(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 101393535 B
(45)授权公告日 2013.01.23

(21)申请号 200710153452.3
(22)申请日 2007.09.19
(73)专利权人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 李欣慧 B·哈格雷夫 B·特蕾西 D·伍德 刘天成 李影 邱杰 滕启明
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 吴立明
(51)Int.Cl.
G06F 11/34(2006.01)
G06F 9/46(2006.01)

(56)对比文件
WO 2006/017388 A1,2006.02.16, 全文.
CN 1549969 A,2004.11.24, 说明书第8页第13-24行、第13页第23行-16页第14行,图4、8.

审查员 赵晓敏

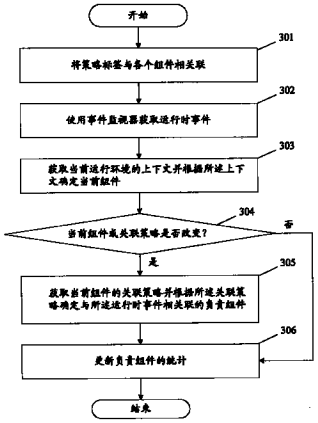
权利要求书 4 页 说明书 6 页 附图 6 页

(54)发明名称

将运行时事件与组件相关联的方法和系统

(57)摘要

本发明涉及将运行时事件与组件相关联的方法和系统。本发明的方法包括：获取运行时事件；获取当前运行环境的上下文并根据所述上下文确定当前组件；获取当前组件的关联策略并根据所述关联策略确定与所述运行时事件相关联的负责组件。本发明的一个目的是将运行时事件与组件相关联。本发明的另一个目的是在有复杂的互依存关系的组件组成的系统中确定哪个组件实际消费了资源。



CN 101393535 B

1. 一种将运行时事件与负责组件相关联的方法，包括：

获取运行时事件；

获取当前运行环境的上下文并根据所述上下文确定当前组件；

获取当前组件的关联策略并根据所述关联策略确定与所述运行时事件相关联的负责组件，

所述方法还包括以下三者中之一：

在组件中附加策略标签；

在策略存储装置中保存组件的至少一组策略标签；或者

为每个组件提供专门的 API 以将策略标签与各个组件相关联，

其中所述策略标签为表示运行时事件与组件之间的关联策略的标签。

2. 根据权利要求 1 的方法，其中获取当前运行环境的上下文并根据所述上下文确定当前组件的步骤包括获取执行堆栈的快照或者查看执行日志，并根据所述快照或者所述日志确定当前组件。

3. 根据权利要求 1 或 2 的方法，进一步包括：

如果在组件中附加了策略标签，则，

将策略标签与组件一起装载到执行堆栈；和

获取当前组件的关联策略的步骤进一步包含从所述上下文中获取所述附加的策略标签。

4. 根据权利要求 1 或 2 的方法，进一步包括：

如果在策略存储装置中保存了组件的至少一组策略标签，则，

获取当前组件的关联策略进一步包含读取策略存储装置中的所述策略标签。

5. 根据权利要求 1 或 2 的方法，如果为每个组件提供了专门的 API 以将策略标签与各个组件相关联，则，

获取当前组件的关联策略进一步包含调用专用 API 获取策略标签。

6. 根据权利要求 1 或 2 的方法，进一步包括：

保存所述运行环境的轨迹；和

递归分析当前运行环境的上下文，确定与所述运行时事件相关联的负责组件。

7. 根据权利要求 3 的方法，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则确定当前组件为负责组件；

当策略标签为 Parent 时，则确定创建当前线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则确定位于执行堆栈中最上面的组件为负责组件。

8. 根据权利要求 4 的方法，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则确定当前组件为负责组件；

当策略标签为 Parent 时，则确定创建当前线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则确定位于执行堆栈中最上面的组件为负责组件。

9. 根据权利要求 5 的方法，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则确定当前组件为负责组件；

当策略标签为 Parent 时，则确定创建当前线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则确定位于执行堆栈中最上面的组件为负责组件。

10. 根据权利要求 1 或 2 的方法，进一步包括：

获取当前运行环境的上下文后判断当前组件或当前组件的关联策略是否改变，如果是，则进一步确定与所述运行时事件相关联的负责组件。

11. 根据权利要求 1、2、7、8、9 中任一个的方法，进一步包括：

更新负责组件的统计。

12. 根据权利要求 1、2、7、8、9 中任一个的方法，其中通过主动采样或由运行时事件被动触发而获取运行时事件。

13. 一种将运行时事件与组件相关联的系统，包括：

事件监视器，用于获取运行时事件；

上下文监视器，用于获取当前运行环境的上下文并根据所述上下文确定当前组件；

推理引擎，用于获取当前组件的关联策略和根据所述关联策略确定与所述运行时事件相关联的负责组件，

所述系统还包括以下三者中之一：

注释器，用于在组件中附加策略标签；

策略存储装置，用于保存组件的策略标签；或者

专用 API，用于为每个组件提供专门的 API 以将策略标签与各个组件相关联，

其中所述策略标签为表示运行时事件与组件之间的关联策略的标签。

14. 根据权利要求 13 的系统，其中上下文监视器进一步包括堆栈巡视器或者执行日志更新器，其中堆栈巡视器用于获取执行堆栈的快照，执行日志更新器用于根据当前程序执行情况更新日志信息，所述上下文监视器根据所述快照或所述日志确定当前组件。

15. 根据权利要求 13 或 14 的系统，如果所述系统还包括注释器，则该系统进一步包括：

加载器，用于将策略标签与组件一起装载到执行堆栈；和

所述上下文监视器进一步从所述当前运行环境的上下文中获取所述附加的策略标签。

16. 根据权利要求 13 或 14 的系统，如果所述系统还包括策略存储装置，则，

所述推理引擎进一步读取策略存储装置中的所述策略标签。

17. 根据权利要求 13 或 14 的系统，其中如果所述系统还包括专用 API，则，

所述推理引擎调用专用 API 获取策略标签。

18. 根据权利要求 13 或 14 的系统，其中

所述上下文监视器进一步保存当前运行环境的轨迹；和

所述推理引擎递归分析当前运行环境的上下文，确定与所述运行时事件相关联的负责组件。

19. 根据权利要求 15 的系统，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则推理引擎确定当前组件为负责组件；

当策略标签为 Parent 时，则推理引擎确定创建线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则推理引擎确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则推理引擎确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则推理引擎确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则推理引擎确定位于执行堆栈中最上面的组件为负责组件。

20. 根据权利要求 16 的系统，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则推理引擎确定当前组件为负责组件；

当策略标签为 Parent 时，则推理引擎确定创建线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则推理引擎确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则推理引擎确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则推理引擎确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则推理引擎确定位于执行堆栈中最上面的组件为负责组件。

21. 根据权利要求 17 的系统，所述策略标签是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，其中

当策略标签为 Self 时，则推理引擎确定当前组件为负责组件；

当策略标签为 Parent 时，则推理引擎确定创建线程的父组件为负责组件；

当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

当策略标签为 Classloader 时，则推理引擎确定装载当前组件的系统组件为负责组件；

当策略标签为 Common 时，则推理引擎确定提供公共服务的组件为负责组件；

当策略标签为 Allocater 时，则推理引擎确定与资源相关联的分配器为负责组件；

当策略标签为 Boundary 时，则推理引擎确定位于执行堆栈中最上面的组件为负责组件。

22. 根据权利要求 13 或 14 的系统，其中

推理引擎进一步判断当前组件或当前组件的关联策略是否改变，如果是，则推理引擎进一步确定与所述运行时事件相关联的负责组件。

23. 根据权利要求 13、14、19、21 中任一个的系统，其中所述推理引擎进一步更新统计池中负责组件的统计。

24. 根据权利要求 13、14、19、21 中任一个的系统，其中通过主动采样或由事件被动触发而获取运行时事件。

将运行时事件与组件相关联的方法和系统

技术领域

[0001] 本发明一般涉及计算机领域，特别是将运行时事件与组件相关联的方法和系统。

背景技术

[0002] 在合作环境中，组件间存在复杂的互依存关系，并且确定资源的真实消费者非常令人迷惑。在合作过程期间，一个组件通常代表另一个组件消费资源。图 1 示出了这种情形。

[0003] 图 1 示出了现有技术中一个富客户端平台 (RCP，Rich ClientPlatform)，它是 Lotus 客户端的核心。RCP 基于 OSGI 规范，OSGI 规范具有在不同组件间频繁发生合作的特征。在此情况中，应该随着不同组件的角色和运行时上下文做出不同的记账决定。图 1 示出了 HTTP 请求的典型过程流。当“HTTP 组件”接受一个业务请求时，它将过程完全分配给“服务小程序组件 1-n”，它们将调用用于功能过程的“工具组件”。当“HTTP 组件”接受来自于管理员的请求以配置相关系统时，它将调用“配置处理器组件”用于功能过程。根据不同的相关配置的分类，它将调用“XML 处理器组件”或者“DB 处理器组件”。显然，“工具组件”代表服务小程序消费资源。但是服务小程序负责它自己的资源消费。为了分析此环境中的资源消费，原来的方法是不能胜任的。

[0004] 因此，需要在此环境中发现资源的真实消费者，这对于在合作环境中做出资源使用情况的清晰分析、做出有效的性能诊断和认识到主要消费点非常重要。为了了解软件组件的资源使用，并且这将对于完善组件设计、运行时过程或者安全做出变化需要有所帮助，已经有了一些涉及资源记账的努力。但是它们都具有缺点，这些缺点使它们不适合解决上述问题。当前的方法没有考虑合作组件间资源的真实消费者。这些方法仅关注单个组件的资源消费。在合作环境中，在没有考虑组件间关系以及没有反映组件和运行时环境的情况下，不可能做出明智的记账决定。这样，很难基于当前方法直接解决上述问题。

[0005] 首先，当前对用于资源记账的两个主要单元是线程和隔离（一种封装的 Java 程序或者不与其他的组件共享状态的应用组件）。很难将不同种类的组件映射到这两个单元。大部分组件模型独立于线程并且在不同组件间具有互依存关系。因此，缺乏对于通常组件单元的支持使当前方法难于作为基础使用。

[0006] 其次，这些方法依靠机械的代码手段。它们提供接口和相关实现的定义并将它们插入代码，这很难随不同环境做出改变。因此，在此环境下，需要发现新的方法以解决资源的真实消费者问题并且做出有效的记账。

发明内容

[0007] 本发明的一个目的是将运行时事件与组件相关联。这样，当一些重要的事件发生时，能够捕捉到精确的上下文，例如，谁在什么条件下删除了一个文件或哪个组件应当为系统崩溃负责。此外，通过将运行时事件与组件相关联，还能够识别那个组件违反了系统原则以及发现系统中易受攻击的点。

[0008] 本发明的另一个目的是在有复杂的互依存关系的组件组成的系统中确定哪个组件实际消费了资源。

[0009] 根据本发明的一个方面，提供一种将运行时事件与组件相关联的方法，包括：获取运行时事件；获取当前运行环境的上下文并根据所述上下文确定当前组件；获取当前组件的关联策略并根据所述关联策略确定与所述运行时事件相关联的负责组件。

[0010] 根据本发明的另一个方面，提供一种将运行时事件与组件相关联的系统，包括：事件监视器，用于获取运行时事件；上下文监视器，用于获取当前运行环境的上下文并根据所述上下文确定当前组件；推理引擎，用于获取当前组件的关联策略和根据所述关联策略确定与所述运行时事件相关联的负责组件。

附图说明

[0011] 通过对结合附图所示出的实施方式进行详细说明，本发明的上述以及其他特征将更加明显，本发明附图中相同的标号表示相同或相似的部件。在附图中，

[0012] 图 1 示出在合作环境中组件间存在复杂的互依存关系的示意图；

[0013] 图 2 示出按照本发明的一个实施例的为各个组件附加策略标签的示意图；

[0014] 图 3 示出按照本发明的一个实施例的将运行时事件与组件相关联的方法的流程图；

[0015] 图 4 示出按照本发明的一个实施例的将运行时事件与组件相关联的系统的方框图；

[0016] 图 5 示出按照本发明的另一个实施例的将运行时事件与组件相关联的系统的方框图；

[0017] 图 6 示出适于实施本发明的计算机系统的结构方框图。

具体实施方式

[0018] 图 2 示出按照本发明的一个实施例的为各个组件附加策略标签的示意图。图 2 中的富客户端平台中的组件与图 1 中的组件有相同的互依存关系。本发明中，为各个组件附加了一个表示运行时事件与该组件之间的关联策略的标签，即策略标签。以下将会结合实施例对此进行详细说明。

[0019] 图 3 示出按照本发明的一个实施例的将运行时事件与组件相关联的方法的流程图。

[0020] 在步骤 301，将策略标签与各个组件相关联。这个步骤可以是在设计阶段通过多种方式实现。

[0021] 在 Jave 程序中可以通过 Java 注释来实现策略标签与组件的关联。注释是 J2SE5.0(Tiger) 中的一个新功能，它为核心 Java 语言带来必要的元数据工具。注释是对代码的修改并应用于包声明、类型声明、构造器、方法、字段、参数和变量。这样可以利用在代码中插入注释来为以后的记账处理提供重要的参考。

[0022] 另一个将策略标签与各个组件相关联的方式是采用策略存储装置。例如，在独立的配置文件中存储一个或多个组件的组件名称以及它所对应的关联策略标签等等。另外，还可以为各个组件针对调试、系统安全或系统测试等多个不同的目的准备多组关联策略标

签。在运行时针对不同目的选择其中的一组。

[0023] 再一个将策略标签与各个组件相关联的方式是为每个组件提供专门的 API。当需要获取组件的策略标签时，可以调用该 API。

[0024] 接下来将详细描述在执行阶段将运行时事件与组件相关联的过程。

[0025] 在步骤 302，获取运行时事件。至少可以通过主动采样或由事件被动触发而获取事件。例如可以通过主动采样来获取一个时间间隔内消耗的 CPU 时间。对于打开文件句柄、套接字或分配存储器这样的事件可以被动地触发来获取运行时事件。

[0026] 在步骤 303，获取当前运行环境的上下文并根据所述上下文确定当前组件。其中，获取当前运行环境的上下文可以是通过堆栈巡视器获取堆栈快照并基于顶堆栈帧来确定当前组件。或者，可以通过查看执行日志来确定当前组件。执行日志是在程序执行过程中记录组件调用过程的。当执行过程进入新的组件时，该组件的信息（包括组件的策略标签）和它们的策略被记录进日志；当执行过程退出某一组件时，该信息也被记录进日志。通过这样的方式，执行日志真实的体现了当前运行时环境。这样，可以通过查看执行日志来查找当前组件。堆栈的快照和执行日志都可以反映当前运行环境，都可以用于查找当前组件及其相关策略标签。本发明只是示例性地给出了获取当前运行环境的上下文的几种方式，事实上，本领域技术人员可以采用目前获取当前运行环境的上下文的各种方式来实现本发明。

[0027] 在步骤 304，判断当前组件或关联策略是否改变。如果未改变，说明在事件发生时，运行堆栈中的当前组件和相关策略没有改变。那么可以判断为该事件负责的组件和上一事件相同，即可直接利用上次事件发生时的分析结果得出负责组件。这样，直接执行步骤 306，以更新负责组件的统计。应当理解，步骤 304 是一个可选的步骤，目的是节省其他判断负责组件的步骤以提高效率。

[0028] 如果步骤 304 的判断结果是否，则进行步骤 305，获取当前组件的关联策略并根据所述关联策略确定与所述运行时事件相关联的负责组件。

[0029] 在策略标签附加到组件的情况下，策略标签是与组件一起装载到执行堆栈中的。这样，在堆栈快照中既有组件又有组件的策略标签。当前组件所附加的策略标签可以从所述快照中获取。相似地，当程序执行进入一个组件时，取出它包含的标签并把标签和组件信息写进日志。这样，同样可以从日志中取得当前组件所附加的策略标签。

[0030] 在策略标签存储在策略存储装置中的情况下，则可以读取策略存储装置中的所述策略标签。例如，通过当前组件名检索策略存储装置中的该组件所对应的策略标签或打开当前组件所对应的配置文件以读取该组件的策略标签。

[0031] 此外，还可以调用专用的 API 来获得当前组件的策略标签。

[0032] 策略标签可以是以下标签中的一个或多个：Parent，Self，Caller，Classloader，Common，Allocater，Boundary，Leakbot 和 HeapAnalyzer。应当理解，根据不同目的可以设计其他策略标签。

[0033] 根据关联策略确定与所述运行时事件相关联的负责组件可以按如下方式实现：

[0034] 当策略标签为 Self 时，则确定当前组件为负责组件；

[0035] 当策略标签为 Parent 时，则确定创建该线程的父组件为负责组件；

[0036] 当策略标签为 Caller 时，则确定调用当前组件的组件为负责组件；

[0037] 当策略标签为 Classloader 时，则确定装载当前组件的系统组件为负责组件；

[0038] 当策略标签为 Common 时，则确定提供公共服务的组件为负责组件；

[0039] 当策略标签为 Allocater 时，则确定与资源相关联的分配器为负责组件；

[0040] 当策略标签为 Boundary 时，则确定位于执行堆栈中最上面的组件为负责组件。在采用执行日志的方式下，因为堆栈最上层的组件在执行过程中已经被记录到日志中，并且作为最后被写入的内容可以从日志中识别出来。

[0041] 根据关联策略的设计，例如需要判断父组件或当前组件的调用者时，需要保存所述执行堆栈的轨迹。根据所述执行堆栈的轨迹的递归，查找父组件或当前组件的调用者等。

[0042] 在步骤 306，更新负责组件的统计。在得到运行时事件和它的负责组件后，可以将它们记录到日志中或统计池中，用于以后的系统分析。应当理解，步骤 306 是可选的步骤。例如，在调试过程中得到运行时事件与负责组件的关联后，显示该关联关系也是可以的。

[0043] 在日志或统计池中记录运行时事件与负责组件的关联后，可以反映运行时的上下文。这样，可以记录谁在什么条件下删除了一个文件或哪个组件应当为系统崩溃负责。此外，通过将运行时事件与组件相关联，还能够识别那个组件违反了系统原则以及发现系统中易受攻击的点。

[0044] 在日志或统计池中记录运行时事件与负责组件的关联后，可以确定在有复杂的互依存关系的组件组成的系统中哪个组件实际消费了资源。使得记账处理更加准确。

[0045] 将运行时事件与组件相关联是有优势的。首先，由组件组成的系统时相互独立开发和交付的。每个组件是自治的，也就是每个组件自己管理它的资源，包括执行它的功能所使用的堆空间。每个组件负责分配它需要的堆空间并在不使用时释放它。这使得组件成为堆管理的一个单元。这样，本发明在管理由第三方组件构成的系统时非常有用。例如，给所有第三方组件附加策略标签，可以识别对系统的堆错误负责的组件。这样可以替换错误的组件或要求该组件的供应商修改该错误。

[0046] 利用本发明的方法还可以帮助查找热点。当前的热点跟踪是方法级的。由于方法的数量通常是非常大的，基于方法的热点跟踪的系统开销非常大，并且容易失去主要的关注点。本发明中的组件是任意粒度的，，可以是一个方法，也可以是一个类，或者一个包，甚至几个包的组合。这样，采用本发明可以实现基于组件的热点跟踪，而实现基于组件的热点跟踪也更为容易。

[0047] 现有的热点追踪主要依赖当前线程的运行堆栈，当创建新线程时新的堆栈被使用，导致新线程丢失了父线程的信息。这样原始执行上下文的丢失会带来决策上的混淆甚至错误。本发明中在组件中附加 Parent 标签，可以找到父线程，这样就解决了该问题，使得统计热点时更加准确。

[0048] 在现有的热点跟踪中，静态资源的初始化通常被记账到激活它的第一个实例之上。由于静态资源是为同一类的所有实例所共享的，这种记账是不公平的。在本发明中用 Classloader 标签，静态资源被记账到类加载器，使得记账更为准确。

[0049] 一些现有的热点跟踪方法利用门限值排除资源计算的对象。当某一方法的资源消耗低于该门限值时，将不再跟踪该方法的调用分支，这样造成跟踪过程中的信息丢失和分析的误差。本发明中在组件中附加 Boundary 标签，指明需要跟踪和资源记帐的范围，例如应用中线程池的使用。作为一种常见的编程模式，程序在运行过程中生成并持有若干线程为不同的请求服务。这些线程消耗的资源与本身无关，而是由于执行用户请求造成的。本

发明将通过标志它们为 Boundary 标签，而把这些线程消耗的资源自动地记在当前运行组件的帐上，从而解决了这一问题。

[0050] 利用本发明的方法还可以帮助查找内存泄漏点。在本发明中可以采用例如 Leakbot 或 HeapAnalyzer 标签找到漏点的候选者。对于候选者标注 Allocater 标签则可跟踪内存分配和释放事件。在一段时间后，导出所有对象的分配、释放事件和事件发生时的堆栈。除去对应与同一对象的内存分配和释放事件，分析剩下的事件及其相应的堆栈可以帮助找出造成内存泄漏的真正原因。

[0051] 图 4 示出按照本发明的一个实施例的将运行时事件与组件相关联的系统的方框图。本发明的将运行时事件与组件相关联的系统包括事件监视器 410，上下文监视器 420 和推理引擎 430。此外还可以包括策略存储装置 440 和统计池 450。

[0052] 事件监视器 410 用于获取运行时事件。事件监视器 410 通过主动采样或由事件被动触发而获取事件。例如可以通过主动采样来获取一个时间间隔内消耗的 CPU 时间。对于打开文件句柄、套接字或分配存储器这样的事件可以被动地通过各个资源的适配器 411、412 和 413 触发来获取运行时事件。适配器 411-413 的实现与资源的类型相关。适配器 411-413 可以是事件处理器或对资源进行定期采样的采样线程。

[0053] 当事件监视器 410 获取一个事件后，通知推理引擎 430。推理引擎 430 通知上下文监视器 420 获取当前运行环境的上下文并根据所述上下文确定当前组件。上下文监视器 420 可以是一个堆栈巡视器或者是一个执行日志读取器。堆栈巡视器获取执行堆栈的快照，执行日志读取器读取程序执行过程中被记录下来的组件调用过程。堆栈的快照与执行日志都是当前运行环境的上下文，都可以用于确定当前组件。此外，上下文监视器 420 还可以遍历所述上下文。如果需要，可以遍历上下文递归地确定当前组件的父组件或当前组件的调用者等等。当上下文监视器 420 获得当前组件后通知推理引擎 430。可选地，推理引擎 430 带有一个缓存，存储上一次事件对应的组件和策略，以及最后分析得到的负责组件。通过与缓存中保存的组件和策略比较，推理引擎进一步判断当前组件或关联策略是否改变，如果改变，推理引擎进一步确定与所述运行时事件相关联的负责组件；否则，直接引用被缓存的负责组件作为本次事件的分析结果。

[0054] 推理引擎 430 获取当前组件的关联策略和根据所述关联策略确定与所述运行时事件相关联的负责组件。

[0055] 在策略标签附加到组件的情况下，策略标签是与组件 1-n 一起装载到执行堆栈 1-n 中的。图 5 示出在 Java 环境下实现本发明的情况。其中注释器 510 可以在组件的设计阶段将标识关联策略的注释附加到组件 1-n 中。带有已注释组件被加载器 520 加载到堆栈 1-n。这样，在堆栈快照中既有组件又有组件的策略标签。当前组件所附加的策略标签可以从所述快照中获取。

[0056] 返回图 4。组件的策略标签可以存储在策略存储装置 440 中。策略存储装置 440 可以是每个组件各有一个配置文件，也可以是所有的策略标签都存储在一个配置文件。推理引擎 430 可以读取读取各个配置文件中策略标签，或者通过例如当前组件名检索策略存储装置中的该组件所对应的策略标签。

[0057] 此外，推理引擎 430 还可以调用专用的 API 来获得当前组件的策略标签。

[0058] 如果推理引擎 430 需要判断父组件或当前组件的调用者时，可以命令上下文监视

器 420 执行上下文的递归，确定与所述运行时事件相关联的负责组件。

[0059] 当推理引擎 430 获得当前事件的负责组件后，进一步在统计池 450 中更新负责组件的统计

[0060] 图 6 示意性示出了可以实现根据本发明的实施例的计算设备的结构方框图。

[0061] 图 6 中所示的计算机系统包括 CPU（中央处理单元）601、RAM（随机存取存储器）602、ROM（只读存储器）603、系统总线 604，硬盘控制器 605、键盘控制器 606、串行接口控制器 607、并行接口控制器 608、显示器控制器 609、硬盘 610、键盘 611、串行外部设备 612、并行外部设备 613 和显示器 614。在这些部件中，与系统总线 604 相连的有 CPU601、RAM602、ROM603、硬盘控制器 605、键盘控制器 606，串行接口控制器 607，并行接口控制器 608 和显示器控制器 609。硬盘 610 与硬盘控制器 605 相连，键盘 611 与键盘控制器 606 相连，串行外部设备 612 与串行接口控制器 607 相连，并行外部设备 613 与并行接口控制器 608 相连，以及显示器 614 与显示器控制器 609 相连。

[0062] 图 6 中每个部件的功能在本技术领域内都是众所周知的，并且图 6 所示的结构也是常规的。这种结构不仅用于个人计算机，而且用于手持设备，如 Palm PC、PDA（个人数据助理）、移动电话等等。在不同的应用中，例如用于实现包含有根据本发明的客户端模块的用户终端或者包含有根据本发明的网络应用服务器的服务器主机时，可以向图 6 中所示的结构添加某些部件，或者图 6 中的某些部件可以被省略。图 6 中所示的整个系统由通常作为软件存储在硬盘 610 中、或者存储在 EPROM 或者其它非易失性存储器中的计算机可读指令控制。软件也可从网络（图中未示出）下载。或者存储在硬盘 610 中，或者从网络下载的软件可被加载到 RAM602 中，并由 CPU601 执行，以便完成由软件确定的功能。

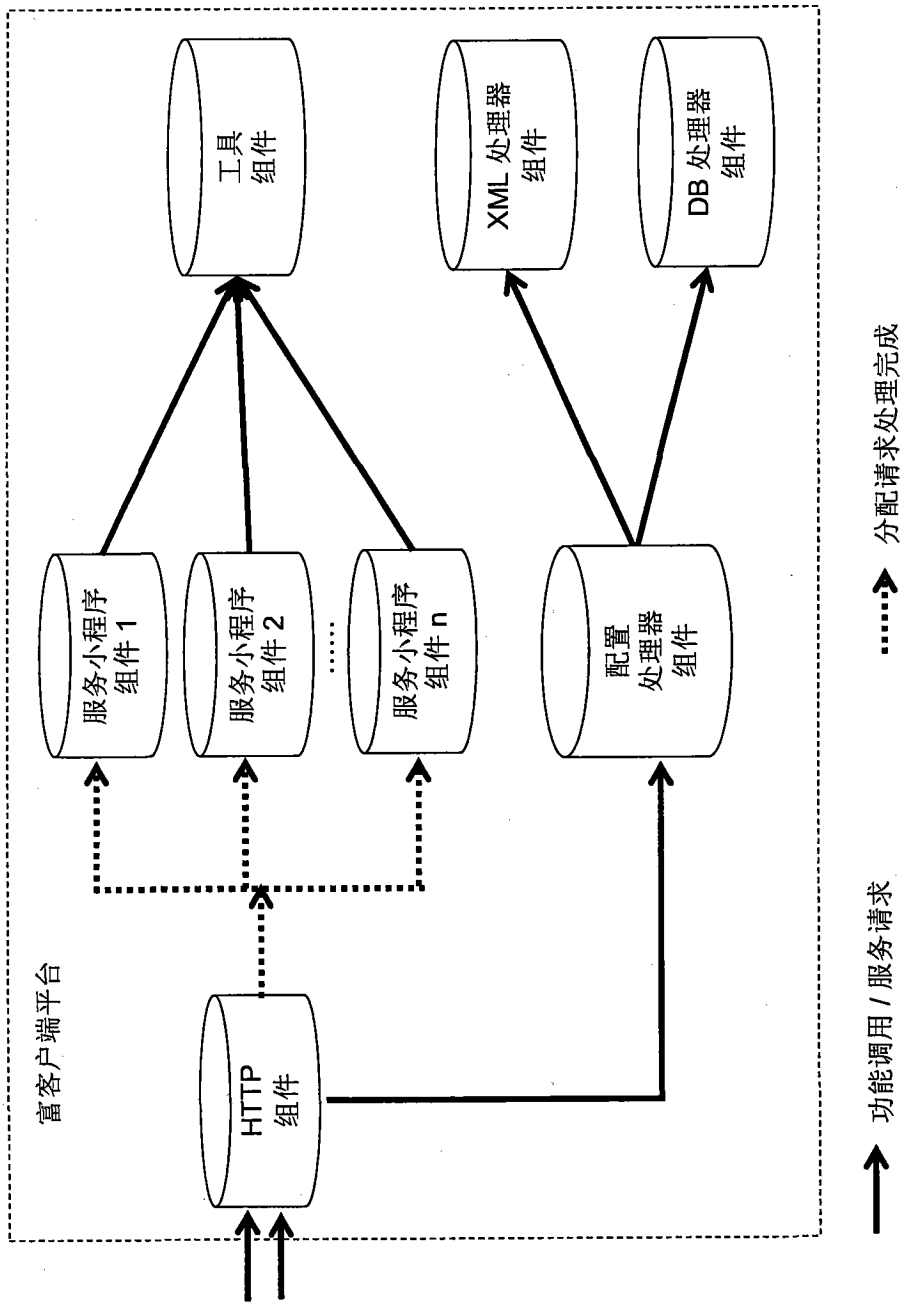
[0063] 尽管图 6 中描述的计算机系统能够支持根据本发明的提供网络内容以供脱机使用的方案，但是该计算机系统只是计算机系统的一个例子。本领域的熟练技术人员可以理解，许多其它计算机系统设计也能实现本发明的实施例。

[0064] 本发明还可以实现为例如由图 6 所示计算机系统所使用的计算机程序产品，其可以包含有用于实现根据本发明的提供网络内容以供脱机使用的网络应用服务器的代码；其还可以包含有用于实现根据本发明的用于获取网络内容以供脱机使用的客户端模块的代码。在使用之前，可以把代码存储在其它计算机系统的存储器中，例如，存储在硬盘或诸如光盘或软盘的可移动的存储器中，或者经由因特网或其它计算机网络进行下载。

[0065] 所公开的本发明的方法可以在软件、硬件、或软件和硬件的结合中实现。硬件部分可以利用专用逻辑来实现；软件部分可以存储在存储器中，由适当的指令执行系统，例如微处理器、个人计算机 (PC) 或大型机来执行。

[0066] 上述实施例以 Java 为例进行了说明。应当理解，本发明并不限于 Java 环境。本发明可以用于任何采用了组件的系统，例如 Java 或 PHP 环境等。

[0067] 虽然已经参考目前考虑到的实施例描述了本发明，但是应该理解本发明不限于所公开的实施例。相反，本发明旨在涵盖所附权利要求的精神和范围之内所包括的各种修改和等同布置。以下权利要求的范围符合最广泛解释，以便包含所有这样的修改及等同结构和功能。


工具
组件
XML 处理器
组件
DB 处理器
组件
服务小程序
组件 1
服务小程序
组件 2
.....
服务小程序
组件 n
配置
处理器
组件
富客户端平台
HTTP
组件
分配请求处理完成
功能调用 / 服务请求

现有技术

图 1

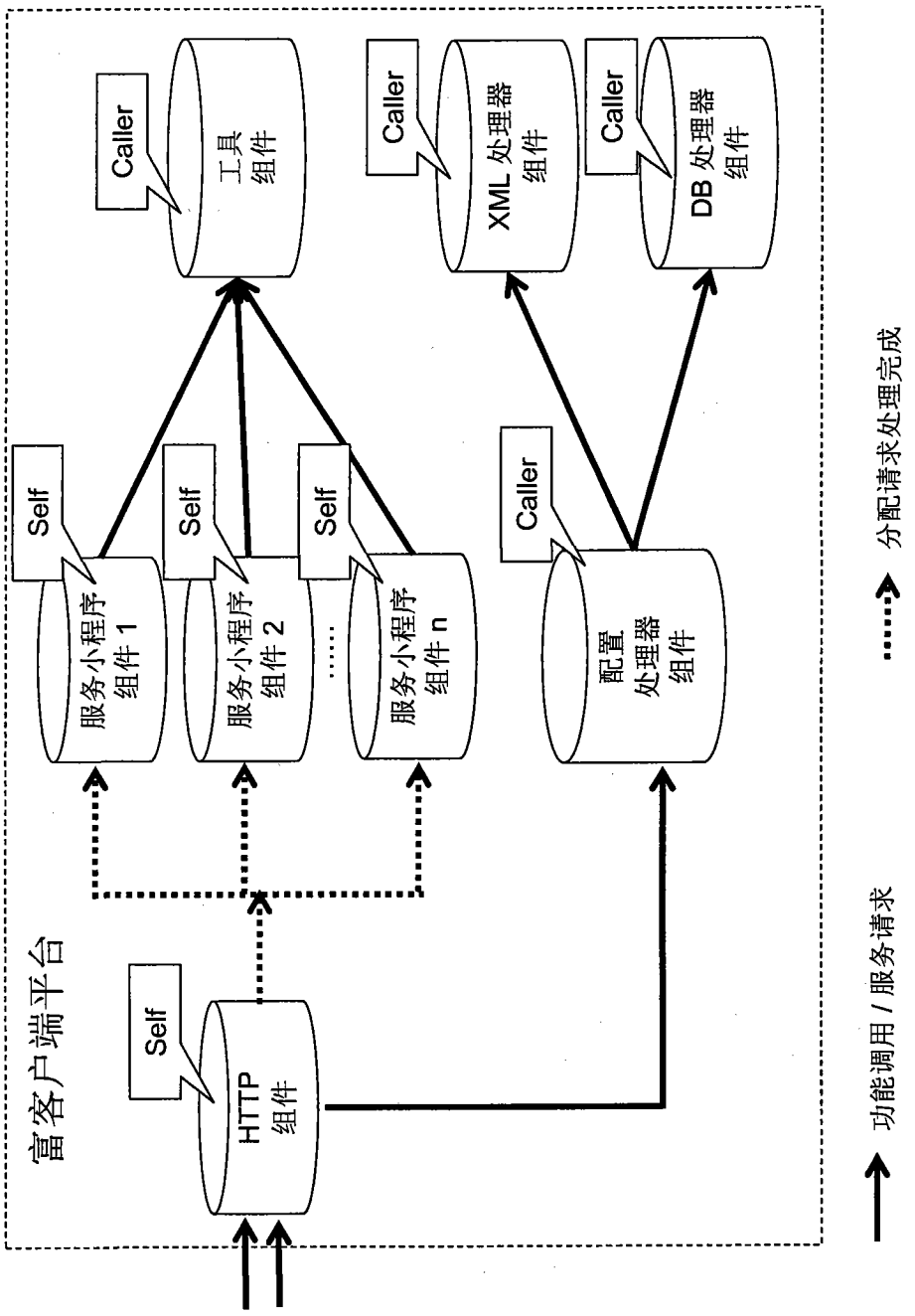

富客户端平台
Caller
工具
组件
Caller
XML 处理器
组件
Caller
DB 处理器
组件
Self
服务小程序
组件 1
Self
服务小程序
组件 2
Self
服务小程序
组件 n
Caller
配置
处理器
组件
Self
HTTP
组件
功能调用 / 服务请求
分配请求处理完成

图 2

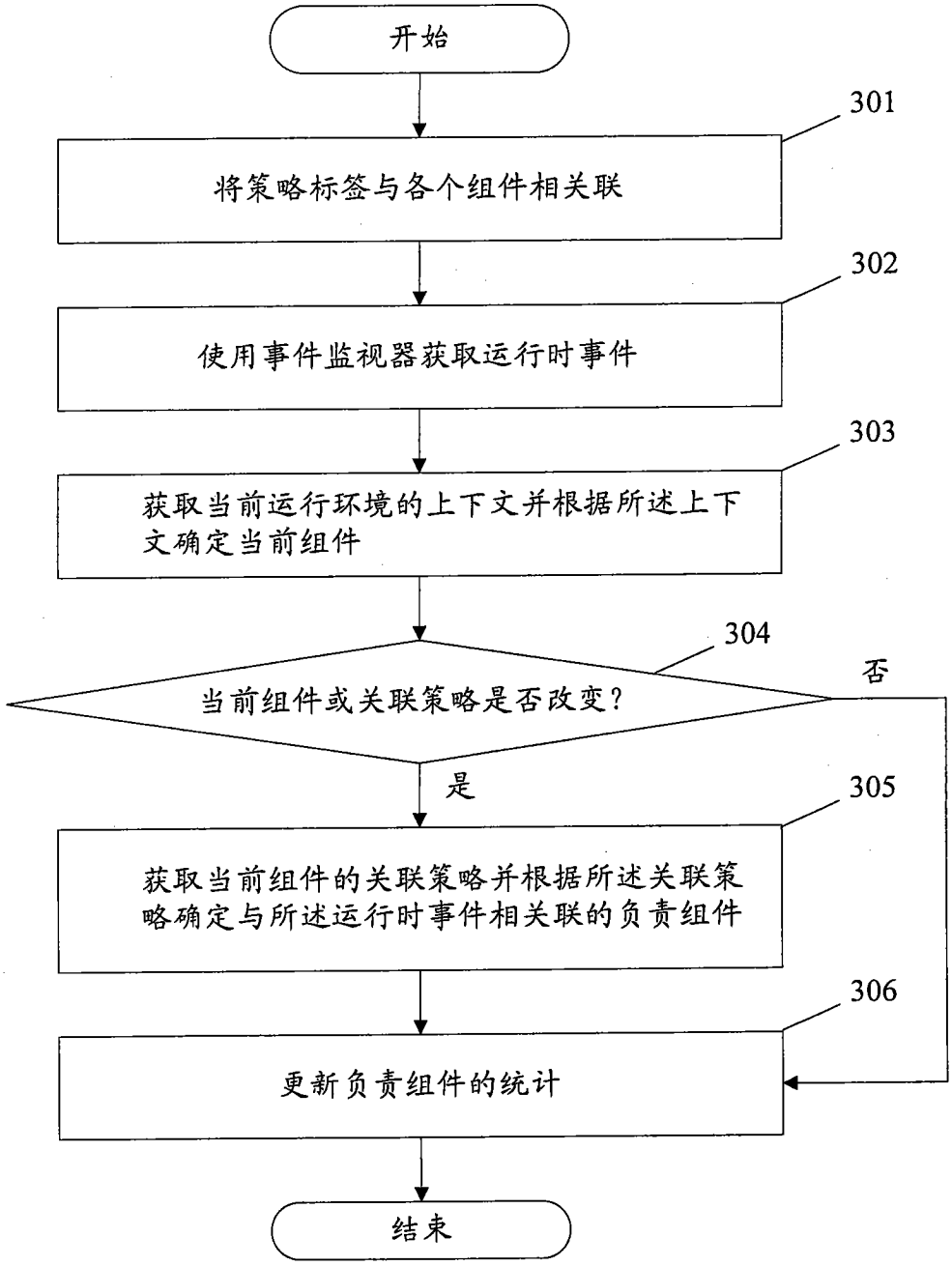

开始
301
将策略标签与各个组件相关联
302
使用事件监视器获取运行时事件
303
获取当前运行环境的上下文并根据所述上下文确定当前组件
304
当前组件或关联策略是否改变?
否
是
305
获取当前组件的关联策略并根据所述关联策略确定与所述运行时事件相关联的负责组件
306
更新负责组件的统计
结束

图 3

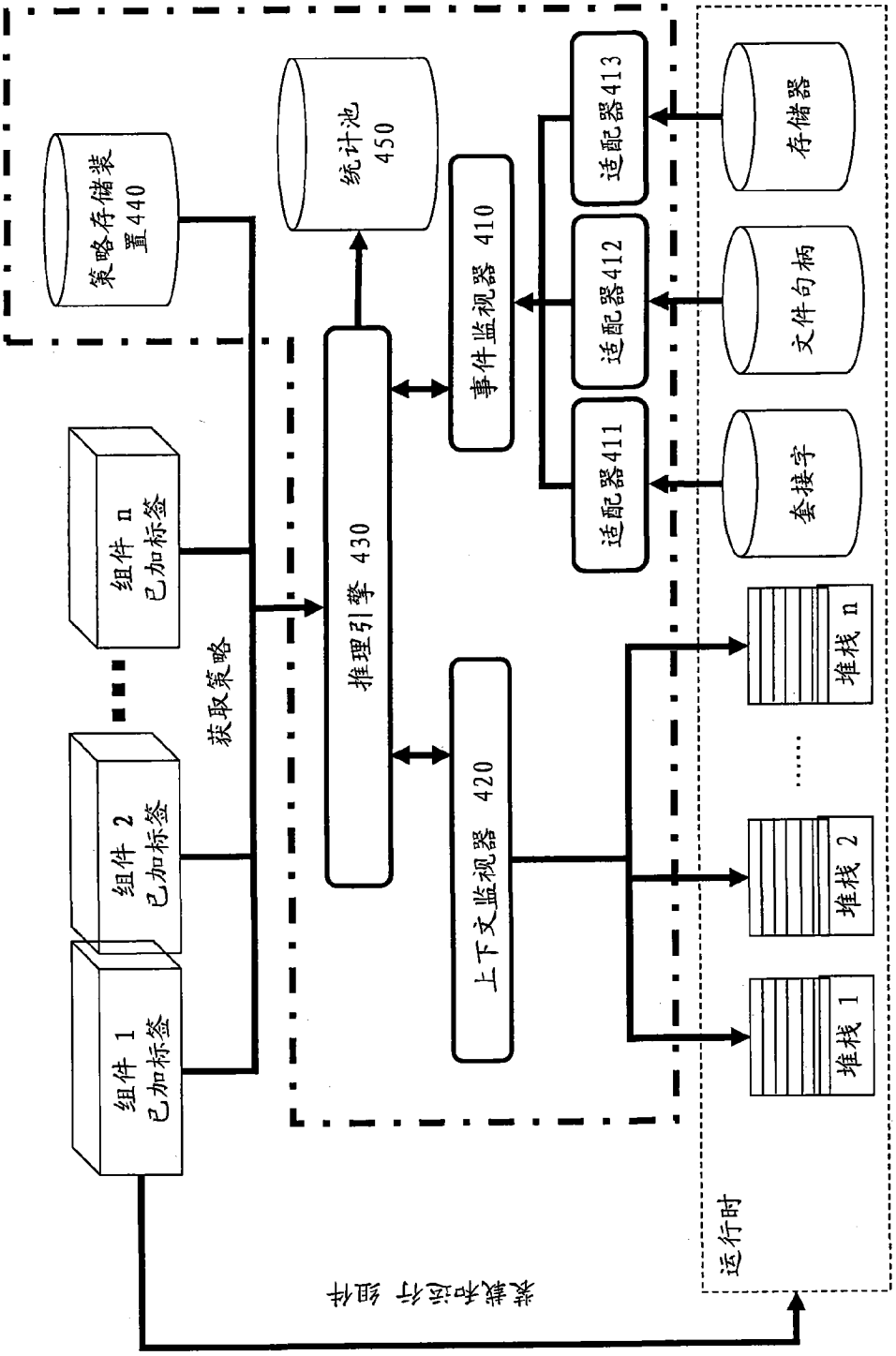

策略存储装置440
统计池450
事件监视器 410
适配器413
存储器
适配器412
文件句柄
适配器411
套接字
组件 n 已加标签
推理引擎 430
获取策略
组件 2 已加标签
上下文监视器 420
堆栈 n
堆栈 2
堆栈 1
组件 1 已加标签
运行时
装载和运行 组件

图 4

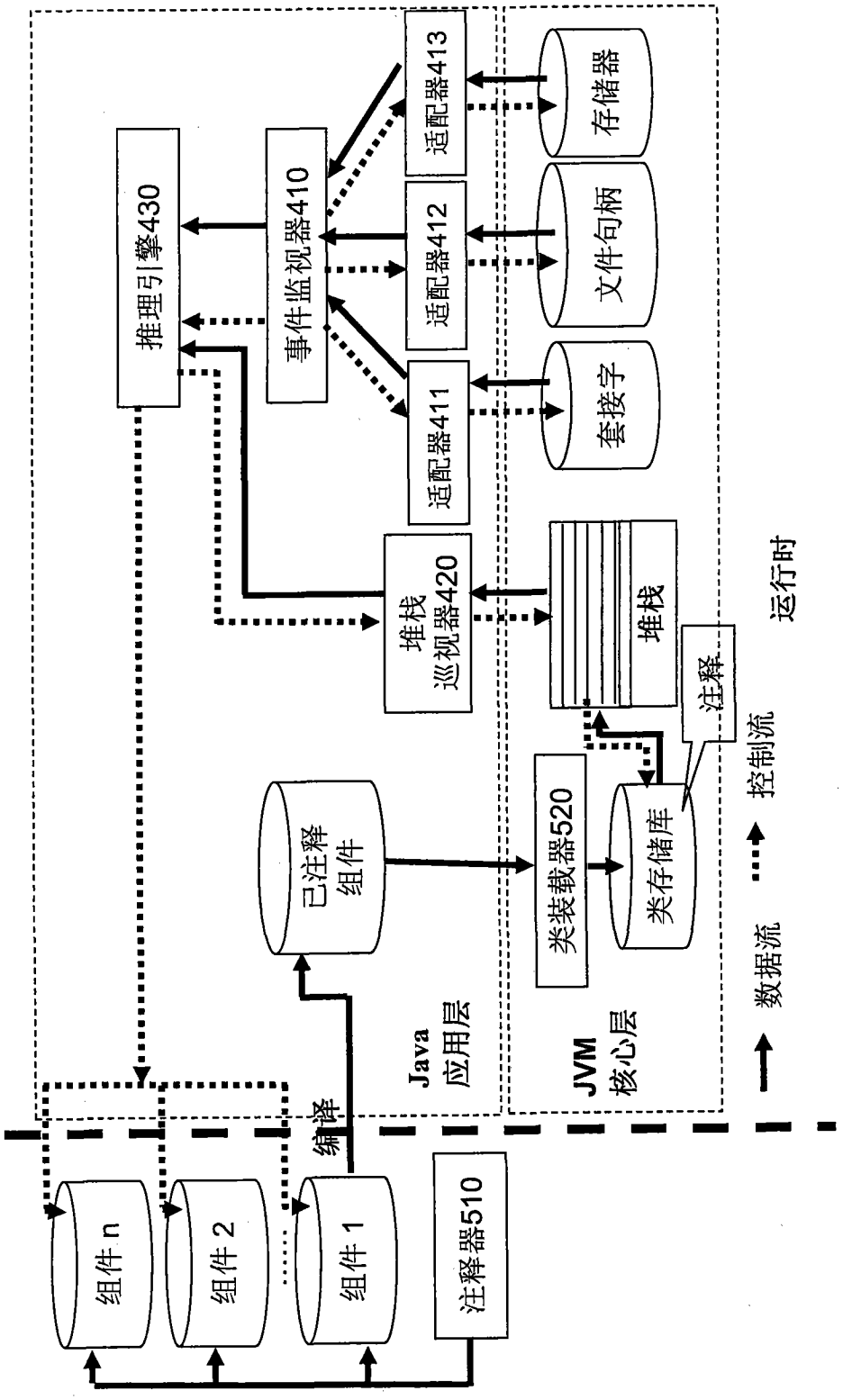

推理引擎430
事件监视器410
适配器413
适配器412
适配器411
存储器
文件句柄
套接字
堆栈
巡视器420
堆栈
注释
已注释
组件
类装载器520
类存储库
组件 n
组件 2
组件 1
注释器510
编译
Java
应用层
JVM
核心层
数据流
控制流
运行时

图 5

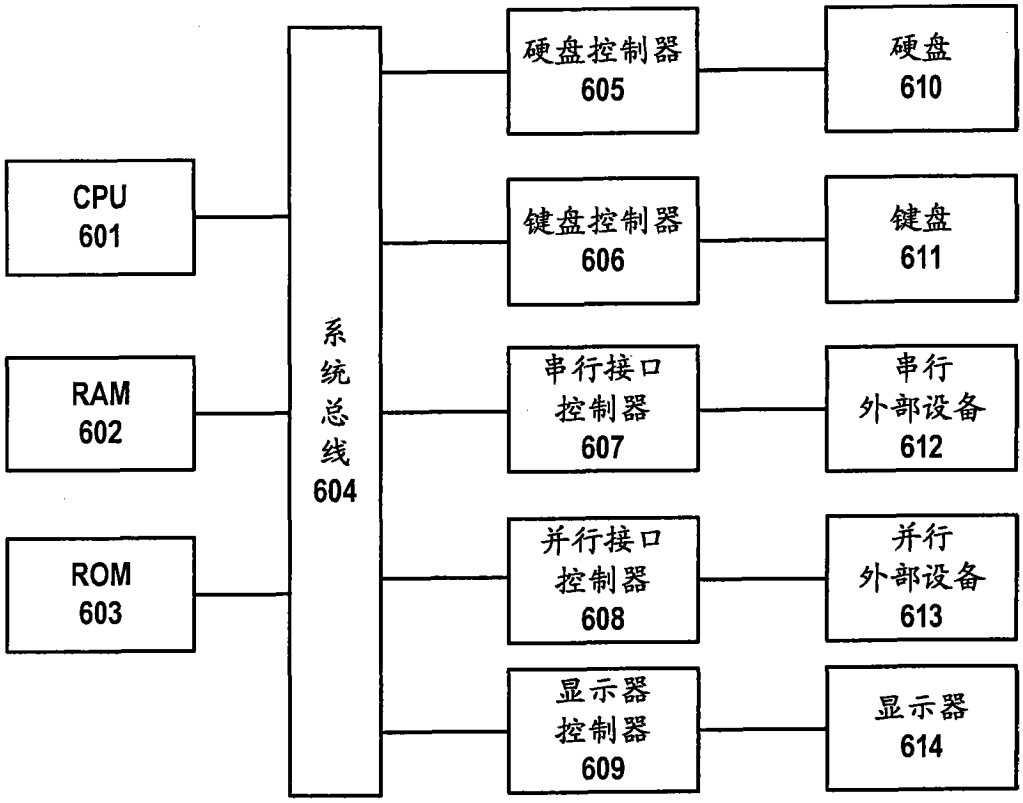

CPU
601
RAM
602
ROM
603
系统总线
604
硬盘控制器
605
键盘控制器
606
串行接口控制器
607
并行接口控制器
608
显示器控制器
609
硬盘
610
键盘
611
串行外部设备
612
并行外部设备
613
显示器
614

图 6