(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 110347572 B
(45)授权公告日 2020.08.11

(21)申请号 201910619854.0

(22)申请日 2019.07.10

(65)同一申请的已公布的文献号
申请公布号 CN 110347572 A

(43)申请公布日 2019.10.18

(73)专利权人 星环信息科技(上海)有限公司
地址 200233 上海市徐汇区虹漕路88号B栋11-12楼

(72)发明人 顾胜晖 荣国平 张贺 邵栋

(74)专利代理机构 北京品源专利代理有限公司 11332
代理人 孟金喆

(51)Int.Cl.
G06F 11/34(2006.01)

(56)对比文件
CN 101320350 A,2008.12.10
CN 109992441 A,2019.07.09
CN 101320350 A,2008.12.10
US 2018196958 A1,2018.07.12

审查员 张帅领

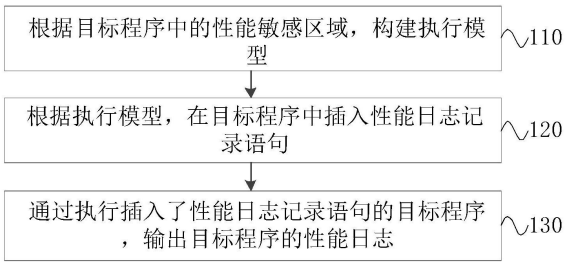
权利要求书3页 说明书10页 附图5页

(54)发明名称

一种性能日志输出方法、装置、系统、设备及介质

(57)摘要

本发明实施例公开了一种性能日志输出方法、装置、系统、设备及介质。其中,性能日志输出方法,包括:根据目标程序中的性能敏感区域,构建执行模型,其中,所述执行模型用于表示所述目标程序中的性能敏感区域的执行信息;根据所述执行模型,在所述目标程序中插入性能日志记录语句;通过执行插入了所述性能日志记录语句的目标程序,输出所述目标程序的性能日志。本发明实施例的技术方案,可缩小性能诊断范围,提高性能诊断有效性。



CN 110347572 B

1.一种性能日志输出方法,其特征在于,包括:

根据目标程序中的性能敏感区域,构建执行模型,其中,所述执行模型用于表示所述目标程序中的性能敏感区域的执行信息,所述性能敏感区域是开发者对所述目标程序的性能关注点;

根据所述执行模型,在所述目标程序中插入性能日志记录语句;

通过执行插入了所述性能日志记录语句的目标程序,输出所述目标程序的性能日志;

其中,通过执行插入了所述性能日志记录语句的目标程序,输出所述目标程序的性能日志,包括:

当开始执行所述性能日志记录语句时,计算所述性能敏感区域的日志输出阈值;

获取所述性能敏感区域的实际执行时间;

比较所述实际执行时间与所述日志输出阈值;

若所述实际执行时间超过所述日志输出阈值,输出所述性能敏感区域的性能信息到日志文件中。

2.根据权利要求1所述的方法,其特征在于,所述根据目标程序中的性能敏感区域,构建执行模型,包括:

根据开发者对所述目标程序的性能关注点确定所述目标程序中的性能敏感区域;

根据所述性能敏感区域以及所述目标程序的执行路径构建所述执行模型;

其中,所述执行信息包括:性能敏感区域的标识信息、性能敏感区域的位置、性能敏感区域的日志输出阈值以及性能敏感区域的子区域标识;所述性能敏感区域的子区域标识是指执行完当前性能敏感区域之后需要执行的下一个性能敏感区域的标识信息;当只有一个性能敏感区域时,所述性能敏感区域的子区域标识设置为空。

3.根据权利要求1所述的方法,其特征在于,所述根据所述执行模型,在所述目标程序中插入性能日志记录语句,包括:

调用性能日志记录工具,根据所述执行模型在所述目标程序中的性能敏感区域处插入所述性能日志记录语句;

其中,所述性能日志记录语句包括起始语句、终止语句及输出语句,所述输出语句用于判断是否输出性能信息到日志文件中。

4.根据权利要求1所述的方法,其特征在于,所述计算性能敏感区域的日志输出阈值,包括:

通过性能日志记录工具获取所述目标程序运行时的操作系统的状态数据,其中,所述操作系统的状态数据包括:CPU频率、CPU占用率、I/O读写速率、内存频率以及内存占用率;

根据所述性能敏感区域所涉及的敏感类型,以及所述操作系统的状态数据,计算所述性能敏感区域所涉及的敏感类型对应的系统状态值;

根据所述性能敏感区域所涉及的敏感类型对应的系统状态值,以及所述性能敏感区域的预期执行时间,确定所述性能敏感区域的日志输出阈值。

5.根据权利要求4所述的方法,其特征在于,所述敏感类型包括:CPU敏感型、I/O敏感型和内存敏感型;

根据所述性能敏感区域所涉及的敏感类型对应的系统状态值,以及所述性能敏感区域的预期执行时间,确定所述性能敏感区域的日志输出阈值,包括:

当所述性能敏感区域涉及至少两个敏感类型时，针对当前敏感类型，根据所述当前敏感类型对应的系统状态值，以及所述性能敏感区域的预期执行时间，计算所述性能敏感区域在所述当前敏感类型下的日志输出阈值；

计算得到所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值之后，对所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值进行加权求和，得到所述性能敏感区域的日志输出阈值。

6.根据权利要求1所述的方法，其特征在于，在所述通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志之后，还包括：

解析所述性能日志中的性能信息，得到解析结果；

根据所述执行模型对所述解析结果进行处理，得到所述目标程序在产生性能问题时的执行状态；

展示所述执行状态；

其中，所述执行状态包括：所述性能日志涉及的性能敏感区域的执行时间、操作的数据量、是否存在错误、执行次数以及执行路径。

7.根据权利要求6所述的方法，其特征在于，在解析所述性能日志中的性能信息，得到解析结果之后，还包括：

根据所述解析结果，确定所述目标程序中在设定时间内没有产生性能问题的性能敏感区域；

去除所述执行模型中所述没有产生性能问题的性能敏感区域。

8.一种性能日志输出装置，其特征在于，包括：

执行模型构建模块，用于根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序的性能敏感区域的执行信息，所述性能敏感区域是开发者对所述目标程序的性能关注点；

语句插入模块，用于根据所述执行模型，在所述目标程序中插入性能日志记录语句；

日志输出模块，用于通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志；

其中，所述日志输出模块，包括：

阈值计算单元，用于当开始执行所述性能日志记录语句时，计算所述性能敏感区域的日志输出阈值；

执行时间获取单元，用于获取所述性能敏感区域的实际执行时间；

时间比较单元，用于比较所述实际执行时间与所述日志输出阈值；

日志输出单元，用于若所述实际执行时间超过所述日志输出阈值，输出所述性能敏感区域的性能信息到日志文件中。

9.一种性能日志输出系统，其特征在于，包括：性能日志记录工具、性能日志分析工具、外部程序调用接口、执行模型读取器；

所述性能日志记录工具，用于根据执行模型，向目标程序注入性能日志记录语句，以记录并输出性能日志，其中，所述执行模型是根据目标程序中的性能敏感区域构建的，所述性能敏感区域是开发者对所述目标程序的性能关注点；

所述外部程序调用接口，是性能日志记录工具提供给外部程序调用的接口；

所述执行模型读取器，用于读取所述执行模型，为性能日志记录工具提供执行信息；

其中，所述日志记录工具还用于：

当开始执行所述性能日志记录语句时，计算所述性能敏感区域的日志输出阈值；

获取所述性能敏感区域的实际执行时间；

比较所述实际执行时间与所述日志输出阈值；

若所述实际执行时间超过所述日志输出阈值，输出所述性能敏感区域的性能信息到日志文件中。

10.根据权利要求9所述的系统，其特征在于，还包括：

所述性能日志分析工具，用于解析所述性能日志，并根据所述执行模型进行统计，以展示所述目标程序在产生性能问题时的执行状态；

相应的，所述执行模型读取器还用于，将读取到的所述执行模型提供给所述性能日志分析工具。

11.一种性能日志输出设备，其特征在于，所述性能日志输出设备包括：

一个或多个处理器；

存储装置，用于存储一个或多个程序；

当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现如权利要求1-7中任一所述的性能日志输出方法。

12.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，该程序被处理器执行时实现如权利要求1-7中任一所述性能日志输出方法。

一种性能日志输出方法、装置、系统、设备及介质

技术领域

[0001] 本发明实施例涉及性能诊断技术，尤其涉及一种性能日志输出方法、装置、系统、设备及介质。

背景技术

[0002] 在软件系统问题分析和诊断中，性能问题由于其模糊的特征，通常难以被定位和分析。因为软件的错误或失效通常以异常的形式存在于日志文件中，但性能问题可能不会引发任何异常(或报错)，日志文件中可能无法记录某些重要的性能信息，在这种情况下，开发人员难以确定问题是否属于性能问题。并且，即便它被证实是一个性能问题，但找出导致这种现象的原因可能仍然是耗时的，性能问题经常会引起连锁反应，很难追查其根本原因，因此性能问题诊断成了开发人员亟需解决的重要问题。

[0003] 现有性能问题诊断方法中，通常做法是记录性能日志，但大多数性能日志都是冗余的，不仅大量占用磁盘空间，还使得日志分析变得耗时且低效，同时，在程序运行过程中记录大量性能日志，还会导致较高的CPU消耗和I/O阻塞，从而影响程序执行效率。另外，开发人员根据经验和偏好在程序中插入的性能日志记录语句，由于缺乏明确的目的以及合理的判断标准，不可避免地会插入一些低效的性能日志记录语句，无法保证性能日志的质量和有效性，从而直接影响性能日志分析精度和效率。

发明内容

[0004] 本发明实施例提供一种性能日志输出方法、装置、系统、设备及介质，以缩小性能诊断范围，提高性能诊断有效性。

[0005] 第一方面，本发明实施例提供了一种性能日志输出方法，所述方法包括：

[0006] 根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序中的性能敏感区域的执行信息；

[0007] 根据所述执行模型，在所述目标程序中插入性能日志记录语句；

[0008] 通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志。

[0009] 第二方面，本发明实施例还提供了一种性能日志输出装置，所述装置包括：

[0010] 执行模型构建模块，用于根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序的性能敏感区域的执行信息；

[0011] 语句插入模块，用于根据所述执行模型，在所述目标程序中插入性能日志记录语句；

[0012] 日志输出模块，用于通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志。

[0013] 第三方面，本发明实施例还提供了一种性能日志输出系统，所述系统包括：

[0014] 性能日志记录工具、性能日志分析工具、外部程序调用接口、执行模型读取器；

[0015] 所述性能日志记录工具，用于向所述目标程序注入性能日志记录语句，以记录并输出性能日志；

[0016] 所述外部程序调用接口，是性能日志记录工具提供给外部程序调用的接口；

[0017] 所述执行模型读取器，用于读取所述执行模型，为性能日志记录工具提供执行信息。

[0018] 第四方面，本发明实施例还提供了一种设备，包括：

[0019] 一个或多个处理器；

[0020] 存储装置，用于存储一个或多个程序；

[0021] 当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现本发明任意实施例提供的性能日志输出方法。

[0022] 第五方面，本发明实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现本发明任意实施例提供的性能日志输出方法。

[0023] 本发明实施例的技术方案，根据目标程序中的性能敏感区域，构建执行模型，并根据执行模型，在目标程序中插入性能日志记录语句，最终通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志。执行模型的构建和使用，有效缩小了性能诊断范围，并且提高了性能诊断有效性。

附图说明

[0024] 图1是本发明实施例一中的一种性能日志输出方法的流程图；

[0025] 图2是本发明实施例二中的一种性能日志输出方法的流程图；

[0026] 图3是本发明实施例三中的一种性能日志输出方法的流程图；

[0027] 图4是本发明实施例三中所适用的一种配置文件的数据结构示意图；

[0028] 图5是本发明实施例三中所适用的一种性能日志记录语句的数据结构示意图；

[0029] 图6是本发明实施例三中所适用的一种目标程序执行状态的展示结果的数据结构示意图；

[0030] 图7是本发明实施例四中的一种性能日志输出装置的结构示意图；

[0031] 图8是本发明实施例五中的一种性能日志输出系统的结构示意图；

[0032] 图9是本发明实施例六提供的一种设备的结构示意图。

具体实施方式

[0033] 下面结合附图和实施例对本发明作进一步的详细说明。可以理解的是，此处所描述的具体实施例仅仅用于解释本发明，而非对本发明的限定。另外还需要说明的是，为了便于描述，附图中仅示出了与本发明相关的部分而非全部结构。

[0034] 实施例一

[0035] 图1为本发明实施例一中的一种性能日志输出方法的流程图，本实施例的技术方案适用于根据开发者的性能关注点进行性能日志输出的情况，该方法可以由性能日志输出装置执行，该装置可以由软件和/或硬件来实现，并可以集成在各种通用计算机设备中，具体包括如下步骤：

[0036] 步骤110、根据目标程序中的性能敏感区域，构建执行模型，其中，执行模型用于表

示目标程序中的性能敏感区域的执行信息。

[0037] 其中，性能敏感区域是开发者对目标程序的性能关注点，该关注点可以是目标程序最容易发生性能问题的区域，也可以是开发者不希望目标程序发生性能问题的区域。性能敏感区域的粒度大小是根据实际业务情况确定的，示例性的，性能敏感区域粒度大小可以是方法级别的。

[0038] 本实施例中，首先由开发者确定性能诊断目标程序中可能出现性能瓶颈的大概模块或方法位置，将其确定为性能敏感区域，再根据目标程序的具体执行路径对性能敏感区域进行整理和组织，最终形成执行模型。

[0039] 其中，执行模型是表示性能诊断目标程序在运行时执行路径和状态的抽象树形结构，树形结构的每个节点代表着性能诊断需求，节点连成的序列代表程序的具体执行路径。

[0040] 步骤120、根据执行模型，在目标程序中插入性能日志记录语句。

[0041] 其中，日志输出语句包括起始语句、终止语句以及输出语句。起始语句和终止语句用于记录其中一个性能敏感区域被执行时的时间戳，输出语句会根据性能敏感区域的实际执行时间以及日志输出阈值之间的关系，判断是否输出性能日志。

[0042] 本实施例中，根据步骤110中生成的执行模型中包含的性能敏感区域，使用性能日志记录工具，在性能诊断目标程序中插入性能日志记录语句。性能日志记录工具以调用库的形式存在。开发者可以调用工具提供的外部程序调用接口(Application Program Interface，API)，在性能敏感区域的起始和结束位置分别注入起始语句和终止语句，并在终止语句之后，注入输出语句。输出语句会根据性能敏感区域的实际执行时间以及日志输出阈值的关系自动决定是否需要输出性能日志。

[0043] 步骤130、通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志。

[0044] 本实施例中，在性能诊断目标程序中插入性能日志记录语句后，在目标程序运行期间，每当执行到输出语句，该输出语句会判断是否输出性能日志。示例性的，当某一性能敏感区域的执行时间大于日志输出阈值时，输出性能日志，反之，则不输出日志信息到日志中。其中，日志输出阈值是在程序开始运行时，根据操作系统状态数据、操作类型以及预期执行时间确定的。

[0045] 本实施例的技术方案，根据目标程序中的性能敏感区域，构建执行模型，并根据执行模型，在目标程序中插入性能日志记录语句，通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志，即本实施例的技术方案中，通过构建执行模型，解决了现有技术中，因记录的性能日志冗余，造成I/O、CPU或内存的额外开销以及日志分析耗时低效的问题，实现了缩小性能诊断范围，提高性能诊断有效性的效果。

[0046] 实施例二

[0047] 图2为本发明实施例二中的一种性能日志输出方法的流程图，本实施例在上述实施例的基础上进一步细化，提供了通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志的具体步骤。下面结合图2对本发明实施例二提供的一种性能日志输出方法进行说明，包括以下步骤：

[0048] 步骤210、根据目标程序中的性能敏感区域，构建执行模型，其中，执行模型用于表示目标程序中的性能敏感区域的执行信息。

[0049] 步骤220、根据执行模型，在目标程序中插入性能日志记录语句。

[0050] 步骤230、当开始执行性能日志记录语句时，计算性能敏感区域的日志输出阈值。

[0051] 本实施例中，在目标程序运行开始时，根据操作系统状态数据(例如，CPU频率、CPU占用率、I/O读写速率等)以及当前性能敏感区域预期执行时间计算不同操作类型的阈值，再根据不同操作类型的阈值按照设定算法计算该性能敏感区域的日志输出阈值。

[0052] 可选的，所述计算性能敏感区域的日志输出阈值，包括：

[0053] 通过性能日志记录工具获取目标程序运行时的操作系统的状态数据，其中，操作系统的状态数据包括：CPU频率、CPU占用率、I/O读写速率、内存频率以及内存占用率；

[0054] 根据性能敏感区域所涉及的敏感类型，以及操作系统的状态数据，计算性能敏感区域所涉及的敏感类型对应的系统状态值；

[0055] 根据性能敏感区域所涉及的敏感类型对应的系统状态值，以及性能敏感区域的预期执行时间，确定性能敏感区域的日志输出阈值。

[0056] 本可选技术方案中，性能日志记录工具会启动一个后台线程，每隔一段时间记录并更新当前操作系统的状态数据，然后计算当前性能敏感区域所包含的各操作类型(例如，CPU敏感类型操作)对应的各系统状态值，从而根据系统状态值以及预期执行时间计算出不同操作类型的日志输出阈值，最后对不同操作类型的阈值加权求和得到该性能敏感区域的日志输出阈值。示例性的，30分钟或1小时更新一次当前操作系统的指标和日志输出阈值。

[0057] 可选的，所述敏感类型包括：CPU敏感型、I/O敏感型和内存敏感型；

[0058] 根据所述性能敏感区域所涉及的敏感类型对应的系统状态值，以及所述性能敏感区域的预期执行时间，确定所述性能敏感区域的日志输出阈值，包括：

[0059] 当所述性能敏感区域涉及至少两个敏感类型时，针对当前敏感类型，根据所述当前敏感类型对应的系统状态值，以及所述性能敏感区域的预期执行时间，计算所述性能敏感区域在所述当前敏感类型下的日志输出阈值；

[0060] 计算得到所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值之后，对所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值进行加权求和，得到所述性能敏感区域的日志输出阈值。

[0061] 本可选的技术方案中，根据操作系统的状态数据和性能敏感区域的预期执行时间计算性能敏感区域的日志输出阈值。具体计算公式如下：

[0062] $$T=\sum_{t}^{T_n}\frac{t_f}{Sys(t)}*t_w$$

[0063] 其中，T代表当前性能敏感区域的日志输出阈值；

[0064] T_n代表操作类型的数量；

[0065] t代表操作类型；

[0066] t_f代表性能敏感区域预期执行时间；

[0067] Sys(t)代表该操作类型对应的系统状态值，不同的操作类型有不同的计算方式，I/O敏感类型和内存敏感类型的操作使用当前I/O读写速率或内存频率乘以对应预设因素(根据经验获取的容忍值)，CPU敏感类型操作使用CPU频率乘以CPU空闲率再乘以预设因素(根据经验获取的容忍值)得出；

[0068] t_w代表操作类型的权重。

[0069] 其中，操作类型的权重可以根据开发者的性能关注点进行灵活调整。示例的，当开发者比较关注CPU性能时，则将CPU敏感类型操作的权值适当调高。

[0070] 步骤240、获取性能敏感区域的实际执行时间。

[0071] 本实施例中，获取输入语句和输出语句记录的性能敏感区域开始执行以及执行结束的时间戳，使结束时间与开始时间做差得到性能敏感区域的实际执行时间。

[0072] 步骤250、比较实际执行时间与日志输出阈值。

[0073] 本实施例中，在当前性能敏感区域执行结束后，由输出语句获取并比较程序实际执行时间以及预先获得的日志输出阈值，以判断是输出性能信息到日志文件中。

[0074] 步骤260、若实际执行时间超过日志输出阈值，输出性能敏感区域的性能信息到日志文件中。

[0075] 本实施例中，如果当前性能敏感区域的实际执行时间超出了预先设定的日志输出阈值，表示当前区域出现了性能问题，则将输出性能信息到日志文件等待性能分析；反之，表示当前区域没有出现性能问题，不输出性能信息到日志文件中。

[0076] 本实施例的技术方案，通过在开始执行性能日志记录语句时，计算性能敏感区域的日志输出阈值，并在程序执行结束后获取性能敏感区域的实际执行时间，若实际执行时间超过所述日志输出阈值，输出性能敏感区域的性能信息到日志文件中。一方面能执行模型的使缩小了性能诊断范围，避免了记录大量冗余的性能日志信息，需要耗费大量时间来进行性能分析的情况，另一方面通过性能日志输出阈值，对性能信息进行选择性输出，即只输出出现性能问题的性能信息到日志文件，更加有针对性，进一步提高了性能诊断有效性。

[0077] 实施例三

[0078] 图3为本发明实施例三提供的一种性能日志输出方法的流程图，本实施例在上述实施例的基础上，进一步进行细化，提供了构建执行模型，在目标程序中插入性能日志记录语句，以及输出目标程序的性能日志之后的具体步骤。下面结合图3对本发明实施例三提供的一种性能日志输出方法进行说明，还包括以下步骤：

[0079] 步骤310、根据开发者对目标程序的性能关注点确定目标程序中的性能敏感区域。

[0080] 本实施例中，由开发者确定性能诊断需求，即确定性能诊断目标程序中容易出现性能问题的区域，或者最不希望出现性能问题，需要重点关注的区域，将这些区域确定为性能敏感区域。

[0081] 步骤320、根据性能敏感区域以及目标程序的执行路径构建执行模型。

[0082] 本实施例中，在确定目标程序的性能敏感区域后，根据目标程序具体执行路径，组织性能敏感区域，形成执行模型，执行模型可以表示目标程序中性能敏感区域的执行信息。

[0083] 其中，所述执行信息包括：性能敏感区域的标识信息、性能敏感区域的位置、性能敏感区域的日志输出阈值以及性能敏感区域的子区域标识；所述性能敏感区域的子区域标识是指执行完当前性能敏感区域之后需要执行的下一个性能敏感区域的标识信息；当只有一个性能敏感区域时，所述性能敏感区域的子区域标识设置为空。

[0084] 执行模型在程序中的表现形式是配置文件，如图4所示。其中，“id”表示性能敏感区域的标识信息，是一个性能敏感区域的唯一标识符；“clazz”和“method”共同组成了性能敏感区域位置，用于确定其在目标程序中的位置；“name”代码此区域的名称，用于直观的显示；“sub”中记录了当前区域子区域标识，记录了子区域的id；“threshold”中记录了此区域

的日志输出阈值。

[0085] 步骤330、调用性能日志记录工具，根据执行模型在目标程序中的性能敏感区域处插入性能日志记录语句。

[0086] 其中，所述性能日志记录语句包括起始语句、终止语句及输出语句，所述输出语句用于判断是否输出性能信息到日志文件中。

[0087] 本实施例中，通过调用性能日志记录工具，将性能日志记录语句插入到目标程序的性能敏感区域。如图5所示，具体的，将起始语句B1插入到性能敏感区域的起始位置，将终止语句B2插入至性能敏感区域结束位置，将输出语句B3插入至终止语句之后。其中，起始语句B1和终止语句B2可以记录性能敏感区域处执行开始和结束的时间戳，用于计算性能敏感区域的实际执行时间，日志输出语句B3根据监测点B0中记录的信息(程序开始执行时计算的日志输出阈值)确定是否输出当前性能敏感区域性能信息至性能日志。

[0088] 步骤340、通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志。

[0089] 步骤350、解析性能日志中的性能信息，得到解析结果。

[0090] 当接收到由输出语句输出的性能日志后，性能日志分析工具将输出的性能日志和预定义的执行模型作为输入，经过性能日志分析后，得到解析结果。

[0091] 步骤360、根据执行模型对解析结果进行处理，得到目标程序在产生性能问题时的执行状态。

[0092] 本实施例中，根据执行模型中包含的性能敏感区域以及具体执行路径，统计性能日志解析结果，以性能敏感区域为单位，分别统计出每个性能敏感区域的执行状态(例如，该区域的执行时间、操作的数据量等)。

[0093] 步骤370、展示执行状态。

[0094] 其中，所述执行状态包括：所述性能日志涉及的性能敏感区域的执行时间、操作的数据量、是否存在错误、执行次数以及执行路径。

[0095] 本实施例中，将步骤360中的统计结果进行展示，展示结果如图6所示，包括各性能敏感区域的执行时间C1，该区域执行次数C2，该区域是否存在错误C3，该区域操作数据量C4以及区域之间的执行路径C5。所展示的执行状态可以帮助开发者理解程序执行的过程和状态，进而分析性能问题出现的原因。

[0096] 步骤380、根据解析结果，确定目标程序中在设定时间内没有产生性能问题的性能敏感区域。

[0097] 本实施例中，每次执行目标程序后，将解析结果与执行模型中包含的性能敏感区域进行一一对照，并记录没有产生性能问题的性能敏感区域。

[0098] 步骤390、去除执行模型中没有产生性能问题的性能敏感区域。

[0099] 本实施例中，如果某一预设的性能敏感区域在预设时间内一直没有产生性能问题，则将改性能敏感区域从执行模型中去除，即不再输出该区域的性能日志。示例性的，预设时间可以根据目标程序的执行情况进行确定和调整。可以理解的是，步骤360和步骤380均是在步骤350之后发生的，这里的标号不对步骤360和步骤380执行的先后顺序进行限制。

[0100] 本实施例中，通过输出的性能日志进行解析，并将解析结果进行统计，最终将目标程序在产生性能问题时的执行状态进行展示，以帮助开发者理解程序执行的过程和状态，

进而分析性能问题出现的原因，同时，还根据解析结果对执行模型进行相应调整，使执行模型中包含的性能敏感区域更加具有针对性，提高性能诊断有效性。

[0101] 实施例四

[0102] 图7为本发明实施例四提供的一种性能日志输出装置的结构示意图，该性能日志输出装置，包括：执行模型构建模块410、语句插入模块420、日志输出模块430。

[0103] 执行模型构建模块410，用于根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序的性能敏感区域的执行信息；

[0104] 语句插入模块420，用于根据所述执行模型，在所述目标程序中插入性能日志记录语句；

[0105] 日志输出模块430，用于通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志。

[0106] 本实施例的技术方案，根据目标程序中的性能敏感区域，构建执行模型，并根据执行模型，在目标程序中插入性能日志记录语句，通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志，即本实施例的技术方案中，通过构建执行模型，解决了现有技术中，因记录的性能日志冗余，造成的I/O、CPU或内存的额外开销以及日志分析耗时低效的问题，实现了缩小性能诊断范围，提高性能诊断有效性的效果。

[0107] 可选的，执行模型构建模块410，包括：

[0108] 性能敏感区域确定单元，用于根据开发者对所述目标程序的性能关注点确定所述目标程序中的性能敏感区域；

[0109] 执行模型构建单元，用于根据所述性能敏感区域以及所述目标程序的执行路径构建所述执行模型；

[0110] 其中，所述执行信息包括：性能敏感区域的标识信息、性能敏感区域的位置、性能敏感区域的日志输出阈值以及性能敏感区域的子区域标识；所述性能敏感区域的子区域标识是指执行完当前性能敏感区域之后需要执行的下一个性能敏感区域的标识信息；当只有一个性能敏感区域时，所述性能敏感区域的子区域标识设置为空。

[0111] 可选的，所述语句插入模块420，具体用于：

[0112] 调用性能日志记录工具，根据所述执行模型在所述目标程序中的性能敏感区域处插入所述性能日志记录语句；

[0113] 其中，所述性能日志记录语句包括起始语句、终止语句及输出语句，所述输出语句用于判断是否输出性能信息到日志文件中。

[0114] 可选的，日志输出模块430，包括：

[0115] 阈值计算单元，用于当开始执行所述性能日志记录语句时，计算所述性能敏感区域的日志输出阈值；

[0116] 执行时间获取单元，用于获取所述性能敏感区域的实际执行时间；

[0117] 时间比较单元，用于比较所述实际执行时间与所述日志输出阈值；

[0118] 日志输出单元，用于若所述实际执行时间超过所述日志输出阈值，输出所述性能敏感区域的性能信息到日志文件中。

[0119] 可选的，所述阈值计算单元，包括：

[0120] 状态数据获取子单元，用于通过性能日志记录工具获取所述目标程序运行时的操

作系统的状态数据，其中，所述操作系统的状态数据包括：CPU频率、CPU占用率、I/O读写速率、内存频率以及内存占用率；

[0121] 系统状态值计算子单元，用于根据所述性能敏感区域所涉及的敏感类型，以及所述操作系统的状态数据，计算所述性能敏感区域所涉及的敏感类型对应的系统状态值；

[0122] 阈值确定子单元，用于根据所述性能敏感区域所涉及的敏感类型对应的系统状态值，以及所述性能敏感区域的预期执行时间，确定所述性能敏感区域的日志输出阈值。

[0123] 可选的，所述敏感类型包括：CPU敏感型、I/O敏感型和内存敏感型；

[0124] 所述阈值确定子单元，具体用于：

[0125] 当所述性能敏感区域涉及至少两个敏感类型时，针对当前敏感类型，根据所述当前敏感类型对应的系统状态值，以及所述性能敏感区域的预期执行时间，计算所述性能敏感区域在所述当前敏感类型下的日志输出阈值；

[0126] 计算得到所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值之后，对所述性能敏感区域在所述至少两个敏感类型下的日志输出阈值进行加权求和，得到所述性能敏感区域的日志输出阈值。

[0127] 可选的，日志输出模块430之后，还包括：

[0128] 信息解析模块，用于解析所述性能日志中的性能信息，得到解析结果；

[0129] 执行状态获取模块，用于根据所述执行模型对所述解析结果进行处理，得到所述目标程序在产生性能问题时的执行状态；

[0130] 执行状态展示模块，用于展示所述执行状态；

[0131] 其中，所述执行状态包括：所述性能日志涉及的性能敏感区域的执行时间、操作的数据量、是否存在错误、执行次数以及执行路径。

[0132] 可选的，所述信息解析模块之后，还包括：

[0133] 区域确定模块，用于根据所述解析结果，确定所述目标程序中在设定时间内没有产生性能问题的性能敏感区域；

[0134] 区域取出模块，用于去除所述执行模型中所述没有产生性能问题的性能敏感区域。

[0135] 本发明实施例所提供的性能日志输出装置可执行本发明任意实施例所提供的性能日志输出方法，具备执行方法相应的功能模块和有益效果。

[0136] 实施例五

[0137] 图8为本发明实施例五提供的一种性能日志输出系统的结构示意图，本实施例可适用于根据开发者的性能关注点进行性能日志输出的情况。

[0138] 如图8所示，所述性能日志输出系统，包括：性能日志记录工具510、性能日志分析工具520、外部程序调用接口530、执行模型读取器540；

[0139] 所述性能日志记录工具510，用于向所述目标程序注入性能日志记录语句，以记录并输出性能日志；

[0140] 所述外部程序调用接口530，是性能日志记录工具提供给外部程序调用的接口；

[0141] 所述执行模型读取器540，用于读取所述执行模型，为性能日志记录工具提供执行信息。

[0142] 本发明实施例中，性能日志记录工具接收执行模型读取器读取的执行模型的配置

文件，并根据上述配置文件向性能诊断目标程序注入性能日志记录语句，然后运行目标程序，输出性能信息，最终通过性能过日志分析工具对输出的新能信息进行解析，以获取目标程序在产生性能问题时的执行状态。

[0143] 可选的，所述日志记录工具510还用于：

[0144] 当开始执行所述性能日志记录语句时，计算所述性能敏感区域的日志输出阈值；

[0145] 获取所述性能敏感区域的实际执行时间；

[0146] 比较所述实际执行时间与所述日志输出阈值；

[0147] 若所述实际执行时间超过所述日志输出阈值，输出所述性能敏感区域的性能信息到日志文件中。

[0148] 本实可选的施例中，在程序执行到性能敏感区域时，性能日志记录工具运行一个后台线程，定期记录和更新操作系统状态信息，并通过系统状态信息按照设定算法计算日志输出阈值，在当前性能敏感区域执行结束后，根据起始语句和终止语句记录的时间戳，计算程序实际执行时间，当实际执行时间超过日志输出阈值时，输出当前性能敏感区域的性能信息到日志文件中。

[0149] 可选的，所述性能日志输出系统，还包括：

[0150] 所述性能日志分析工具，用于解析所述性能日志，并根据所述执行模型进行统计，以展示所述目标程序在产生性能问题时的执行状态；

[0151] 相应的，所述执行模型读取器还用于，将读取到的所述执行模型提供给所述性能日志分析工具。

[0152] 本可选的实施例中，在输出性能日志后，由性能日志分析工具解析性能日志，得到解析结果后，还要根据执行模型中包含的性能敏感区域以及其具体执行路径对解析结果进行统计，最终展示如图6所示，包括每个性能敏感区域执行时间、执行次数等执行信息以及具体执行路径的执行状态。

[0153] 本实施例根据目标程序中的性能敏感区域，构建执行模型，并根据执行模型，在目标程序中插入性能日志记录语句，最终通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志。执行模型的构建和使用，有效缩小了性能诊断范围，同时，通过对输出的性能日志进行解析，并将解析结果进行统计，最终将目标程序在产生性能问题时的执行状态进行展示，帮助开发者理解程序执行的过程和状态，还根据解析结果对执行模型进行相应调整，使执行模型中包含的性能敏感区域更加具有针对性，提高性能诊断有效性。

[0154] 实施例六

[0155] 图9为本发明实施例六提供的一种设备的结构示意图，如图9所示，该设备包括处理器60和存储器61；设备中处理器60的数量可以是一个或多个，图9中以一个处理器60为例；设备中的处理器60和存储器61可以通过总线或其他方式连接，图9中以通过总线连接为例。

[0156] 存储器61作为一种计算机可读存储介质，可用于存储软件程序、计算机可执行程序以及模块，如本发明实施例中的一种性能日志输出方法对应的程序指令/模块(例如，性能日志输出装置中的执行模型构建模块410、语句插入模块420和日志输出模块430)。处理器60通过运行存储在存储器61中的软件程序、指令以及模块，从而执行设备的各种功能应用以及数据处理，即实现上述的性能日志输出方法。

[0157] 该方法包括：

[0158] 根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序中的性能敏感区域的执行信息；

[0159] 根据所述执行模型，在所述目标程序中插入性能日志记录语句；

[0160] 通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志。

[0161] 存储器61可主要包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需的应用程序；存储数据区可存储根据终端的使用所创建的数据等。此外，存储器61可以包括高速随机存取存储器，还可以包括非易失性存储器，例如至少一个磁盘存储器件、闪存器件、或其他非易失性固态存储器件。在一些实例中，存储器61可进一步包括相对于处理器60远程设置的存储器，这些远程存储器可以通过网络连接至设备。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0162] 实施例七

[0163] 本发明实施例七还提供一种包含计算机可执行指令的存储介质，所述计算机可执行指令在由计算机处理器执行时用于执行一种性能日志输出方法，该方法包括：

[0164] 根据目标程序中的性能敏感区域，构建执行模型，其中，所述执行模型用于表示所述目标程序中的性能敏感区域的执行信息；

[0165] 根据所述执行模型，在所述目标程序中插入性能日志记录语句；

[0166] 通过执行插入了所述性能日志记录语句的目标程序，输出所述目标程序的性能日志。

[0167] 当然，本发明实施例所提供的包含计算机可执行指令的存储介质，其计算机可执行指令不限于如上所述的方法操作，还可以执行本发明任意实施例所提供的性能日志输出方法中的相关操作。

[0168] 通过以上关于实施方式的描述，所属领域的技术人员可以清楚地了解到，本发明可借助软件及必需的通用硬件来实现，当然也可以通过硬件实现，但很多情况下前者是更佳的实施方式。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在计算机可读存储介质中，如计算机的软盘、只读存储器(Read-Only Memory，ROM)、随机存取存储器(Random Access Memory，RAM)、闪存(FLASH)、硬盘或光盘等，包括若干指令用以使得一台计算机设备(可以是个人计算机，服务器，或者网络设备等)执行本发明各个实施例所述的方法。

[0169] 值得注意的是，上述一种性能日志输出装置的实施例中，所包括的各个单元和模块只是按照功能逻辑进行划分的，但并不局限于上述的划分，只要能够实现相应的功能即可；另外，各功能单元的具体名称也只是为了便于相互区分，并不用于限制本发明的保护范围。

[0170] 注意，上述仅为本发明的较佳实施例及所运用技术原理。本领域技术人员会理解，本发明不限于这里所述的特定实施例，对本领域技术人员来说能够进行各种明显的变化、重新调整和替代而不会脱离本发明的保护范围。因此，虽然通过以上实施例对本发明进行了较为详细的说明，但是本发明不仅仅限于以上实施例，在不脱离本发明构思的情况下，还可以包括更多其他等效实施例，而本发明的范围由所附的权利要求范围决定。

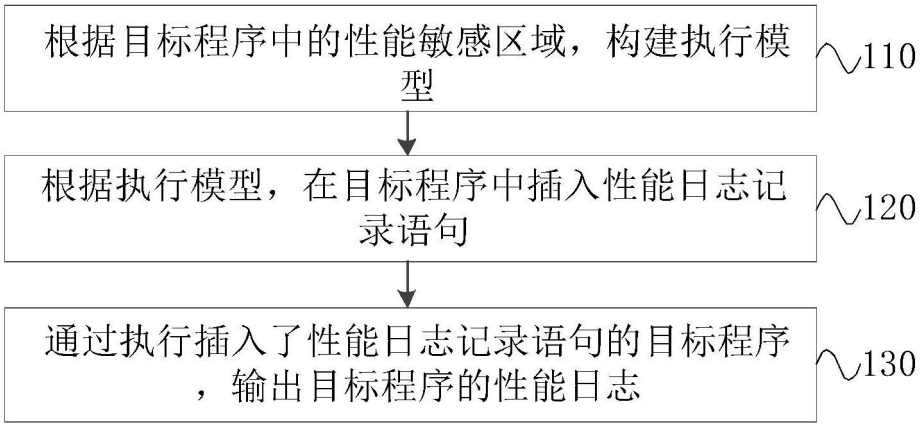

根据目标程序中的性能敏感区域，构建执行模型
110
根据执行模型，在目标程序中插入性能日志记录语句
120
通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志
130

图1


根据目标程序中的性能敏感区域，构建执行模型
210
根据执行模型，在目标程序中插入性能日志记录语句
220
当开始执行性能日志记录语句时，计算性能敏感区域的日志输出阈值
230
获取性能敏感区域的实际执行时间
240
比较实际执行时间与日志输出阈值
250
若实际执行时间超过日志输出阈值，输出性能敏感区域的性能信息到日志文件中
260

图2


根据开发者对目标程序的性能关注点确定目标程序中的性能敏感区域
310
根据性能敏感区域以及目标程序的执行路径构建执行模型
320
调用性能日志记录工具，根据执行模型在目标程序中的性能敏感区域处插入性能日志记录语句
330
通过执行插入了性能日志记录语句的目标程序，输出目标程序的性能日志
340
解析性能日志中的性能信息，得到解析结果
350
根据执行模型对解析结果进行处理，得到目标程序在产生性能问题时的执行状态
360
展示执行状态
370
根据解析结果，确定目标程序中在设定时间内没有产生性能问题的性能敏感区域
380
去除执行模型中没有产生性能问题的性能敏感区域
390

图3

```
- id: g1
  clazz: 'com.example.ClassA'
  method: 'init'
  name: 'ClassA init'
  threshold:
    - typ: IO_R
      percent: 1
      factor: 500
  sub: [g2, g4]
- id: g2
  clazz: 'com.example.ClassB'
  method: 'load'
  name: 'ClassB load'
  threshold:
    - typ: IO_W
      percent: 0.5
      factor: 5
    - typ: IO_R
      percent: 0.5
      factor: 500
  sub: [g3, g4]
- id: g3
  clazz: 'com.example.ClassC'
  method: 'calculate'
  name: 'ClassC calculate'
  threshold:
    - typ: CPU
      percent: 1
      factor: 500000
- id: g4
  clazz: 'com.example.ClassD'
  method: 'calculate'
  name: 'ClassD calculate'
  threshold:
    - typ: CPU
      percent: 1
      factor: 500000
```

图4

```
public void function_A() {
  PerfCheckpoint checkpoint = PerfLogger.checkpoint().start();
  ...
  // Other code
  ...
  checkpoint.setSize(dataSize).stop();
  PerfLogger.log(checkpoint);
}
```

B0 B1 B2 B3

图5

```
+-----------------------------------+
| 8fe658a1-dd49-4cf1-bfac-c2e2a75cc698 |
+-----------------------------------+
Total logging statements: 14

g1: 115ms 395ns, 217 tasks
+- g2: 69ms 357ns, 216 tasks
|  +- g3: 15ms 614ns, 150.8 MiB, 184 tasks, [Exception]
|  +- g4: 25ms 342ns, 23.9 MiB, 21 tasks
|  \- Unknown time: 28ms
+- g4: 29ms 23ns, 29.8 MiB, 30 tasks
\- Unknown time: 17ms
```

C1 C2 C3 C4 C5

图6

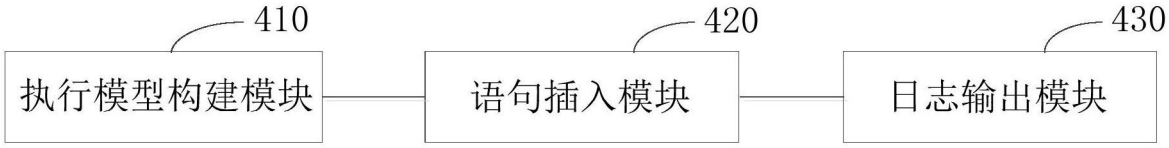


图7


性能日志输出系统
510
性能日志记录工具
530
外部程序调用接口
性能日志分析工具
520
执行模型读取器
540

图8

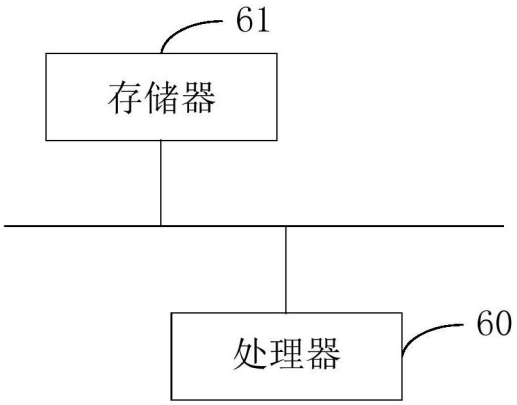

61
存储器
处理器
60

图9