[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
*G06F 11/36 (2006.01)*

[12] 发明专利说明书

专利号 ZL 200710001769.5

[45] 授权公告日 2009 年 10 月 28 日

[11] 授权公告号 CN 100555240C

[22] 申请日 2007.1.16
[21] 申请号 200710001769.5
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 程 龙 兰东俊 王庆波 叶 萌 陈 滢
[56] 参考文献
CN 1648870A 2005.8.3
CN 1873617A 2006.12.6
CN 1490724A 2004.4.21
审查员 艾 攀

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 杜 娟

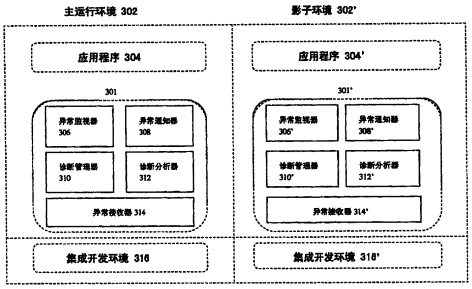
权利要求书 3 页 说明书 11 页 附图 4 页

[54] 发明名称

# 用于诊断应用程序的方法和系统

[57] 摘要

一种用于诊断应用程序的方法及其系统，其中方法包括步骤：复制应用程序的主运行环境，从而得到其影子环境；以及主运行环境与影子环境关于应用程序的故障而进行交互作用。 其中，在主运行环境中执行步骤：监视系统异常的发生；和在发现系统异常的情况下，将系统异常信息发送给所述影子环境；以及在影子环境中执行步骤：从所述主运行环境接收系统异常信息，打开用于诊断的日志/跟踪功能，从而获得与系统异常有关的诊断日志/跟踪文件；和根据获得的诊断日志/跟踪文件，分析诊断结果。 本发明还提供用于实现上述应用程序诊断方法的系统和计算机程序产品。 本发明可以使用户容易地对应用程序进行诊断而不影响应用服务器的运行性能，优化了基于集成开发环境的日志机制。



1、一种用于对应用程序进行诊断的系统，包括所述应用程序的主运行环境及其至少一个影子环境，所述影子环境是通过对所述主运行环境进行复制而得到的，该系统还至少包括所述主运行环境中的诊断模块和所述影子环境中的诊断模块，二者之间可以关于应用程序的故障而进行交互作用，并且，

所述主运行环境中的诊断模块包括：

异常监视器，用于监视系统异常的发生；和

异常通知器，用于在所述异常监视器发现系统异常的情况下，将系统异常信息发送给所述影子环境；

并且，

所述影子环境中的诊断模块包括：

异常接收器，用于与所述异常通知器相通信，以接收所述系统异常信息；

诊断管理器，用于根据所述异常接收器接收到的所述系统异常信息，打开用于诊断的日志/跟踪功能，从而获得与所述系统异常有关的诊断日志/跟踪文件；和

诊断分析器，用于根据所述诊断管理器获得的诊断日志/跟踪文件，分析诊断结果。

2、如权利要求 1 所述的系统，其中所述主运行环境中的诊断模块与所述影子环境中的诊断模块之间的交互作用包括系统异常信息的交换。

3、如权利要求 1 所述的用于对应用程序进行诊断的系统，其中所述异常监视器通过监视基本的系统日志文件来监视所述系统异常的发生。

4、如权利要求 1 所述的用于对应用程序进行诊断的系统，其中所述主运行环境中的异常通知器与所述影子环境中的异常接收器之间的通信是 Socket 通信。

5、如权利要求 1 所述的用于对应用程序进行诊断的系统，其中通过 JMX 调用或者系统开关打开所述用于诊断的日志/跟踪功能。

6、如权利要求 1 所述的用于对应用程序进行诊断的系统，其中所述系统遵循 J2EE 标准，并且所述主运行环境和所述影子环境实现在 WebSphere 应用服务器平台或 WebLogic 应用服务器平台上。

7、一种用于诊断应用程序的方法，包括以下步骤：

复制所述应用程序的主运行环境，从而得到其至少一个影子环境；以及

所述主运行环境与所述影子环境关于应用程序的故障而进行交互作用。

8、如权利要求 7 所述的方法，其中所述交互作用包括系统异常信息的交换。

9、如权利要求 7、8 中任一项所述的用于诊断应用程序的方法，所述主运行环境与所述影子环境之间的交互作用包括：

在所述主运行环境中执行下述步骤：

监视系统异常的发生；和

在发现系统异常的情况下，将系统异常信息发送给所述影子环境；

以及，

在所述影子环境中执行下述步骤：

从所述主运行环境接收所述系统异常信息，

根据接收到的所述系统异常信息，打开用于诊断的日志/跟踪功能，从而获得与所述系统异常有关的诊断日志/跟踪文件；和

根据获得的诊断日志/跟踪文件，分析诊断结果。

10、如权利要求 9 所述的用于诊断应用程序的方法，其中通过监视基本的系统日志文件来监视所述系统异常的发生。

11、如权利要求 9 所述的用于诊断应用程序的方法，其中所述主运行环境与所述影子环境之间的通信为 Socket 通信。

12、如权利要求 9 所述的用于诊断应用程序的方法，其中通过

JMX 调用或者系统开关打开所述用于诊断的日志/跟踪功能。

13、如权利要求 9 所述的用于诊断应用程序的方法，其中所述方法遵循 J2EE 标准，并且所述主运行环境和所述影子环境实现在 WebSphere 应用服务器平台或 WebLogic 应用服务器平台上。

# 用于诊断应用程序的方法和系统

## 技术领域

本发明涉及计算机程序的故障诊断。更具体地说，本发明涉及利用环境复制来诊断应用程序的方法、系统和计算机程序产品。

## 背景技术

随着互联网的迅猛发展和 WWW 应用的快速增长，Java 语言成为互联网上受到广泛欢迎的开发与编程语言。目前 Java 家族中包括三个主要的成员：J2ME(Java 2 Micro Edition，Java 2 微型版)、J2SE(Java 2 Standard Edition, Java 2 标准版)、和 J2EE（Java 2 Enterprise Edition，Jave 2 企业版）。其中，J2EE 由于具有跨平台可移植性、可获得开放源码库、具有巨大的服务器端部署库、以及涵盖大多数 W3C（万维网联盟）标准等优点，在企业应用程序开发中非常受欢迎。目前，在 J2EE 服务器上运行着数以百万计的 J2EE 应用程序，而且越来越多的 J2EE 应用程序正在开发中。

随着 J2EE 应用程序的流行，调试和问题确定（problem determination）成为一个重要的问题。目前已经发布了几个与该问题相关的标准，例如“Java 管理扩展规范（Java Management Extensions(JMX) Specification）”、“日志 API 规范（Logging API Specification （JSR47））”、和“Java 虚拟机的监视和管理规范（Monitoring and Management Specificaiton for the Java Virtual Machine（JSR174））”，等等。

对系统进行诊断时所发现的问题例如可以分为以下五个类别：

（1）功能或集成错误；

（2）性能不良；

（3）崩溃；

（4）挂起；

（5）存储器泄露。

在上面列出的五种问题中，后三种问题是难以检测的。原因在于它们通常出现在高容量的情况下，或者出现在长时间运行之后。因此，通常难以获取足够的信息以用于问题确定。可能出现问题的方面包括：JVM（Java 虚拟机）本身、本机码（native code）、Java 应用程序、系统或系统资源、子系统（例如数据库节点）、硬件，等等。

在遵循 J2EE 标准的各种应用服务器环境中提供了一些可用于问题确定的信息，例如包括：

Java 转储（JavaDump）：当 JVM 意外终止时默认地产生该 Java 转储数据，该数据概括了 JVM 在该时刻的状态；

堆转储（HeapDump）：堆转储数据根据用户的请求产生。对于堆转储的定时的更精确的控制也可以利用 Xdump:heap 选项来规定；

系统转储（SystemDump）：该系统转储数据也是由 JVM 产生的，它包含关于活动的进程、线程和系统存储器的信息，并通过 Xdump:system 选项来规定；

跟踪数据（Trace data）：该文件包含运行中的 JVM 所收集的详细数据；

快照跟踪（Snap trace）：该文件包含运行中的 JVM 所收集的少量跟踪数据，与通常的跟踪数据相似；

剖析(Profiling)：该文件是一种较高级别的日志文件，能够对应用服务器的活动提供非常详细的记录；

垃圾收集数据（Garbage collection data）：该数据由 JVM 利用 verbose:gc 选项产生，它用来分析在户应用程序的垃圾收集中的问题；

以及可以用于问题确定的其他数据，例 JIT（Just In Time）数据、类加载数据、和共享的类，等等。

尽管以上这些信息可以用于问题确定，用户仍然面临在成本与有用性之间进行选择的两难境地。一方面，应用服务器在运行中有可能意外地崩溃/转储，而无法预测其时间。找到崩溃的原因是一件很困难

的事情。尽管系统提供了一些最基本的可用于分析问题的信息，包括基本的日志文件(例如 SystemOut.log 等等)和前面提到过的转储文件，但是基本的日志文件不能提供关于应用服务器的活动的足够详细的信息，因而仅仅依赖这些日志文件不足以在应用服务器发生崩溃的情况下确定问题，而通常的转储文件仅包含应用服务器转储时的当前状态，而不包含其活动的历史记录，因此也不足以确定问题。另一方面，如果用户在运行时打开用于诊断的日志/跟踪功能以详细地记录应用服务器的活动，则系统性能将严重恶化，因为这些日志/跟踪功能的启用会占用大量的存储空间并使系统的运行速度显著下降，并且还有可能引起意想不到的问题。另外，太多的日志/跟踪数据意味着知识爆炸，同样使得用户很难确定问题。因此，在生产环境中，打开这样的用于诊断的日志/跟踪功能是不切实际的。

图 1 中示出了各种问题确定手段的有用性与其实现成本之间的关系。从图 1 中可以看出，具有较高有用性的手段意味着更高的成本，因此在生产环境中是难以实施的。

发明内容

本发明的主要目的是克服现有技术中的上述问题，提供一种能够在不影响应用服务器的运行性能的情况下，使用户容易地对应用程序进行诊断的方法、系统和计算机程序产品。

根据本发明的一个方面，提供一种用于对应用程序进行诊断的系统，包括所述应用程序的主运行环境及其至少一个影子环境，所述影子环境是通过对所述主运行环境进行复制而得到的，该系统还至少包括所述主运行环境中的诊断模块和所述影子环境中的诊断模块，二者之间可以关于应用程序的故障而进行交互作用，并且，所述主运行环境中的诊断模块包括：异常监视器，用于监视系统异常的发生；和异常通知器，用于在所述异常监视器发现系统异常的情况下，将系统异常信息发送给所述影子环境；并且，所述影子环境中的诊断模块包括：异常接收器，用于与所述异常通知器相通信，以接收所述系统异常信

息；诊断管理器，用于根据所述异常接收器接收到的所述系统异常信息，打开用于诊断的日志/跟踪功能，从而获得与所述系统异常有关的诊断日志/跟踪文件；和诊断分析器，用于根据所述诊断管理器获得的诊断日志/跟踪文件，分析诊断结果。

根据本发明的另一个方面，提供一种用于诊断应用程序的方法，包括以下步骤：复制所述应用程序的主运行环境，从而得到其至少一个影子环境；以及所述主运行环境与所述影子环境关于应用程序的故障而进行交互作用。优选地，所述主运行环境与所述影子环境之间的交互作用包括：在所述主运行环境中执行下述步骤：监视系统异常的发生；和在发现系统异常的情况下，将系统异常信息发送给所述影子环境；以及，在所述影子环境中执行下述步骤：从所述主运行环境接收所述系统异常信息，根据接收到的所述系统异常信息，打开用于诊断的日志/跟踪功能，从而获得与所述系统异常有关的诊断日志/跟踪文件；和根据获得的诊断日志/跟踪文件，分析诊断结果。

根据本发明的另一个方面，提供一种计算机程序产品，包括程序代码，当所述程序代码被计算机执行时，使得计算机执行本发明的应用程序诊断方法。

与对应用程序进行问题确定的现有方法相比，本发明的优点是，当应用程序中出现异常时，系统可以自动地在“影子环境”（或者称为“副运行环境”）中打开用于诊断的日志/跟踪功能，因此能够详细地记录应用服务器的活动而不影响主运行环境中的应用服务器的运行性能。当主运行环境中的实例由于故障而崩溃时，则可以打开影子环境中的另一个实例的详细记录来对应用程序的故障进行诊断。另外，由于在影子环境中记录的仅仅是与系统异常相关的信息，因此用户可以根据所得到的诊断日志/跟踪文件容易地检测到应用服务器中发生的动作并找到故障原因，降低了应用服务器的工作负荷并能够实现按需记录错误，从而优化了基于集成开发环境的日志机制。本发明既可以用作基于集成开发环境开发应用程序的设计时工具，也可以用作运行时工具。

附图说明

根据以下参照附图对本发明优选实施例的详细描述，本发明的上述及其它目的、特征和优点将变得更加清楚，附图中采用相同或相似的标记来表示相同或相似的部件。

图 1 中示出了各种问题确定手段的有用性与实现成本之间的关系；

图2示意性地说明了本发明的利用环境复制来对应用程序进行诊断的原理；

图3示出了根据本发明优选实施例的、能够对应用程序进行诊断的系统的示意性框图；

图4示出了根据本发明优选实施例的、用于诊断应用程序的方法的流程图。

具体实施方式

图2示意性地说明了本发明的利用环境复制来对应用程序进行诊断的原理。本发明的核心思想是：根据应用程序的主运行环境复制一个或多个影子环境，该影子环境专门用来进行应用程序的故障诊断。本具体实施方式部分基于一个影子环境进行说明，但本领域技术人员根据本发明必然明了，两个或两个以上的影子环境同样能够解决本发明的技术问题并取得相应的技术效果，且本领域技术人员基于本发明由一个影子环境构造更多的影子环境不需要任何创造性的劳动。当运行应用程序时，分别启动主运行环境及其影子环境。与此同时，也分别在这两个环境中启动了“轻量级”诊断（S201和S203）。所谓“轻量级”诊断，是指根据系统提供的一些基本信息对系统状态进行监视，以便监视系统中发生的异常事件。当在主运行环境中检测到系统异常事件时（S205）,则将其汇报给影子环境，并在影子环境中启动“重量级”诊断（S207），而主运行环境继续照常工作。所谓“重量级”诊断，是指打开用于诊断的日志/跟踪功能，以便详细地记录应用服务器的运

行信息以供诊断分析使用。然后，在影子环境中，根据在“重量级”诊断中获得的详细的诊断日志/跟踪文件进行应用程序的诊断分析，从而识别问题并采取相应的措施。需要指出的是，图 2 仅仅是为了说明本发明的原理而在时间轴上示意性地而非按比例地表示出应用程序诊断系统的工作流。

图 3 示出了根据本发明优选实施例的、对 J2EE 应用程序进行诊断的系统 300 的示意性框图。

该系统 300 包括 J2EE 应用程序 304 的主运行环境 302。该主运行环境 302 可以实现在符合 J2EE 标准的应用服务器平台、例如 IBM 公司的 WebSphere 应用服务器平台（包括 WebSphere Application Server 5.X 版、WebSphere Application Server 6.X 版,等等）上。WebSphere 应用服务器平台用来部署作为 Java 服务器方代码实现的应用程序 304,并且能够在 Microsoft 公司的 Windows NT 与 Windows 2000 操作系统上使用，也可以在大多数 UNIX 操作系统变体（例如 Sun Solaris、IBM AIX、HP-UX）以及 IBM AS/400 操作系统上使用，作为应用程序与操作系统之间的中间件。WebSphere 应用服务器平台的集成开发环境 316 则囊括了 J2EE 应用程序 304 的构建、测试与部署的整个生命周期，为 WebSphere 应用服务器提供了完整的开发、测试环境。集成开发环境 316 例如可以由 WebSphere 产品家族中的 WSAD( WebSphere Studio Application Developer )或者 RAD(Rational Application Developer)来实现。

系统 300 还包括一个影子环境 302’（或称为“副运行环境”），专门用于应用程序的诊断。该影子环境 302’是通过对主运行环境 302 进行复制而得到的。所谓的“复制”，指的是基于与主运行环境 302 同样的 Websphere 应用服务器平台或 WebBlog 应用服务器（并且注意，上述应用服务器只是示例性的，事实上任何应用服务器中间件都是适合的），按照与主运行环境 302 中的应用程序 304 同样的方式部署一个相同的应用程序 304’。主运行环境 302 和影子环境 302’中的

集成开发环境316、316’并不是必须的。在优选的实施方式中，应用程序304’的集成开发环境316’与应用程序304的集成开发环境316相同，但本发明不限于此，而是可以采用不同的集成开发环境来构建、测试与部署应用程序304和304’。

为了实现应用程序的诊断，系统300还包括分别位于主运行环境302和影子环境302’中的诊断模块301和301’。如下面将详细说明的，这两个诊断模块之间将关于应用程序的故障而进行交互作用，事实上主运行环境302和影子环境302’中的诊断模块并不限于只有一个，而是可以有多个。另外可以进行相互作用或者相互间通讯的各种诊断模块都是适合于本发明的。作为优选，诊断模块301和301’分别包括以下子模块：异常监视器306和306’、异常通知器308和308’、异常接收器314和314’、诊断管理器310和310’，以及诊断分析器312和312’。需要说明的是，由于影子环境302’是主运行环境302的副本，因此图3中示出诊断模块301’包括与诊断模块301中的子模块306-314同样的子模块306’-314’。但是，在执行应用程序诊断的过程中，主运行环境中302和影子环境302’中起作用的子模块是不同的。在图3所示的实施例中，参与应用程序诊断的子模块包括主运行环境302中的异常监视器306和异常通知器308，以及影子环境302’中的异常接收器314’、诊断管理器310’和诊断分析器312’，而其他子模块不参与应用程序的诊断。

当然，这是从描述本实施例的角度而言的。在实际应用中，由于两个运行环境302和302’是完全相同的，所以其中任一个环境都可以作为主运行环境，而另外一个环境则作为该主运行环境的“影子”环境。在图3中所示的环境302’作为主运行环境而环境302作为其影子环境的情况下，参与应用程序诊断的子模块相应地变为环境302’中的异常监视器306’和异常通知器308’，以及环境302中的异常接收器314、诊断管理器310和诊断分析器312。

诊断模块301中的异常监视器306用于对系统的状态进行监视，以便监视系统异常事件的发生。例如，当系统日志文件中有异常信息、

例如 SystemErr.log 时，异常监视器 306 分析该错误消息以便识别错误类别。异常通知器 308 用于将异常监视器 306 检测到的异常信息连同识别出的错误类别发送到影子环境 302'。影子环境 302' 的诊断模块 301' 中的异常接收器 314' 用于与主运行环境 302 中的异常通知器 308 相通信，并从异常通知器 308 接收所述异常信息。诊断管理器 310' 用于根据所述异常接收器 314' 接收到的所述系统异常信息，打开与其相关的、用于诊断的日志/跟踪功能，从而获得与该系统异常有关的诊断日志/跟踪文件。诊断分析器 312' 用于根据所述诊断管理器 310' 获得的诊断日志/跟踪文件，分析诊断结果。

图 4 详细示出了根据本发明优选实施例的、用于诊断应用程序的方法 400 的流程图。

首先，在步骤 402，对应用程序 304 的主运行环境 302 进行复制，得到一个影子环境 302'，以便用来进行应用程序的故障诊断。根据优选的实施方式，所述影子环境 302' 包括与主运行环境 302 中的集成开发环境 316 相同的集成开发环境 316'，以及部署在其上的、与主运行环境 302 中的应用程序 304 相同的应用程序 304'。

在步骤 404，启动主运行环境 302 和影子环境 302' 中的集成开发环境 316 和 316'。

在步骤 406，主运行环境 302 中的异常监视器 306 监视系统中的异常事件，例如内存不足、堆栈溢出、线程中断、线程死锁、找不到指定的文件、输入输出处理错误、数据库处理错误，等等。作为示例，异常监视器 306 可以通过监视系统日志文件来监视异常事件的发生。在运行中，WebSphere 应用服务器将系统消息写到一些基本的日志文件中。这些日志文件例如包括 JVM(标准)日志，该日志通过将 JVM 的 System.out 和 System.err 流重定向到独立的日志文件而创建。WebSphere 应用服务器将格式化的消息写到 System.out 流。另外，应用程序和其他代码可以写入这些流，这通过使用流定义的 print() 和 println() 方法实现。在默认的情况下，经重定向而创建的日志文

件被存储为<WAS_HOME>/logs/<server_name>/SystemOut.log 文件和 <WAS_HOME>/logs/<server_name>/SystemErr.log 文件。System.out日志用于监控正在运行的WebSphere应用服务器的状态是否正常，而System.err日志则包含异常堆栈信息。日志条目的基本格式例如为

<时间戳><线程ID><组件><事件类型><消息ID><消息>。

当在步骤406中异常监视器306发现系统日志文件中有异常信息、例如SystemErr.log文件时，异常监视器306在步骤410中分析该错误消息并通过例如关键字匹配的方式识别错误类别。

在步骤412，主运行环境302中的异常通知器308将异常监视器306检测到的异常信息连同所识别的错误类别发送到影子环境302'。影子环境302'中的异常接收器314'用于接收该异常信息及识别的错误类别。根据本优选实施例，主运行环境302中的异常通知器308与影子环境302'中的异常接收器314'之间采用TCP Socket方式进行通信，其中异常接收器314'持续监听主运行环境302中的特定端口是否有连接请求，异常通知器308发出连接请求后，异常接收器314'向其发回确认消息，从而在二者之间建立起连接，用于异常信息的发送。本领域技术人员可以理解，主运行环境302与影子环境302'之间也可以采用其他任何合适的方式进行通信。

接下来，在步骤422，影子环境302'中的诊断管理器310'开始应用程序诊断。

在步骤424，诊断管理器310'按照策略自动地打开一组预定的、用于诊断的日志功能（例如剖析功能、IBM活动日志功能）和/或跟踪功能，以便详细地记录与所接收的异常信息相关的运行信息。这可以通过JMX调用打开日志/跟踪功能或者通过系统开关打开日志/跟踪功能来完成。

在步骤426，影子环境302'中的诊断分析器312'根据所述诊断管理器310'获得的诊断日志/跟踪文件，分析诊断结果。该诊断分

析器 312’ 例如可以由 Websphere 应用服务器平台提供的日志分析器来实现。该日志分析器是一种图形化用户接口（GUI）诊断工具，可以读取一个或多个日志/跟踪，合并所有数据并依次显示日志/跟踪条目。优选地，该诊断工具自带一个 XML 数据库即“症状数据库（Symptom database）”，其包含一些常见问题的信息串、错误原因以及恢复步骤。日志分析器自动地将诊断日志/跟踪文件中的每个错误记录与症状数据库中的已知问题的集合相比较并显示匹配项，使得用户可以获得错误消息的解释、和有关错误原因及如何恢复错误的信息。

由此，本发明提供了一种利用环境复制的应用程序诊断方法和系统。本发明可以采取全部硬件实现、全部软件实现或者包含硬件和软件元素两者实现的形式。在优选的实施例中，本发明是以软件方式实现的，其包括但不局限于固件、常驻软件、微代码等等。

此外，本发明可以采取计算机程序产品的形式，并可以从计算机可读介质访问。该计算机可读介质提供程序代码以结合计算机或者任何指令执行系统来使用。所述计算机可读介质可以是电子的、磁性的、光学的、电磁的、红外的或者半导体器件或装置。计算机可读介质的具体例子包括半导体或者固态存储器、磁带、可移动的计算机软盘、随机存取存储器（RAM）、只读存储器（ROM）、硬磁盘和光盘。光盘的当前示例包括光盘只读存储器（CD-ROM）、读/写光盘（CD-R/W）和数字多用盘（DVD），等等。

上面已经参照特定实施例对本发明进行了详细描述。但是，对优选实施例的详细描述仅仅是示例性的，而不应被理解为限制性的。例如，在说明书中是结合IBM公司的Websphere应用服务器平台来描述本发明的。但是本领域技术人员可以理解，本发明不限于此，而是也可以应用于遵循 J2EE 标准的其他应用服务器平台、例如 BEA 公司的 WebLogic 应用服务器平台，等等。另外，本发明也不限于 J2EE 标准，而是可以应用于基于遵循任何工业标准的集成开发环境的应用程序开发、测试过程。

在不脱离本发明的精神和范围的情况下，本领域的技术人员可以进行各种修改和替换。本申请的保护范围应该由后附的权利要求书来确定。

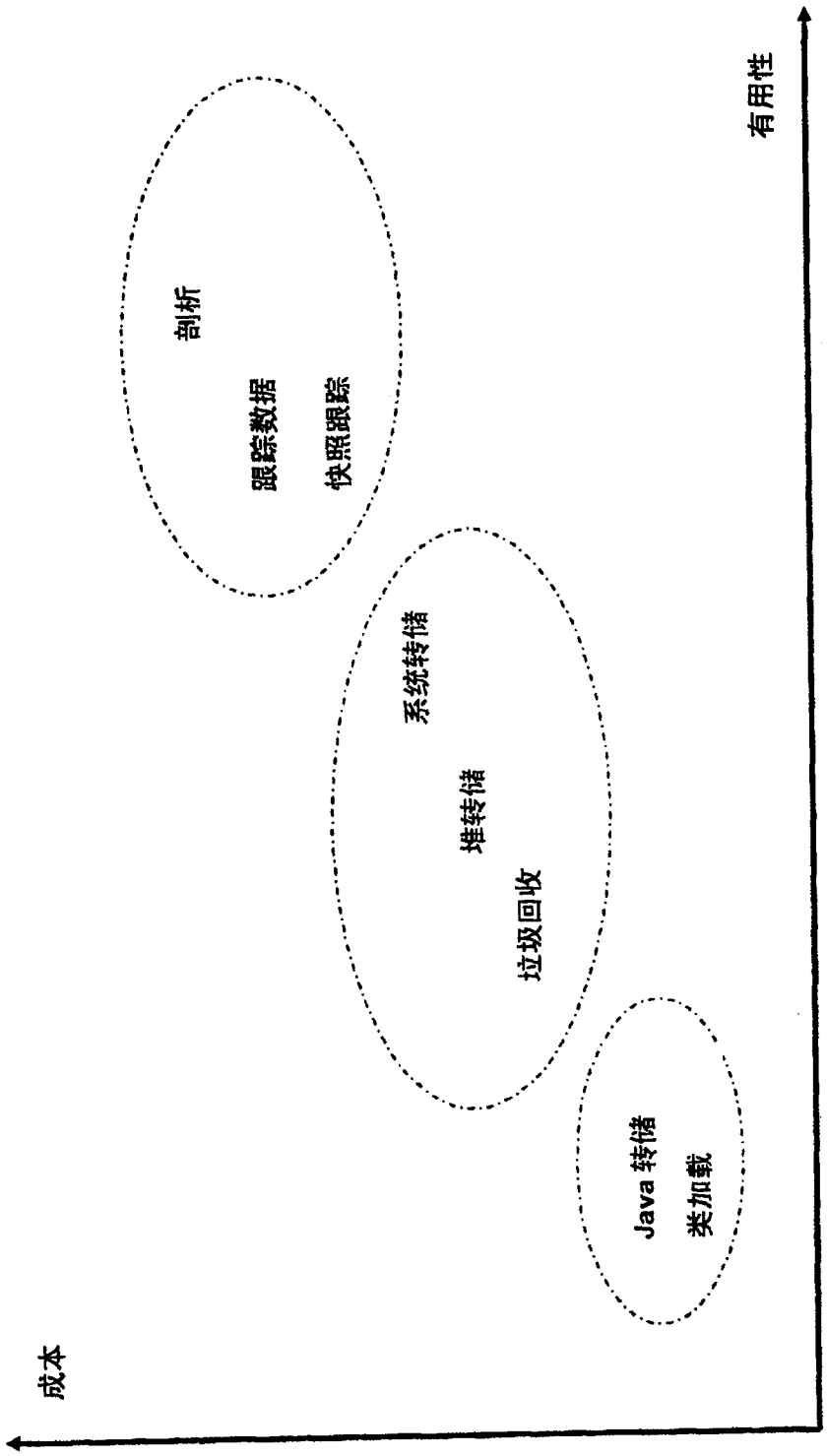



图 1

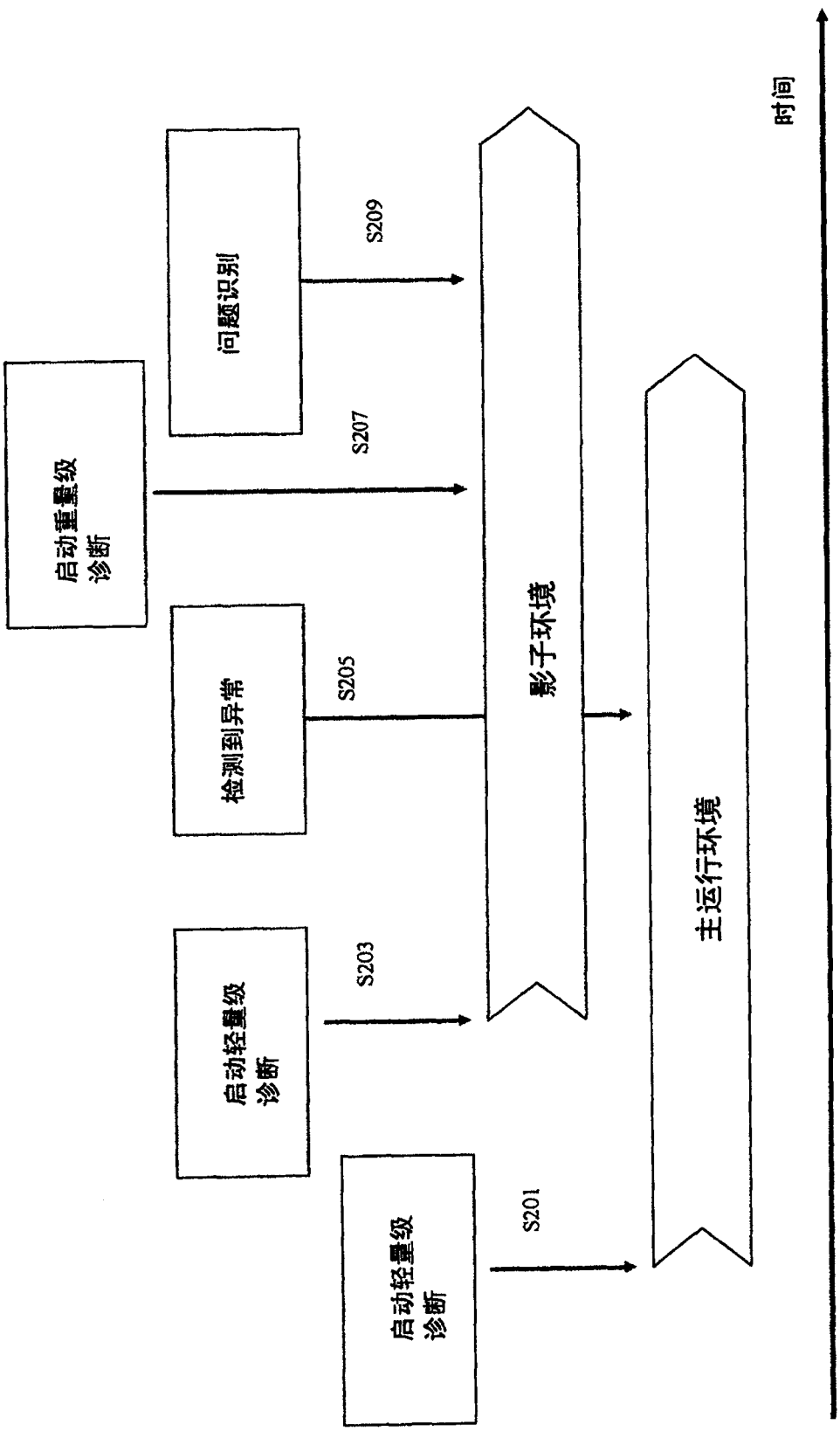

时间
主运行环境
影子环境
启动轻量级
诊断
S201
启动轻量级
诊断
S203
检测到异常
S205
启动重量级
诊断
S207
问题识别
S209

图 2

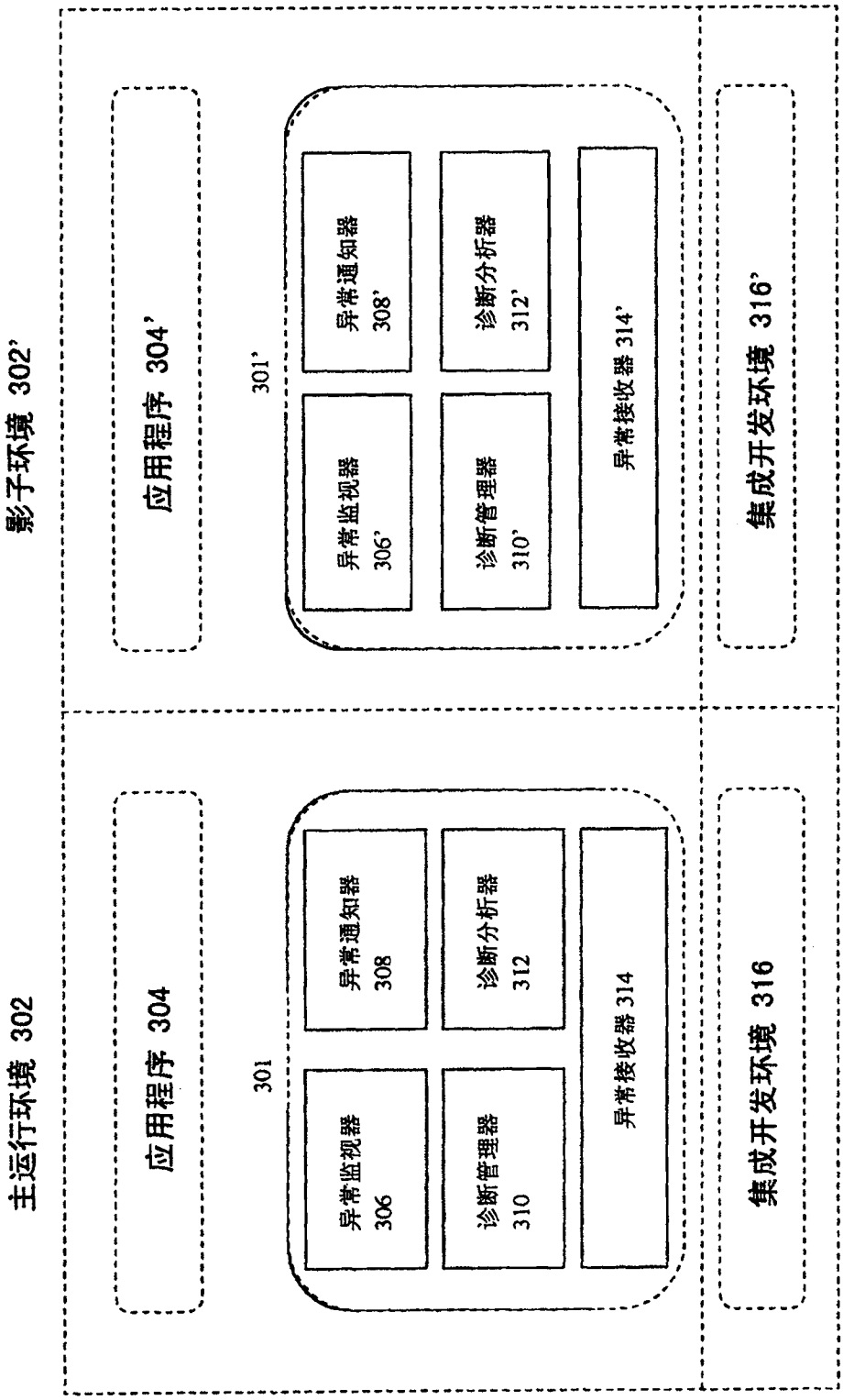

主运行环境 302
影子环境 302'
应用程序 304
301
异常监视器 306
异常通知器 308
诊断管理器 310
诊断分析器 312
异常接收器 314
集成开发环境 316
应用程序 304'
301'
异常监视器 306'
异常通知器 308'
诊断管理器 310'
诊断分析器 312'
异常接收器 314'
集成开发环境 316'

图 3

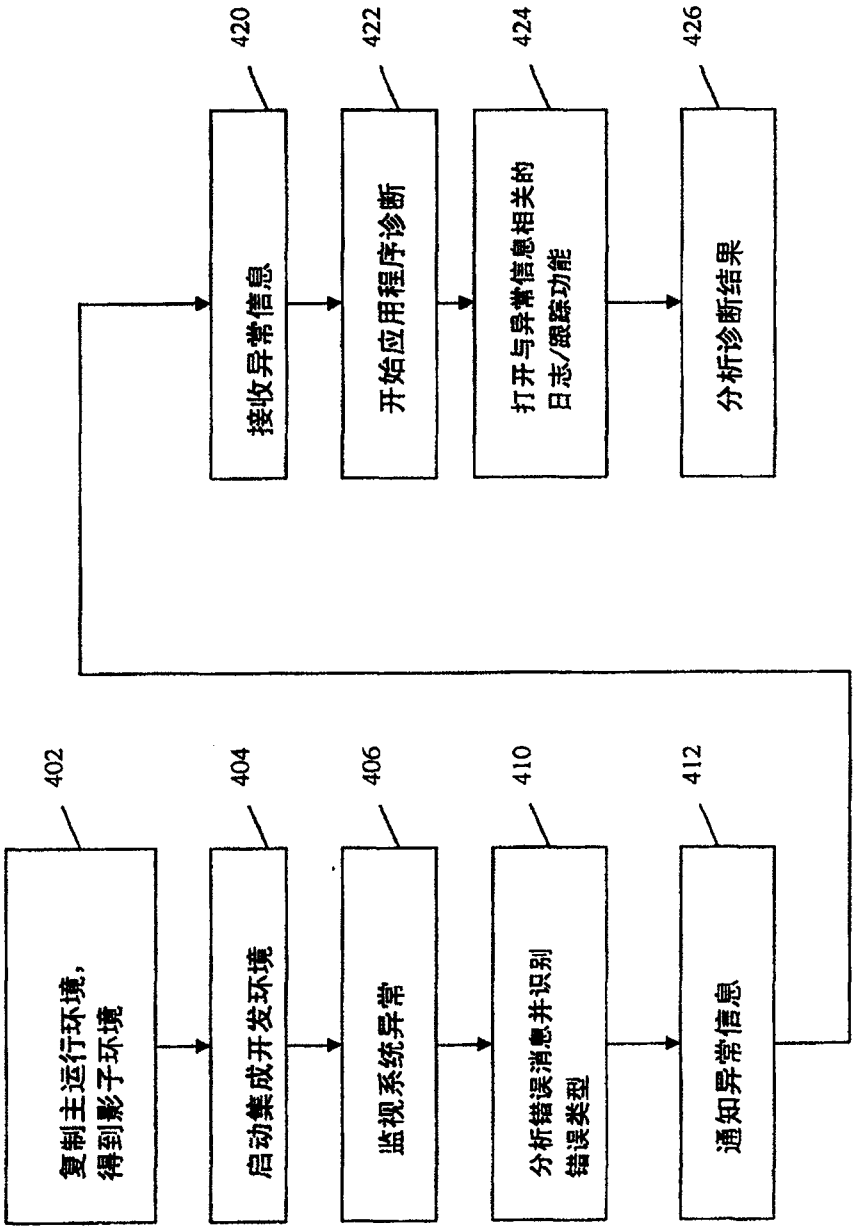

402
复制主运行环境，得到影子环境
404
启动集成开发环境
406
监视系统异常
410
分析错误消息并识别错误类型
412
通知异常信息
420
接收异常信息
422
开始应用程序诊断
424
打开与异常信息相关的日志/跟踪功能
426
分析诊断结果

图 4