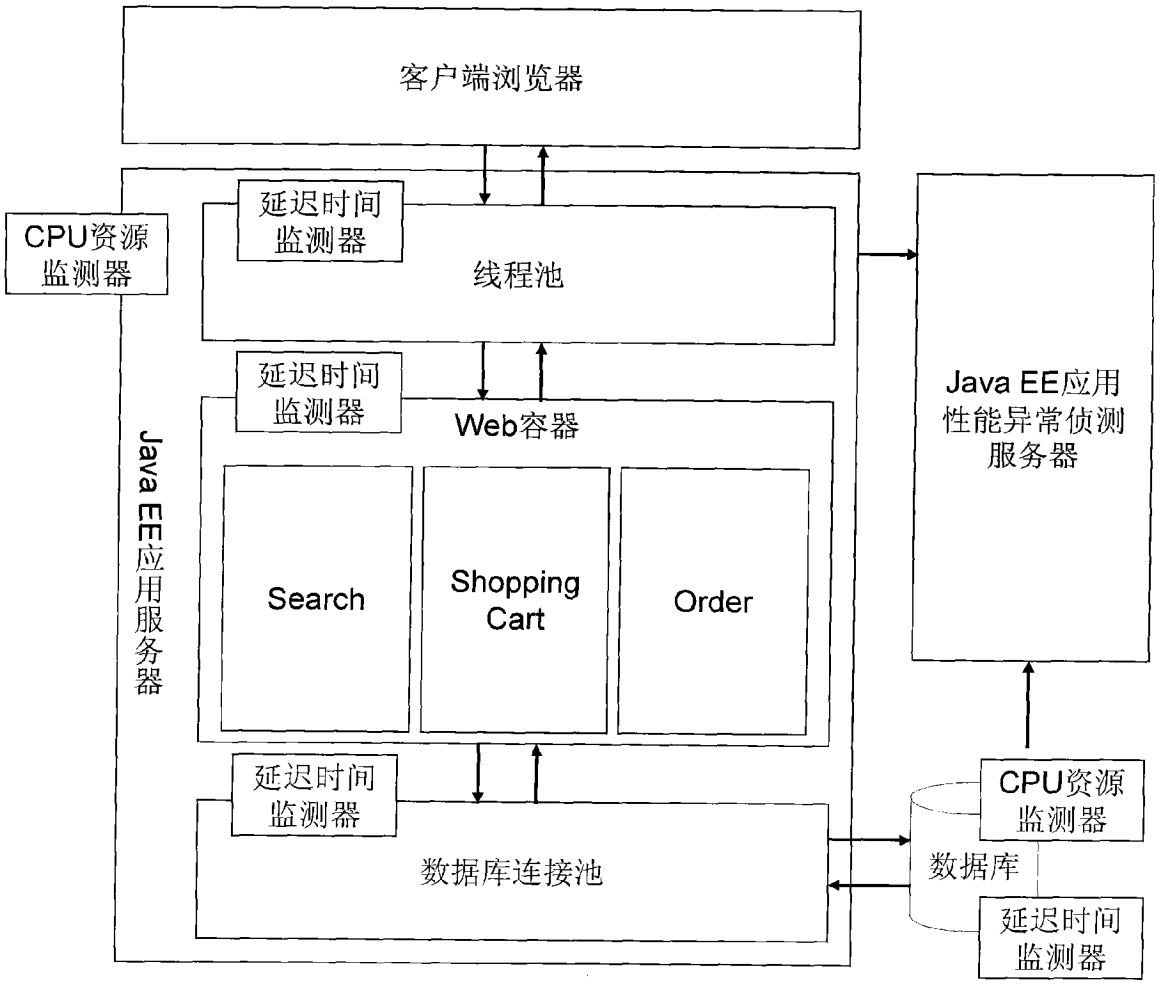

客户端浏览器
CPU资源
监测器
延迟时间
监测器
线程池
延迟时间
监测器
Web容器
Java EE应用
服务器
Search
Shopping
Cart
Order
延迟时间
监测器
数据库连接池
Java EE应用
性能异常侦测
服务器
CPU资源
监测器
数据库
延迟时间
监测器

图 4

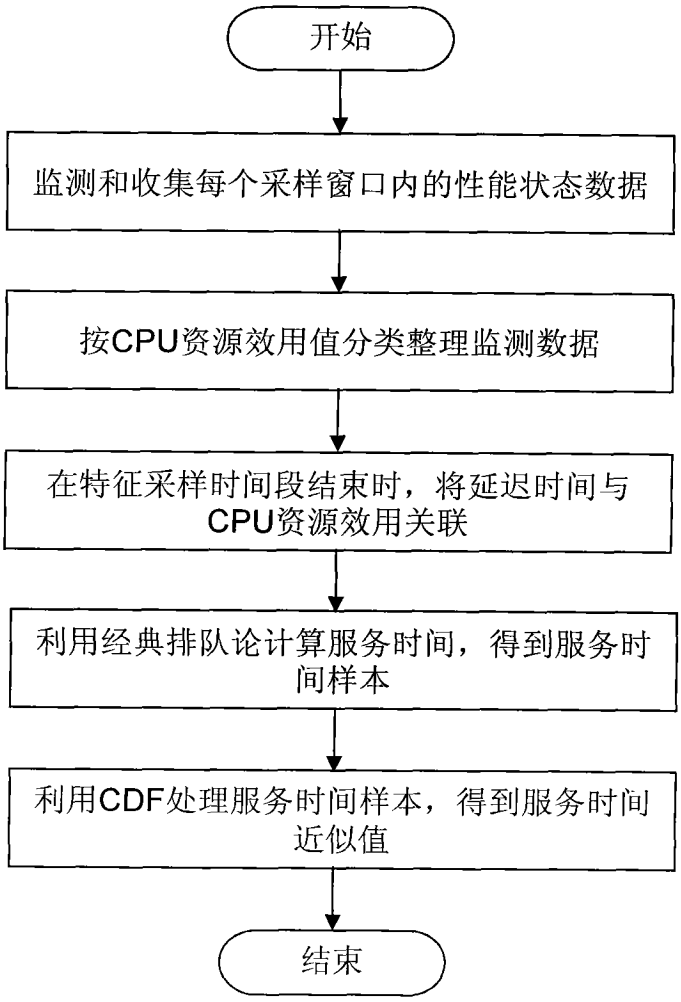

开始
监测和收集每个采样窗口内的性能状态数据
按CPU资源效用值分类整理监测数据
在特征采样时间段结束时，将延迟时间与CPU资源效用关联
利用经典排队论计算服务时间，得到服务时间样本
利用CDF处理服务时间样本，得到服务时间近似值
结束

图 5