(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112925552 B
(45)授权公告日 2023.07.28

(21)申请号 202110219902.4

(22)申请日 2021.02.26

(65)同一申请的已公布的文献号
申请公布号 CN 112925552 A

(43)申请公布日 2021.06.08

(73)专利权人 北京百度网讯科技有限公司
地址 100085 北京市海淀区上地十街10号百度大厦2层

(72)发明人 王翠萍

(74)专利代理机构 北京猷德知识产权代理有限公司 16084
专利代理师 范继晨

(51)Int.Cl.
G06F 8/658(2018.01)
G06F 8/656(2018.01)

(56)对比文件
CN 102279749 A,2011.12.14
CN 104636471 A,2015.05.20
CN 108595326 A,2018.09.28
CN 109002295 A,2018.12.14
CN 110704309 A,2020.01.17
CN 112181429 A,2021.01.05
US 2017193205 A1,2017.07.06
WO 2019077607 A1,2019.04.25
Hiral H. Karer 等.Dead code elimination technique in eclipse compiler for Java.2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).2015,全文.
唐佩佳;徐云;钟旭阳.基于标记语言的跨平台并行编程框架设计.计算机系统应用.2020,(第10期),全文.
许福;杨湛宇;陈志泊;孙钰;张海燕.开源代码仓库增量分析方法.清华大学学报(自然科学版).2018,(第07期),全文.

审查员 吴单单

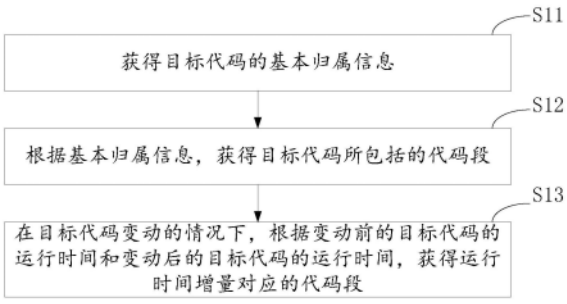
权利要求书2页 说明书11页 附图8页

(54)发明名称

代码处理方法、装置、设备及存储介质

(57)摘要

本公开提供了一种代码处理方法、装置、设备及存储介质，涉及信息流和大数据等技术领域。具体实现方案为：获得目标代码的基本归属信息；根据所述基本归属信息，获得所述目标代码所包括的代码段；在所述目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段。本公开实施例能够细粒度确定目标代码发生运行时间波动的代码段，提高代码迭代的质量。



CN 112925552 B

1.一种代码处理方法,包括:

获得目标代码的基本归属信息,其中,所述基本归属信息,包括所述目标代码的代码内容之间的调用关系、代码段的开始和/或结束信息;

根据所述基本归属信息,获得所述目标代码所包括的代码段;

在所述目标代码变动的情况下,根据变动前的目标代码的运行时间和变动后的目标代码的运行时间,获得运行时间增量对应的代码段;

所述方法还包括:确定变动前的目标代码的各代码段的运行时间;

所述在所述目标代码变动的情况下,根据变动前的目标代码的运行时间和变动后的目标代码的运行时间,获得运行时间增量对应的代码段,包括:在所述目标代码变动的情况下,对所述目标代码进行重新的静态扫描,获得所述目标代码的变动内容的基本归属关系;根据所述变动内容的基本归属关系,确定变动后的目标代码的各代码段;根据所述变动后的目标代码的各代码段,确定所述变动后的目标代码的各代码段的运行时间;根据所述变动前的目标代码的各代码段的运行时间和所述变动后的目标代码的各代码段的运行时间,获得运行时间增量对应的代码段。

2.根据权利要求1所述的方法,其中,所述获得目标代码的基本归属信息,包括:

对所述目标代码进行静态扫描,获得所述目标代码的原有内容的基本归属关系。

3.根据权利要求2所述的方法,其中,所述根据所述基本归属信息,获得所述目标代码所包括的代码段,包括:

获取所述目标代码的分段标记;

根据所述分段标记和所述目标代码的原有内容的基本归属关系,获得所述目标代码所包括的代码段。

4.一种代码处理装置,包括:

归属信息模块,用于获得目标代码的基本归属信息,其中,所述基本归属信息,包括所述目标代码的代码内容之间的调用关系;

代码段模块,用于根据所述基本归属信息,获得所述目标代码所包括的代码段;

时间增量模块,用于在所述目标代码变动的情况下,根据变动前的目标代码的运行时间和变动后的目标代码的运行时间,获得运行时间增量对应的代码段;

所述装置还包括:变动前运行时间模块,用于确定变动前的目标代码的各代码段的运行时间;

所述时间增量模块包括:变动后运行时间单元,用于在所述目标代码变动的情况下,对所述目标代码进行重新的静态扫描,获得所述目标代码的变动内容的基本归属关系;根据所述变动内容的基本归属关系,确定变动后的目标代码的各代码段;根据所述变动后的目标代码的各代码段,确定所述变动后的目标代码的各代码段的运行时间;增量对应单元,用于根据所述变动前的目标代码的各代码段的运行时间和所述变动后的目标代码的各代码段的运行时间,获得运行时间增量对应的代码段。

5.根据权利要求4所述的装置,其中,所述归属信息模块包括:

静态扫描单元,用于对所述目标代码进行静态扫描,获得所述目标代码的原有内容的基本归属关系。

6.根据权利要求5所述的装置,其中,所述代码段模块包括:

分段标记单元，用于获取所述目标代码的分段标记；

分段标记处理单元，用于根据所述分段标记和所述目标代码的原有内容的基本归属关系，获得所述目标代码所包括的代码段。

7. 一种电子设备，其特征在于，包括：

至少一个处理器；以及

与所述至少一个处理器通信连接的存储器；其中，

所述存储器存储有可被所述至少一个处理器执行的指令，所述指令被所述至少一个处理器执行，以使所述至少一个处理器能够执行权利要求1-3中任一项所述的方法。

8.一种存储有计算机指令的非瞬时计算机可读存储介质，其特征在于，所述计算机指令用于使计算机执行权利要求1-3中任一项所述的方法。

代码处理方法、装置、设备及存储介质

技术领域

[0001] 本公开涉及计算机技术领域，尤其涉及信息流和大数据等技术领域。

背景技术

[0002] 随着计算机技术的发展，计算机信息和数据量也越来越多。计算机代码的更新迭代速度也越来越快。在计算机代码进行更新时，往往是对代码进行一小段内容的更新而非全面更新。但是即使是一小段代码内容的更新，也有可能产生新的问题，导致运行时间显著增加等情况。

[0003] 由于在更新迭代代码时，更新的内容只是整个全量代码中的一小部分，因此，若更新后的代码产生运行时间显著增减的情况，则因为代码量巨大而分析难度增加。

发明内容

[0004] 本公开提供了一种代码处理方法、装置、设备及存储介质。

[0005] 根据本公开的一方面，提供了一种代码处理方法，包括：

[0006] 获得目标代码的基本归属信息；

[0007] 根据基本归属信息，获得目标代码所包括的代码段；

[0008] 在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段。

[0009] 根据本公开的另一方面，提供了一种代码处理装置，包括：

[0010] 归属信息模块，用于获得目标代码的基本归属信息；

[0011] 代码段模块，用于根据基本归属信息，获得目标代码所包括的代码段；

[0012] 时间增量模块，用于在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段。

[0013] 根据本公开的另一方面，提供了一种电子设备，包括：

[0014] 至少一个处理器；以及

[0015] 与该至少一个处理器通信连接的存储器；其中，

[0016] 该存储器存储有可被该至少一个处理器执行的指令，该指令被该至少一个处理器执行，以使该至少一个处理器能够执行本公开任一实施例中的方法。

[0017] 根据本公开的另一方面，提供了一种存储有计算机指令的非瞬时计算机可读存储介质，该计算机指令用于使计算机执行本公开任一实施例中的方法。

[0018] 根据本公开的另一方面，提供了一种计算机程序产品，包括计算机程序，该计算机程序被处理器执行时实现本公开任一实施例中的方法。

[0019] 根据本公开的技术，能够获得运行时间增量对应的代码段，将时间增量细化到代码段粒度，从而便于后续异常分析。有利于查找引起变动后的目标代码运行时间增加的原因。

[0020] 应当理解，本部分所描述的内容并非旨在标识本公开的实施例的关键或重要特

征，也不用于限制本公开的范围。本公开的其它特征将通过以下的说明书而变得容易理解。

附图说明

[0021] 附图用于更好地理解本方案，不构成对本公开的限定。其中：

[0022] 图1是根据本公开一实施例的代码处理方法流程示意图；

[0023] 图2是根据本公开另一实施例的代码处理方法流程示意图；

[0024] 图3是根据本公开又一实施例的代码处理方法流程示意图；

[0025] 图4A是根据本公开一示例的代码处理方法流程示意图；

[0026] 图4B是根据本公开一示例的代码增量示意图；

[0027] 图4C是根据本公开一示例的代码增量另一种情况示意图；

[0028] 图4D是根据本公开一示例的代码增量又一种情况示意图；

[0029] 图4E是根据本公开一示例的代码增量又一种情况示意图；

[0030] 图4F是根据本公开一示例的代码段示意图；

[0031] 图5是根据本公开一实施例的代码处理装置主要结构示意图；

[0032] 图6是根据本公开另一实施例的代码处理装置主要结构示意图；

[0033] 图7是根据本公开又一实施例的代码处理装置主要结构示意图；

[0034] 图8是根据本公开又一实施例的代码处理装置主要结构示意图；

[0035] 图9是根据本公开又一实施例的代码处理装置主要结构示意图；

[0036] 图10是根据本公开又一实施例的代码处理装置主要结构示意图；

[0037] 图11是根据本公开又一实施例的代码处理装置主要结构示意图；

[0038] 图12是用来实现本公开实施例的代码处理方法的电子设备的框图。

具体实施方式

[0039] 以下结合附图对本公开的示范性实施例做出说明，其中包括本公开实施例的各种细节以助于理解，应当将它们认为仅仅是示范性的。因此，本领域普通技术人员应当认识到，可以对这里描述的实施例做出各种改变和修改，而不会背离本公开的范围和精神。同样，为了清楚和简明，以下的描述中省略了对公知功能和结构的描述。

[0040] 本公开实施例提供一种代码处理方法，如图1所示，包括：

[0041] 步骤S11：获得目标代码的基本归属信息；

[0042] 步骤S12：根据基本归属信息，获得目标代码所包括的代码段；

[0043] 步骤S13：在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段。

[0044] 本实施例中，目标代码可以包括多个代码段。

[0045] 代码段可以按照目标代码中包含的函数进行划分，比如，包含目标函数A的代码内容可以划分为一个代码段。

[0046] 在一种可能的实现方式中，可以将代码中的一种标记作为开始标记或结束标记，比如，将最小层级的“{”符号作为开始标记，将最小层级的“}”符号作为结束标记，通过对目标代码的静态扫描，获得开始标记和结束标记，将开始标记和结束标记之间的代码内容作为代码段。

[0047] 在另一种可能的实现方式中，可以根据整个目标代码的执行阶段进行代码段的划分，将存在时间先后顺序的内容，分为一组，将每一组代码内容按照代码的执行顺序进行代码段的划分，并记录代码段的开始和结束标记。

[0048] 在一种可能的实现方式中，还可以对整个目标代码进行人工标注，写入开始和结束标记，后续根据开始和结束等标记和基本归属关系，确定代码段。

[0049] 在另一种可能的实现方式中，还可以将一行有实际信息的代码作为代码段。比如，除过“//”、“{”等占用一行的符号之外，可以将存在具体代码内容的一行代码作为一个代码段。

[0050] 在另一种可能的实现方式中，可以将目标代码中的一个策略、一个算子分别作为代码段。

[0051] 在一种可能的实现方式中，代码段可以为耗时代码段或不耗时代码段。本公开实施例可以对所有耗时代码段和不耗时代码段进行运行时间检测、判断，也可以仅针对耗时代码段进行时间检测和判断。

[0052] 本实施例中，目标代码的基本归属信息，可以包括目标代码的代码内容之间的调用关系，比如函数调用关系。

[0053] 在一种可能的实现方式中，可以根据代码中的开始和结束标记，以及基本归属信息，获得目标代码所包括的代码段。比如，函数A在目标代码段中的位置在开始标记B和结束标记C的范围之外，但是B和C之间的代码内容调用函数A，则A属于B和C之间的代码段。

[0054] 在一种可能的实现方式中，基本归属关系中，可包含代码段的开始和/或结束信息。

[0055] 本实施例中，在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段，具体可以通过对每个代码段设置对应的运行时间检测模块，检测各代码段对应的运行时间。

[0056] 在具体实现方式中，可根据需要缩小代码段的粒度，以便准确获取时间增量对应的代码段。

[0057] 本公开实施例在代码变动的情况下，获得运行时间增量对应的代码段，将时间增量细化到代码段粒度，从而便于后续异常分析。有利于查找引起变动后的目标代码运行时间增加的原因。

[0058] 在一种可能的实现方式中，获得目标代码的基本归属信息，包括：

[0059] 对目标代码进行静态扫描，获得目标代码的原有内容的基本归属关系。

[0060] 本实施例中，可采用封装、开源的静态代码扫描工具，对目标代码进行静态扫描，获得基本归属关系。

[0061] 在一种可能的实现方式中，通过静态扫描，还可获得代码的其它特征信息，如父函数等。

[0062] 在一种可能的实现方式中，对目标代码进行静态扫描，可以包括对目标代码中指定的代码特征进行识别。

[0063] 在另一种可能的实现方式中，识别结果可以输出为Json(JavaScript Object Notation,JS对象简谱)文件，便于解析。在Json文件中，可包含目标代码中命中的代码特征、代码的父函数等。

[0064] 在具体方式中，可以将代码特征设置为划分代码段的起始位置和结束位置，通过静态扫描，可以获得起始位置、结束位置、父函数等，还可以对起始位置和结束位置打标记，从而得到目标代码的代码段。

[0065] 本实施例中，目标代码的原有内容，可以是目标代码发生变动或进行迭代前的原有内容。

[0066] 本实施例中，通过静态扫描，能够获得目标代码所包含的代码段，从而后续能够对代码段进行运行时间测试，获得更细粒度的运行时间与代码的对应关系。

[0067] 在一种实施方式中，根据基本归属信息，获得目标代码所包括的代码段，包括：

[0068] 获取目标代码的分段标记；

[0069] 根据分段标记和目标代码的原有内容的基本归属关系，获得目标代码所包括的代码段。

[0070] 在一种可能的实现方式中，分段标记可以是根据设定的特征、由静态代码扫描工具扫描目标代码获得的。

[0071] 根据分段标记和目标代码的原有内容的基本归属关系，获得目标代码所包括的代码段，具体可以包括：将开始的分段标记和结束的分段标记之间的代码内容、以及开始标记和结束标记之间的代码内容所调用的代码内容，作为一个代码段。

[0072] 本公开实施例中，能够根据代码段之间的归属关系和目标代码的分段标记，获得目标代码所包含的代码段，从而便于后续对各代码段的运行时间进行检测。

[0073] 在一种实施方式中，如图2所示，代码处理方法还包括：

[0074] 步骤S21：确定变动前的目标代码的各代码段的运行时间。

[0075] 变动前的目标代码，为目标代码的全部原始内容。

[0076] 本实施例中，能够获得变动前的目标代码的各代码段的运行时间，从而能够在目标代码发生变动后，与变动的代码段的运行时间进行对比，确定增加的运行时间是否由目标代码变动造成。

[0077] 在一种实施方式中，在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段，包括：

[0078] 在目标代码变动的情况下，确定变动后的目标代码的各代码段的运行时间；

[0079] 根据变动前的目标代码的各代码段的运行时间和变动后的目标代码的各代码段的运行时间，获得运行时间增量对应的代码段。

[0080] 本实施例中，可以在每次目标代码发生更新迭代时，均进行一次时间测试，以确保变动的代码段在运行时间方面不存在异常。

[0081] 本实施例中，代码变动的情况下，确定运行时间增量对应的代码段，从而能够根据运行时间增量对应的代码段与发生变动的代码段之间的关系，确定运行时间增加是否正常。

[0082] 在一种实施方式中，在目标代码变动的情况下，确定变动后的目标代码的各代码段的运行时间，包括：

[0083] 在目标代码变动的情况下，对目标代码进行重新的静态扫描，获得目标代码的变动内容的基本归属关系；

[0084] 根据变动内容的基本归属关系，确定变动后的目标代码的各代码段；

[0085] 根据变动后的目标代码的各代码段，确定变动后的目标代码的各代码段的运行时间。

[0086] 本实施例中，变动后的目标代码的各代码段，包括未变动的代码段和发生变动的代码段。

[0087] 目标代码的变动，可以包括代码的增加、删除、替换、顺序改变等。

[0088] 本实施例中，确定变动后的目标代码的各代码段的运行时间，从而能够根据运行时间增量对应的代码段与发生变动的代码段之间的关系，确定运行时间增加是否正常。

[0089] 在一种实施方式中，如图3所示，代码处理方法还包括：

[0090] 步骤S31：根据运行时间的时间增量对应的代码段和目标代码发生变动的代码段，确定时间增量是否正常。

[0091] 本实施例中，确定时间增量是否正常，从而可以确定迭代后的代码在运行时间方面的质量是否符合要求，保证代码迭代的效果。

[0092] 在一种实施方式中，根据运行时间的时间增量对应的代码段和目标代码发生变动的代码段，确定时间增量是否正常，包括：

[0093] 在运行时间的时间增量对应的代码段非目标代码发生变动的代码段的情况下，确定时间增量异常。

[0094] 本实施例中，确定时间增量异常的情况下，可以对异常原因进行排查与清除，保证代码迭代的质量。

[0095] 在一种实施方式中，代码处理方法还包括：

[0096] 在运行时间的时间增量的代码段为目标代码发生变动的代码段运行时序之后的代码段、且运行时间的时间增量的代码段与目标代码发生变动的代码段存在时序依赖关系的情况下，确定运行时间的时间增量对应的代码段非目标代码发生变动的代码段；

[0097] 和/或，在运行时间的时间增量的代码段与目标代码发生变动的代码段不存在时序依赖关系的情况下，确定运行时间的时间增量对应的代码段非目标代码发生变动的代码段。

[0098] 本公开实施例中，通过白盒代码分析来自动生成增量代码所影响的耗时的代码段，彻底释放了通过人工CR(Code Review，代码复核)代码来分析模块性能的人工成本。

[0099] 本公开一种示例中，代码处理方法包括如图4A所示的步骤：

[0100] 步骤S41：底层工具适配。

[0101] 本公开示例可采用任何具有静态代码扫描的工具进行目标代码变动前和变动后的静态扫描。为了能够让使工具支持识别目标代码中每个执行阶段的起始位置和结束位置，需要对此目标代码的静态扫描工具进行适配，使得静态扫描工具按照适配的特征确定目标代码中的起始位置和结束位置。

[0102] 在一种具体的实现方式中，可以采用Dapper框架进行特征。Dapper是.NET下的一种ORM框架。.NET是微软当代的操作平台，它允许人们在其上构建各种应用方式，使人们尽可能通过简单的方式，多样化地、最大限度地从网站获取信息，解决网站之间的协同工作，并打破计算机、设备、网站、各大机构和工业界间的障碍——即所谓的“数字孤岛”，从而实现因特网的全部潜能，搭建起第三代互联网平台。后缀为net是网络服务公司，为个人或商业提供服务。针对ORM，其中O是Object(对象)；R是Relation(关系)；M是Mapping(映射)；ORM

是一种对象关系映射的技术。

[0103] 在底层工具适配阶段，可以在Dapper框架下，进行指定模块的代码特征标注。指定模块的代码即目标代码。

[0104] 步骤S42：代码扫描。每次迭代时，采用静态代码扫描工具对指定模块的全量代码和增量代码进行静态代码扫描，扫描结果可输出为Json文件，后续步骤中对扫描结果进行二次分析。

[0105] 每次迭代升级时，对所属模块进行全量、增量两次静态代码扫描，得到全量、增量代码特征(全量代码特征和增量代码特征可以包括宏定义、静态代码特征、函数调用)，续步骤中对此结果进行二次分析。

[0106] 耗时预判的实现需要在业务代码中对耗时的代码段的起始位置和结束位置进行标注(特征标注)，并且在静态代码扫描(例如Cppcheck)工具中对标注代码进行特征识别，需要在业务代码和Cppcheck代码中进行适配开发(特征开发)。

[0107] 分析业务代码到耗时的代码段的映射关系，要在Cppcheck中识别出每个耗时的代码段的起始和结束，需要固定的代码特征来标识，以feed-gr业务线为例，我们在业务代码中加入了宏定义来标识每个阶段和开始和结束。比如，设置目标代码的起始位置的代码特征(起始标记)为：FEED_DAPPER_FEATURE_BEGIN；设置结束位置的代码特征(结束标记)为：FEED_DAPPER_FEATURE_END。

[0108] 在一种具体实现方式中，实现依赖静态代码扫描工具可以为Cppcheck，使用Cppcheck进行代码扫描，可以得到代码特征、函数间的调用关系等，将Cppcheck结果输出为Json文件，对Json文件进行分析，可以得到全局或者某一代码段内的函数调用链关系。

[0109] 在一种具体的实现方式中，通过解析全量代码的特征文件得到所有耗时统计阶段在代码中的位置，包括每个阶段的起始行、结束行、所在函数、所在文件等信息，如下表1所示，展示了部分节点的起始位置和结束位置，以及所在函数和文件。

[0110]

节点	文件	函数	起始位置	结束位置
目标代码中的节点 1（比如行号）	文件 1	funcA	位置 1	位置 2

[0111] 表1

[0112] 全量代码扫描的结果中还包含代码段中每个函数的调用关系，可以分析得到每个耗时的代码段内所有执行到的函数，将全量代码和耗时的代码段与函数进行映射。比如表2所示的代码段和其全部对应的函数的映射关系。

[0113]

代码段	函数

[0114]

代码段 1	funcA
代码段 1	funcC
代码段 2	funcA
代码段 3	funcD

[0115] 表2

[0116] 步骤S43:耗时的代码段预估。本步骤中,可预先估计改动的代码影响的代码段。

[0117] 通过对指定模块的目标代码进行全量代码扫描的结果,可以得出模块全部执行阶段的起始位置(Begin)和结束位置(End)以及每个代码段的函数调用信息;通过对增量代码的分析,可以得出增量代码的代码特征,得到增量代码所对应的代码段。将增量代码和全量阶段进行映射,可以预估得出增量代码所影响的耗时的代码段。

[0118] 比如,增量代码为函数的增量代码,代码变动前为函数funcA,变动后为函数funcB。通过变动前后,funcA和funcB的变化(Difference),得到函数funcB为增量代码对应的函数。通过函数funcB在目标代码中的被调用关系,将变动的函数映射到对应的代码段。

[0119] 耗时预估根据全量代码扫描和增量代码扫描的结果进行确定。增量代码扫描和全量的区别在于增量结果中只包含了增量代码的特征,也就是变更代码的函数名、文件名等。

[0120] 在增量代码扫描时,增量代码(diff_code)可能直接位于某个耗时的代码段内;增量代码也可能位于函数中,该函数在某个耗时的代码段内被调用。前者可以称为显示的变动;后者可称为隐式的变动。

[0121] 显示的变动首先可参照图4B所示,在名称为a.cc的模块中,代码段“mmr”处于目标代码的120行到160行,增量代码“diff_code”处于目标代码的130到150行,那么可以认为增量代码所直接影响的耗时的代码段为mmr阶段,显示变动还包括增量代码开始于mmr起始位置之前的位置,但是行号仍处于160-180的行号之间,如图4C所示。或者结束于mmr结束之后的位置,但是行号仍处于160-180的行号之间,如图4D所示。

[0122] 隐式的变动可参照图4E所示。增量代码改动在A函数,A函数内没有直接运行的耗时代码段,而B函数调用了A函数,所以增量代码因为函数调用的关系间接作用在B函数所运行的代码段。

[0123] 综上,增量代码直接影响和间接影响到的耗时节点,在耗时预估时,可通过表格直观展示直接影响的代码段和间接影响的代码段。如下表3和

[0124] 表4所示。

[0125]

增量函数名	直接影响的耗时的代码段	源文件
funcB	阶段 1	源文件 1

[0126] 表3

[0127]

增量函数名	间接影响的耗时的代码段	源文件
funcC	阶段 2	源文件 2
funcC	阶段 1	源文件 2
funcC	阶段 3	源文件 2
funcB	阶段 2	源文件 1
funcB	阶段 3	源文件 1

[0128] 表4

[0129] 本示例以增量代码为例，在代码发生删减、替换、改变依赖包等改动操作时，可参照本示例执行代码处理方法。

[0130] 步骤S44：异常波动消除。耗时的代码段预估之后结合性能测试各个阶段指标数据以及指定模块的目标代码执行的代码段的流程拓扑，可以对异常指标进行矫正。比如，性能测试的结果存在运行时间异常，时间增量超过人工设定的阈值，判断新增代码对应的代码段是否为异常代码段，则为代码改动引起可对代码进行矫正。

[0131] 在不能完全消除性能测试延迟抖动的所有来源时，就会出现执行阶段的耗时波动，这种情况下需要采用耗时预估结果结合模块内部执行流程来矫正性能测试的异常阶段指标，从而消除一些非正常的波动。

[0132] 如图4F所示，每个方框代表一个代码段。图4F模拟了一个模块内部阶段间的执行和调度流程，假设性能测试结果显示代码段G耗时上涨，就可以根据白盒代码分析的结果对此异常进行矫正，来判断性能测试结果是否符合代码改动预期。

[0133] 在一种可能的实现方式中，仍然参照图4F，假设白盒代码分析得到的改动阶段为代码段C，通过模块执行阶段的流程拓扑判断G阶段执行在改动阶段之前，那么认为本次运行时间波动是由于代码改动导致，符合耗时的代码段预估。

[0134] 在另一种可能的实现方式中，仍然参照图4F，假设白盒代码分析得到的改动阶段为B、D、E、F、J、K、H、I、M、N等阶段，从图中可以的出代码改动阶段与异常阶段不存在调度依赖关系，那么认为本次运行时间波动非代码改动导致，不符合耗时节点预估。

[0135] 在另一种可能的实现方式中，仍然参照图4F，假设白盒分析得出的代码改动阶段为G、L，而性能测试异常阶段为G，异常阶段正好为代码改动所影响的阶段，那么认为本次运行时间波动是由于代码改动导致，符合耗时的代码段预估。

[0136] 本公开实施例还提供一种代码处理装置，如图5所示，包括：

[0137] 归属信息模块51，用于获得目标代码的基本归属信息；

[0138] 代码段模块52，用于根据基本归属信息，获得目标代码所包括的代码段；

[0139] 时间增量模块53，用于在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段。

[0140] 在一种实施方式中，如图6所示，归属信息模块包括：

[0141] 静态扫描单元61，用于对目标代码进行静态扫描，获得目标代码的原有内容的基本归属关系。

[0142] 在一种实施方式中，如图7所示，代码段模块包括：

[0143] 分段标记单元71，用于获取目标代码的分段标记；

[0144] 分段标记处理单元72，用于根据分段标记和目标代码的原有内容的基本归属关系，获得目标代码所包括的代码段。

[0145] 在一种实施方式中，如图8所示代码处理装置还包括：

[0146] 变动前运行时间模块81，用于确定变动前的目标代码的各代码段的运行时间。

[0147] 在一种实施方式中，如图9所示，时间增量模块包括：

[0148] 变动后运行时间单元91，用于在目标代码变动的情况下，确定变动后的目标代码的各代码段的运行时间；

[0149] 增量对应单元92，用于根据变动前的目标代码的各代码段的运行时间和变动后的目标代码的各代码段的运行时间，获得运行时间增量对应的代码段。

[0150] 在一种实施方式中，变动后运行时间单元还用于：

[0151] 在目标代码变动的情况下，对目标代码进行重新的静态扫描，获得目标代码的变动内容的基本归属关系；

[0152] 根据变动内容的基本归属关系，确定变动后的目标代码的各代码段；

[0153] 根据变动后的目标代码的各代码段，确定变动后的目标代码的各代码段的运行时间。

[0154] 在一种实施方式中，如图10所示，代码处理装置还包括：

[0155] 状态模块101，用于根据运行时间的时间增量对应的代码段和目标代码发生变动的代码段，确定时间增量是否正常。

[0156] 在一种实施方式中，状态模块还用于：

[0157] 在运行时间的时间增量对应的代码段非目标代码发生变动的代码段的情况下，确定时间增量异常。

[0158] 在一种实施方式中，如图11所示，代码处理装置还包括：

[0159] 第一增量代码段模块111，用于在运行时间的时间增量的代码段为目标代码发生变动的代码段运行时序之后的代码段、且运行时间的时间增量的代码段与目标代码发生变动的代码段存在时序依赖关系的情况下，确定运行时间的时间增量对应的代码段非目标代码发生变动的代码段；

[0160] 和/或，第二增量代码段模块112，用于在运行时间的时间增量的代码段与目标代码发生变动的代码段不存在时序依赖关系的情况下，确定运行时间的时间增量对应的代码段非目标代码发生变动的代码段。

[0161] 本公开实施例各装置中的各单元、模块或子模块的功能可以参见上述数据处理方法中的对应描述，在此不再赘述。

[0162] 根据本公开的实施例，本公开还提供了一种电子设备、一种可读存储介质和一种计算机程序产品。

[0163] 图12示出了可以用来实施本公开的实施例的示例电子设备120的示意性框图。电子设备旨在表示各种形式的数字计算机，诸如，膝上型计算机、台式计算机、工作台、个人数字助理、服务器、刀片式服务器、大型计算机、和其它适合的计算机。电子设备还可以表示各种形式的移动装置，诸如，个人数字处理、蜂窝电话、智能电话、可穿戴设备和其它类似的计算装置。本文所示的部件、它们的连接和关系、以及它们的功能仅仅作为示例，并且不意在限制本文中描述的和/或要求的本公开的实现。

[0164] 如图12所示，电子设备120包括计算单元121，其可以根据存储在只读存储器(ROM)122中的计算机程序或者从存储单元128加载到随机访问存储器(RAM)123中的计算机程序来执行各种适当的动作和处理。在RAM123中，还可存储电子设备120操作所需的各种程序和数据。计算单元121、ROM122以及RAM123通过总线124彼此相连。输入输出(I/O)接口125也连接至总线124。

[0165] 电子设备120中的多个部件连接至I/O接口125，包括：输入单元126，例如键盘、鼠标等；输出单元127，例如各种类型的显示器、扬声器等；存储单元128，例如磁盘、光盘等；以

及通信单元129，例如网卡、调制解调器、无线通信收发机等。通信单元129允许电子设备120通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据。

[0166] 计算单元121可以是各种具有处理和计算能力的通用和/或专用处理组件。计算单元121的一些示例包括但不限于中央处理单元(CPU)、图形处理单元(GPU)、各种专用的人工智能(AI)计算芯片、各种运行机器学习模型算法的计算单元、数字信号处理器(DSP)、以及任何适当的处理器、控制器、微控制器等。计算单元121执行上文所描述的各个方法和处理，例如代码处理方法。例如，在一些实施例中，代码处理方法可被实现为计算机软件程序，其被有形地包含于机器可读介质，例如存储单元128。在一些实施例中，计算机程序的部分或者全部可以经由ROM122和/或通信单元129而被载入和/或安装到电子设备120上。当计算机程序加载到RAM123并由计算单元121执行时，可以执行上文描述的代码处理方法的一个或多个步骤。备选地，在其他实施例中，计算单元121可以通过其他任何适当的方式(例如，借助于固件)而被配置为执行代码处理方法。

[0167] 本文中以上描述的系统和技术的各种实施方式可以在数字电子电路系统、集成电路系统、场可编程门阵列(FPGA)、专用集成电路(ASIC)、专用标准产品(ASSP)、芯片上系统的系统(SOC)、负载可编程逻辑设备(CPLD)、计算机硬件、固件、软件、和/或它们的组合中实现。这些各种实施方式可以包括：实施在一个或者多个计算机程序中，该一个或者多个计算机程序可在包括至少一个可编程处理器的可编程系统上执行和/或解释，该可编程处理器可以是专用或者通用可编程处理器，可以从存储系统、至少一个输入装置、和至少一个输出装置接收数据和指令，并且将数据和指令传输至该存储系统、该至少一个输入装置、和该至少一个输出装置。

[0168] 用于实施本公开的方法的程序代码可以采用一个或多个编程语言的任何组合来编写。这些程序代码可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器或控制器，使得程序代码当由处理器或控制器执行时使流程图和/或框图中所规定的功能/操作被实施。程序代码可以完全在机器上执行、部分地在机器上执行，作为独立软件包部分地在机器上执行且部分地在远程机器上执行或完全在远程机器或服务器上执行。

[0169] 在本公开的上下文中，机器可读介质可以是有形的介质，其可以包含或存储以供指令执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的程序。机器可读介质可以是机器可读信号介质或机器可读储存介质。机器可读介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备，或者上述内容的任何合适组合。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便捷式紧凑盘只读存储器(CD-ROM)、光学储存设备、磁储存设备、或上述内容的任何合适组合。

[0170] 为了提供与用户的交互，可以在计算机上实施此处描述的系统和技术，该计算机具有：用于向用户显示信息的显示装置(例如，CRT(阴极射线管)或者LCD(液晶显示器)监视器)；以及键盘和指向装置(例如，鼠标或者轨迹球)，用户可以通过该键盘和该指向装置来将输入提供给计算机。其它种类的装置还可以用于提供与用户的交互；例如，提供给用户的反馈可以是任何形式的传感反馈(例如，视觉反馈、听觉反馈、或者触觉反馈)；并且可以用任何形式(包括声输入、语音输入、或者触觉输入来接收来自用户的输入。

[0171] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如,作为数据服务器)、或者包括中间件部件的计算系统(例如,应用服务器)、或者包括前端部件的计算系统(例如,具有图形用户界面或者网络浏览器的用户计算机,用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术的实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如,通信网络)来将系统的部件相互连接。通信网络的示例包括:局域网(LAN)、广域网(WAN)和互联网。

[0172] 计算机系统可以包括客户端和服务器。客户端和服务器一般远离彼此并且通常通过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计算机程序来产生客户端和服务器的关系。

[0173] 应该理解,可以使用上面所示的各种形式的流程,重新排序、增加或删除步骤。例如,本公开中记载的各步骤可以并行地执行也可以顺序地执行也可以不同的次序执行,只要能够实现本公开公开的技术方案所期望的结果,本文在此不进行限制。

[0174] 上述具体实施方式,并不构成对本公开保护范围的限制。本领域技术人员应该明白的是,根据设计要求和其他因素,可以进行各种修改、组合、子组合和替代。任何在本公开的精神和原则之内所作的修改、等同替换和改进等,均应包含在本公开保护范围之内。

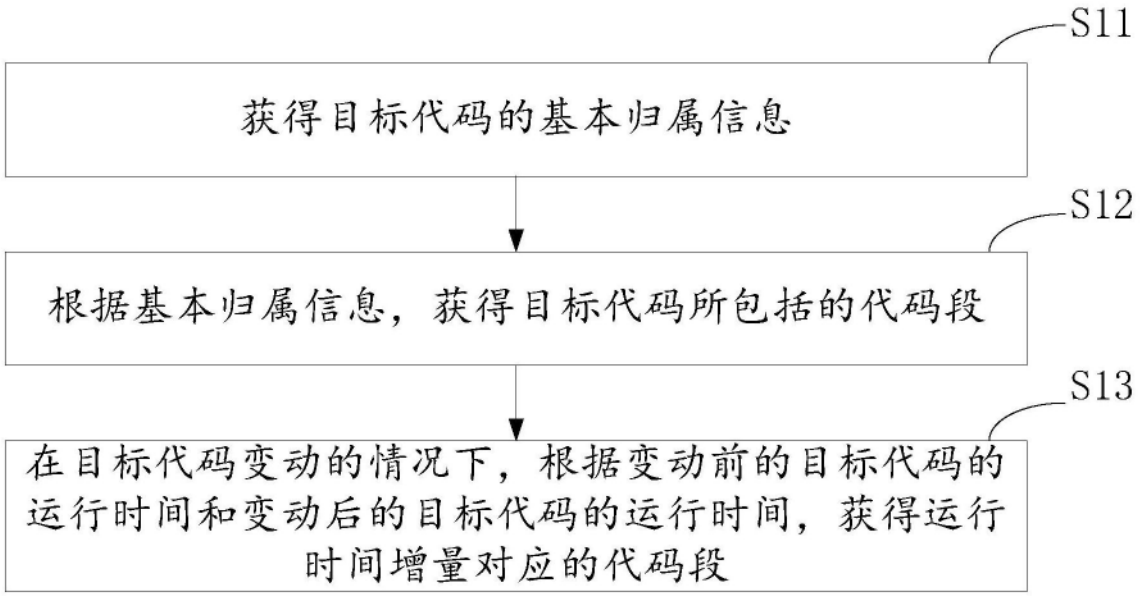

S11
获得目标代码的基本归属信息
S12
根据基本归属信息，获得目标代码所包括的代码段
S13
在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段

图1

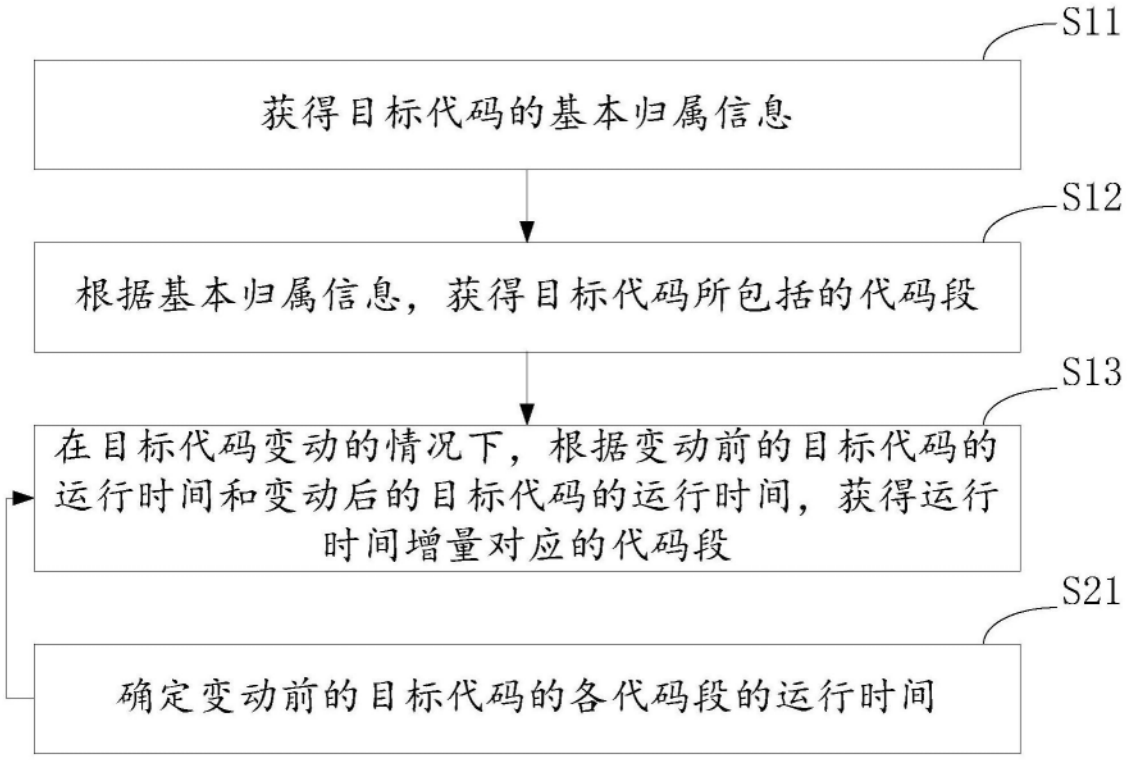

S11
获得目标代码的基本归属信息
S12
根据基本归属信息，获得目标代码所包括的代码段
S13
在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段
S21
确定变动前的目标代码的各代码段的运行时间

图2

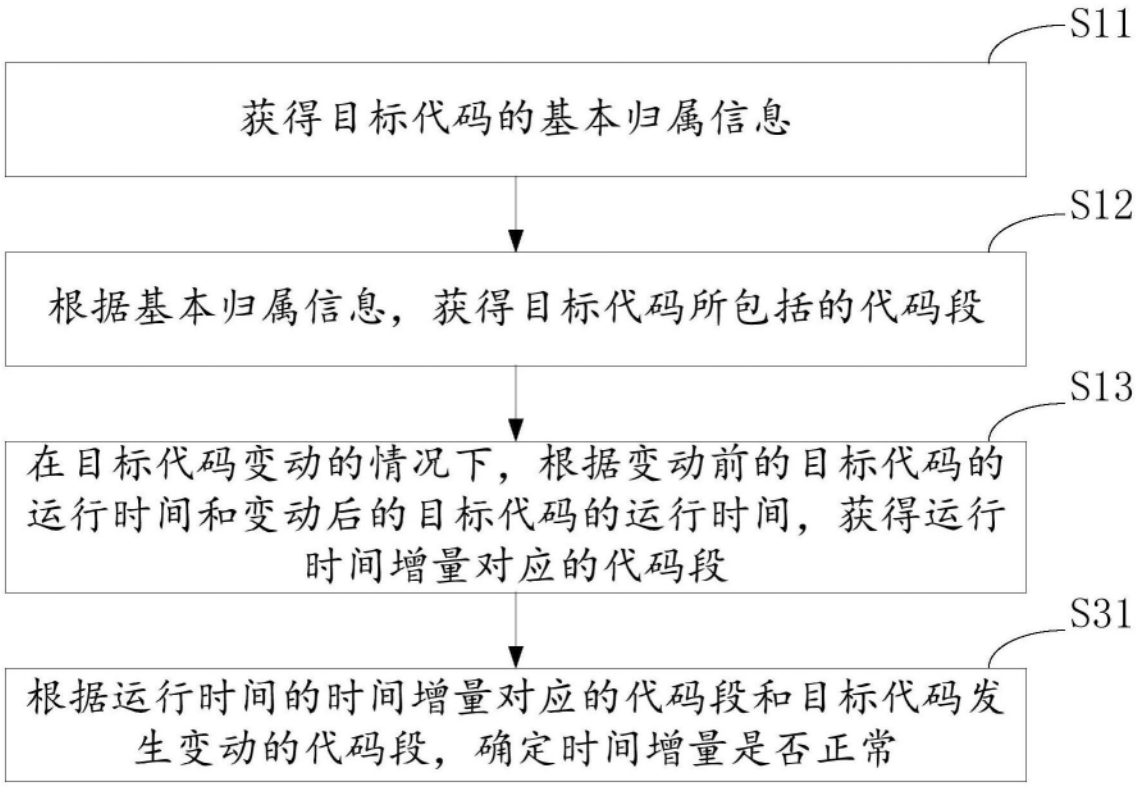

S11
获得目标代码的基本归属信息
S12
根据基本归属信息，获得目标代码所包括的代码段
S13
在目标代码变动的情况下，根据变动前的目标代码的运行时间和变动后的目标代码的运行时间，获得运行时间增量对应的代码段
S31
根据运行时间的时间增量对应的代码段和目标代码发生变动的代码段，确定时间增量是否正常

图3

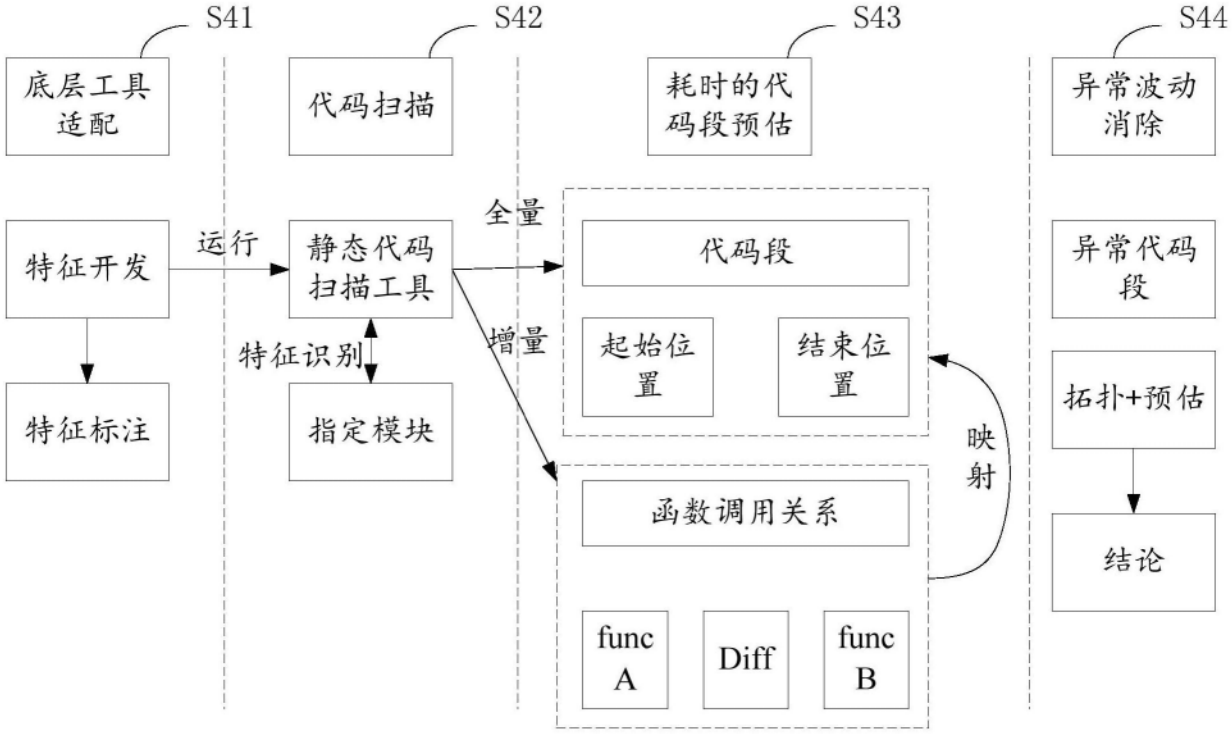

S41
S42
S43
S44
底层工具适配
代码扫描
耗时的代码段预估
异常波动消除
特征开发
运行
静态代码扫描工具
全量
增量
代码段
起始位置
结束位置
异常代码段
特征识别
特征标注
指定模块
拓扑+预估
映射
函数调用关系
结论
func A
Diff
func B

图4A


a.cc
mmr起始标记
160行
mmr代码起始
增量代码
mmr代码结束
mmr结束标记
180行

图4B


a.cc
增量代码
170行
mmr起始标记
160行
mmr代码起始
增量代码
mmr代码结束
mmr结束标记
180行

图4C


a. cc
mmr起始标记
160行
mmr代码起始
增量代码
mmr代码结束
mmr结束标记
180行
增量代码
170行

图4D

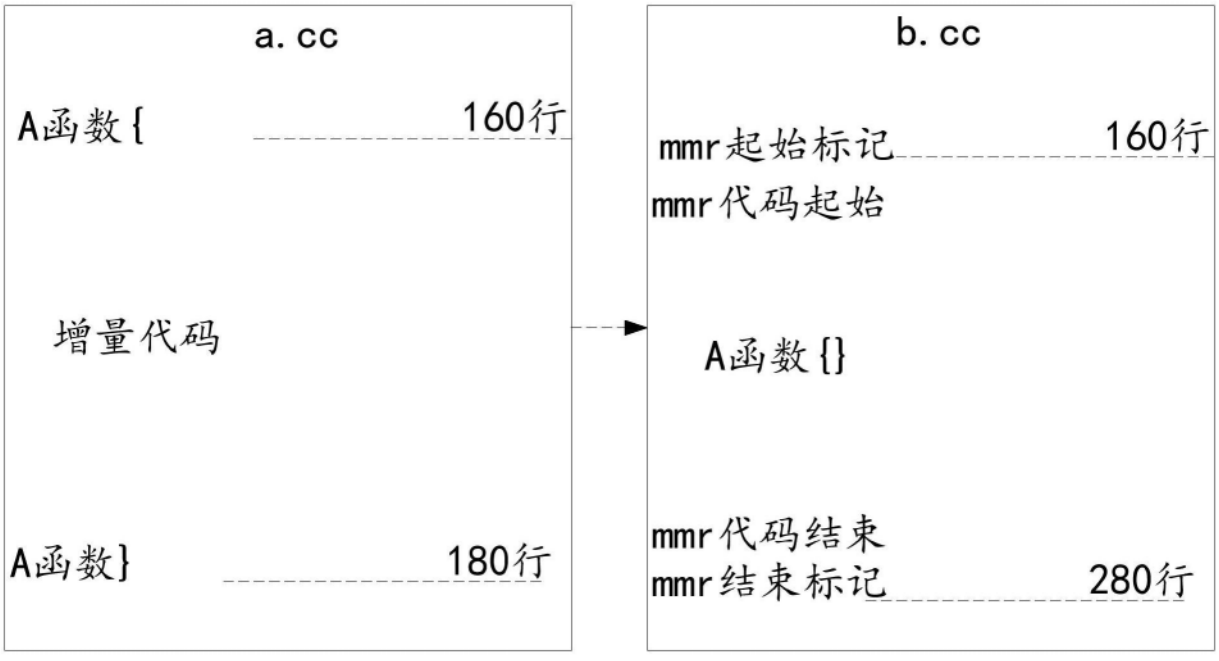

a. cc
A函数{
160行
增量代码
A函数}
180行
b. cc
mmr起始标记
160行
mmr代码起始
A函数{}
mmr代码结束
mmr结束标记
280行

图4E

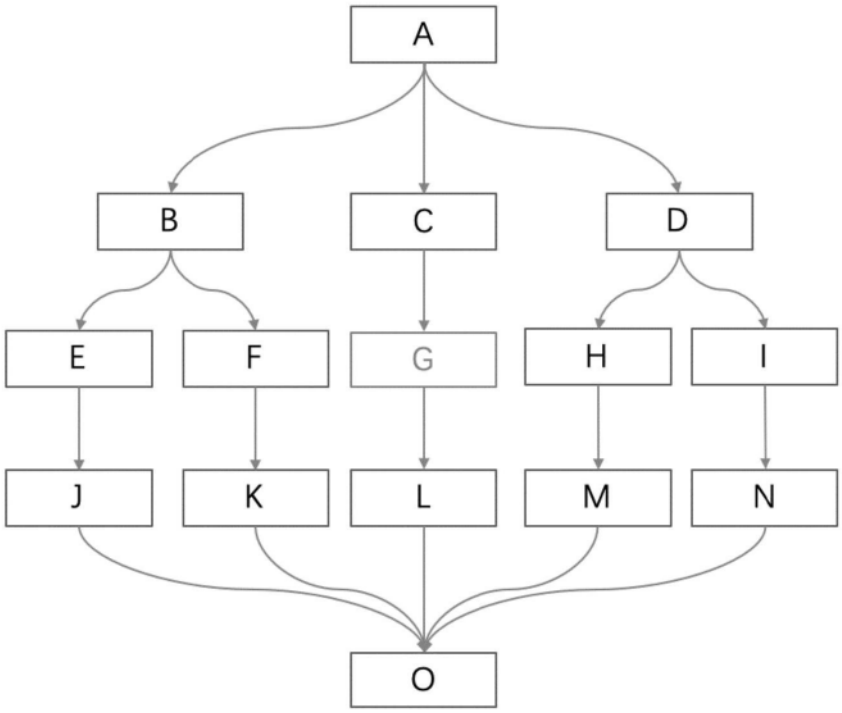


图4F

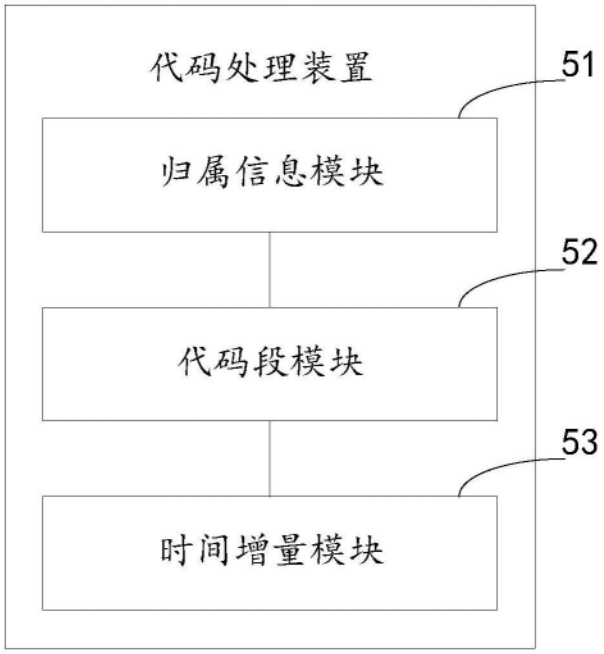


图5

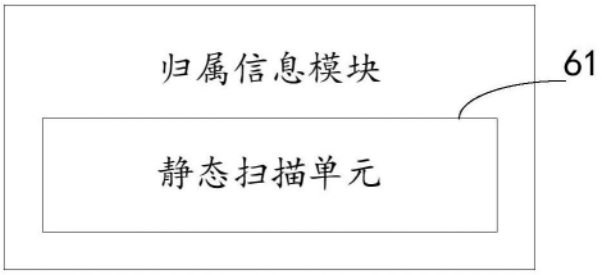

归属信息模块
61
静态扫描单元

图6

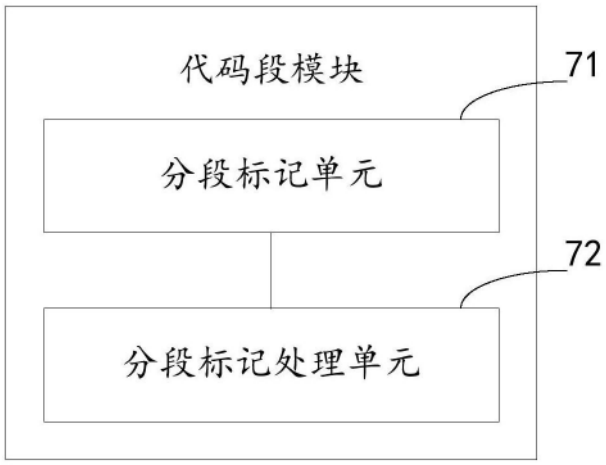

代码段模块
71
分段标记单元
72
分段标记处理单元

图7

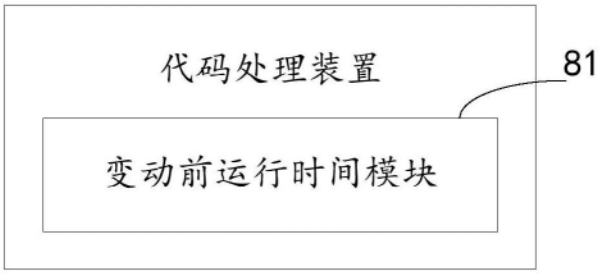

代码处理装置
81
变动前运行时间模块

图8

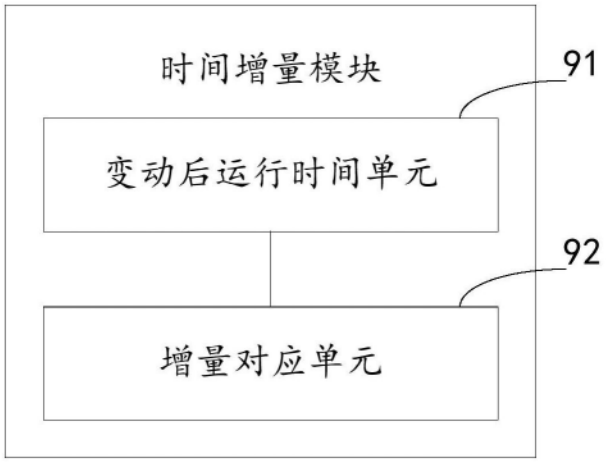

时间增量模块
91
变动后运行时间单元
92
增量对应单元

图9


代码处理装置
101
状态模块

图10

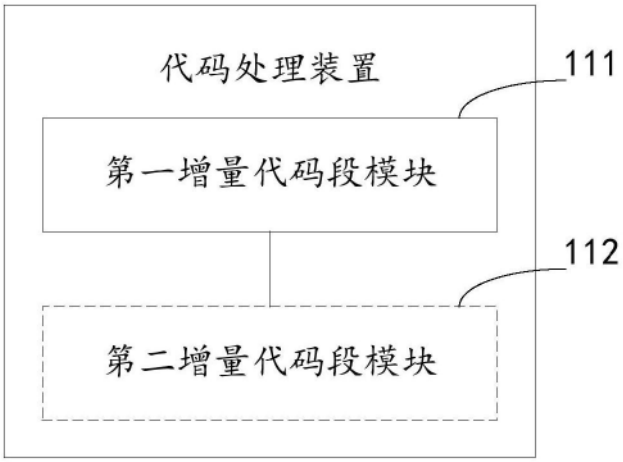

代码处理装置
111
第一增量代码段模块
112
第二增量代码段模块

图11

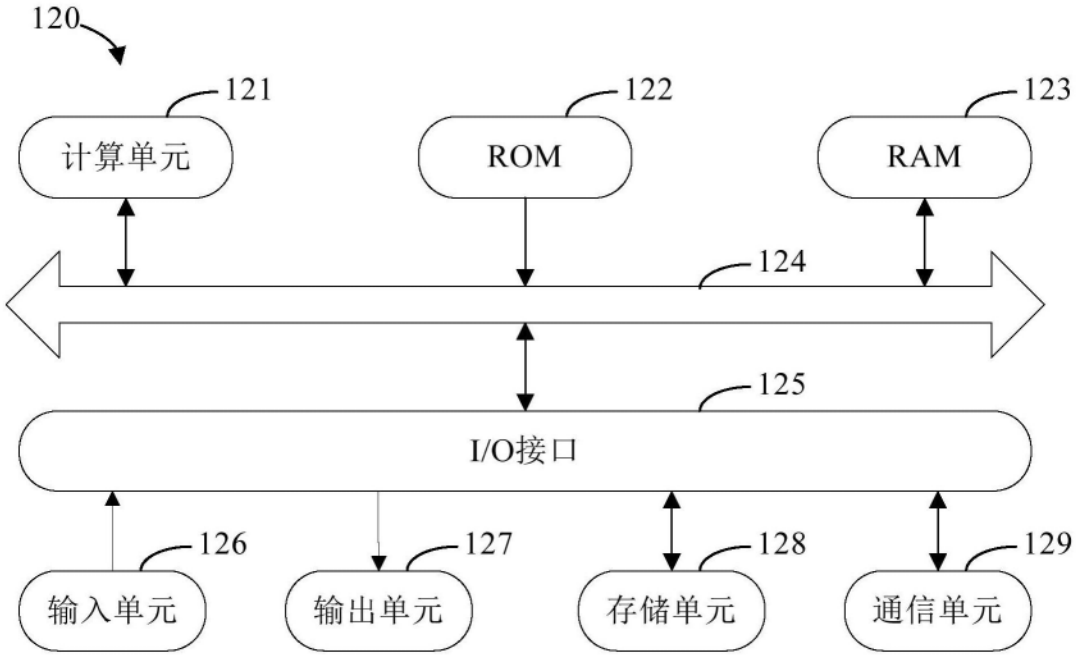

120
121
计算单元
122
ROM
123
RAM
124
125
I/O接口
126
输入单元
127
输出单元
128
存储单元
129
通信单元

图12