(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 109313601 B
(45)授权公告日 2022.05.24

(21)申请号 201780036501.7

(22)申请日 2017.05.08

(65)同一申请的已公布的文献号
申请公布号 CN 109313601 A

(43)申请公布日 2019.02.05

(30)优先权数据
62/333,786 2016.05.09 US
62/333,798 2016.05.09 US
62/333,804 2016.05.09 US
62/333,809 2016.05.09 US
62/333,811 2016.05.09 US
62/340,256 2016.05.23 US
15/588,521 2017.05.05 US
15/588,523 2017.05.05 US
15/588,526 2017.05.05 US
15/588,531 2017.05.05 US

(85)PCT国际申请进入国家阶段日
2018.12.12

(86)PCT国际申请的申请数据
PCT/US2017/031593 2017.05.08

(87)PCT国际申请的公布数据
WO2017/196748 EN 2017.11.16

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 E·S·钱

(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
专利代理师 周衡威

(51)Int.Cl.
*G06F 11/30*(2006.01)
*G06F 11/34*(2006.01)
*G06F 11/36*(2006.01)
*G06F 12/02*(2006.01)

(56)对比文件
US 2014310235 A1,2014.10.16
CN 1829977 A,2006.09.06
CN 105190564 A,2015.12.23
CN 105144112 A,2015.12.09

审查员 刘凤霞

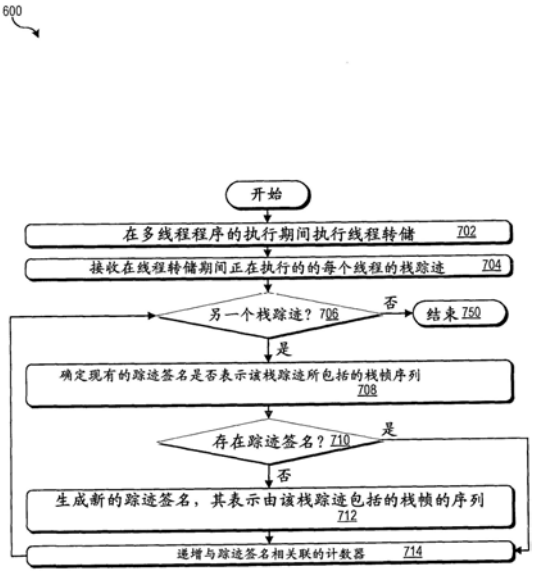
权利要求书5页 说明书52页 附图17页

(54)发明名称

# 用于编码栈踪迹信息的方法、系统和介质

(57)摘要

实施例提供了一种线程归类方法，该方法使用归类签名以紧凑的形式表示栈踪迹。一些实施例可以接收包括栈帧序列的栈踪迹。一些实施例可以基于该栈帧序列生成表示集合的踪迹签名。一些实施例可以接收一个或多个后续栈踪迹。对于一个或多个后续栈踪迹中的每一个，一些实施例可以确定是否已生成表示该后续栈踪迹内包括的栈帧序列的后续踪迹签名。如果不是，则一些实施例可以基于踪迹签名和基于该踪迹签名生成了的其它后续踪迹签名生成表示后续栈帧序列的后续踪迹签名。




CN 109313601 B

1.一种计算机实现的方法,包括:

由计算机接收包括第一栈帧序列的第一栈踪迹;

至少部分地基于所述第一栈帧序列生成表示所述第一栈帧序列的第一踪迹签名;

由计算机接收一个或多个后续栈踪迹;以及

对于所述一个或多个后续栈踪迹中的至少一个后续栈踪迹:

确定是否已生成表示所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹所包括的后续栈帧序列的后续踪迹签名;以及

至少部分地基于确定尚未生成所述后续踪迹签名,执行以下操作:

根据所述一个或多个后续栈踪迹中的至少一个后续栈踪迹,确定包括第二栈帧序列的第二栈踪迹签名;

确定所述第一栈帧序列与所述第二栈帧序列共享匹配的栈帧序列;

至少部分地基于所述匹配的栈帧序列来确定第二踪迹签名;以及

至少部分地基于所述第二踪迹签名以及至少部分地基于所述第二踪迹签名生成了的其它后续踪迹签名,生成表示所述后续栈帧序列的所述后续踪迹签名。

2.如权利要求1所述的计算机实现的方法:

其中,生成第二踪迹签名包括:

确定第一栈帧序列中包括的栈帧的非匹配子序列与第二栈帧序列中包括的栈帧的非匹配子序列不同;

生成表示栈帧的匹配子序列的第一段签名;

生成表示包括在第二栈帧序列中的栈帧的非匹配子序列的第二段签名;以及

生成第二踪迹签名,其中第二踪迹签名包括第一段签名和第二段签名。

3.如权利要求2所述的计算机实现的方法,其中,生成第二踪迹签名还包括:

生成对包括在第一栈帧序列中的栈帧的非匹配子序列进行归类的第三段签名;以及

修改第一踪迹签名以包括第一段签名和第三段签名。

4.如权利要求2所述的计算机实现的方法,其中,第一踪迹签名与第一计数器相关联,所述第一计数器在接收到包括第一栈帧序列的任何栈踪迹时递增,并且第二踪迹签名与第二计数器相关联,所述第二计数器在接收到包括第二栈帧序列的任何栈踪迹时递增。

5.如权利要求4所述的计算机实现的方法,

其中,第一段签名与第三计数器相关联,所述第三计数器在接收到包括栈帧的所述匹配子序列的任何栈踪迹时递增;

其中,第二段签名与第四计数器相关联,所述第四计数器在接收到包括第二栈帧序列中包括的栈帧的非匹配子序列的任何栈踪迹时递增;

其中,当生成第一段签名时,第三计数器复制自第一计数器;以及

其中,当生成第二段签名时,第四计数器复制自第二计数器。

6.如权利要求1所述的计算机实现的方法,其中,所述第一踪迹签名与第一元组对应,其中所述第一元组包括第一二叉树,所述第一二叉树包括作为所述第一二叉树的根节点的第一节点,其中所述第一节点表示所述第一栈帧序列。

7.如权利要求6所述的计算机实现的方法,

其中,确定是否已生成表示所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹

所包括的后续栈帧序列的后续踪迹签名包括确定一个或多个先前生成的节点是否表示所述后续栈帧序列,所述一个或多个先前生成的节点包括所述第一节点;以及

其中,生成表示后续栈帧序列的后续踪迹签名包括:

确定包括在未由所述一个或多个先前生成的节点表示的后续栈帧序列中的栈帧的一个或多个子序列;

生成表示包括在后续栈帧序列中的栈帧的所述一个或多个子序列的一个或多个附加节点;

将所述一个或多个附加节点中的至少一个附加节点并入到一个或多个先前生成的元组的一个或多个先前生成的二叉树中,所述一个或多个先前生成的二叉树包括所述第一二叉树,并且所述一个或多个先前生成的元组包括所述第一元组;

生成一个或多个附加二叉树,其中,所述一个或多个附加二叉树中的至少一个附加二叉树包括所述一个或多个附加节点中的至少一个附加节点;以及

生成包括所述一个或多个附加二叉树的附加元组,其中所述后续踪迹签名与所述附加元组对应。

8.如权利要求7所述的计算机实现的方法,其中,每个元组、每个节点和每个栈帧由整数唯一地标识。

9.如权利要求6所述的计算机实现的方法,

其中生成第二踪迹签名包括:

确定第一栈帧序列和第二栈帧序列共享栈帧的匹配子序列,并且第一栈帧序列中包括的栈帧的非匹配子序列与第二栈帧序列中包括的栈帧的非匹配子序列不同;

生成表示栈帧的所述匹配子序列的第二节点;

生成表示包括在第二栈帧序列中的栈帧的非匹配子序列的第三节点;

生成包括第二节点的第二二叉树和包括第三节点的第三二叉树;以及

生成包括第二二叉树和第三二叉树的第二元组,其中第二节点和第三节点是同级节点,并且第二踪迹签名与第二元组对应。

10.如权利要求9所述的计算机实现的方法,其中,生成第二踪迹签名还包括:

生成表示栈帧的所述匹配子序列的第四节点;

生成表示包括在第一栈帧序列中的栈帧的非匹配子序列的第五节点;以及

将第四节点和第五节点添加到第一二叉树作为第一节点的子节点。

11.一种计算系统,包括:

一个或多个处理器;以及

所述一个或多个处理器能够访问的存储器,所述存储器存储一条或多条指令,所述一条或多条指令当由所述一个或多个处理器执行时使所述一个或多个处理器:

接收包括第一栈帧序列的栈踪迹;

至少部分地基于所述第一栈帧序列生成表示所述第一栈帧序列的第一踪迹签名,其中所述第一踪迹签名对应于第一元组,其中所述第一元组包括第一二叉树,所述第一二叉树包括作为所述第一二叉树的根节点的第一节点,其中所述第一节点表示所述第一栈帧序列;

接收一个或多个后续栈踪迹,所述一个或多个后续栈踪迹包括第二栈踪迹,所述第二

栈踪迹包括第二栈帧序列;以及

对于所述一个或多个后续栈踪迹中的至少一个后续栈踪迹:

确定是否已生成表示所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹所包括的后续栈帧序列的后续踪迹签名;以及

至少部分地基于确定尚未生成所述后续踪迹签名,执行以下操作:

至少部分地基于所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹,确定包括第二栈帧序列的第二栈踪迹签名;

至少部分地基于比较所述第一栈帧序列与所述第二栈帧序列,生成第二节点;

生成第二元组,第二元组包括第二二叉树,其中所述第二二叉树包括所述第二节点;

生成表示第二栈帧序列的第二踪迹签名,所述第二踪迹签名对应于所述第二元组;以及

至少部分地基于所述第一踪迹签名和至少部分地基于所述第一踪迹签名生成了的其它后续踪迹签名,生成表示所述后续栈帧序列的所述后续踪迹签名。

12.如权利要求11所述的计算系统,

其中,生成所述第二踪迹签名包括:

确定第一栈帧序列和第二栈帧序列共享栈帧的匹配子序列,并且第一栈帧序列中包括的栈帧的非匹配子序列与第二栈帧序列中包括的栈帧的非匹配子序列不同;

生成表示栈帧的所述匹配子序列的第一段签名;

生成表示包括在第二栈帧序列中的栈帧的非匹配子序列的第二段签名;以及

生成第二踪迹签名,其中第二踪迹签名包括第一段签名和第二段签名。

13.如权利要求11所述的计算系统,其中,生成所述第二踪迹签名包括:

确定所述第一栈帧序列与所述第二栈帧序列共享匹配的栈帧序列;以及

至少部分地基于所述匹配的栈帧序列生成第二踪迹签名。

14.如权利要求11所述的计算系统,

其中,确定是否已生成表示所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹所包括的后续栈帧序列后续踪迹签名包括确定一个或多个先前生成的节点是否表示后续栈帧序列,所述一个或多个先前生成的节点包括所述第一节点;以及

其中,生成表示后续栈帧序列的后续踪迹签名包括:

确定包括在未由所述一个或多个先前生成的节点表示的后续栈帧序列中的栈帧的一个或多个子序列;

生成表示包括在后续栈帧序列中的栈帧的所述一个或多个子序列的一个或多个附加节点;

将所述一个或多个附加节点中的至少一个附加节点并入到一个或多个先前生成的元组的一个或多个先前生成的二叉树中,所述一个或多个先前生成的二叉树包括所述第一二叉树并且所述一个或多个先前生成的元组包括所述第一元组;

生成一个或多个附加二叉树,其中,所述一个或多个附加二叉树中的至少一个附加二叉树包括所述一个或多个附加节点中的至少一个附加节点;以及

生成包括所述一个或多个附加二叉树的附加元组,其中所述后续踪迹签名与所述附加元组对应。

15.一种非瞬态计算机可读介质,存储一条或多条指令,所述一条或多条指令当由一个或多个处理器执行时使所述一个或多个处理器:

接收包括第一栈帧序列的第一栈踪迹;

至少部分地基于所述第一栈帧序列生成表示所述第一栈帧序列的第一踪迹签名,其中所述第一踪迹签名对应于第一元组,其中所述第一元组包括第一二叉树,所述第一二叉树包括作为所述第一二叉树的根节点的第一节点,其中所述第一节点表示所述第一栈帧序列;

由计算机接收一个或多个后续栈踪迹,所述一个或多个后续栈踪迹包括第二栈踪迹,所述第二栈踪迹包括第二栈帧序列;以及

对于所述一个或多个后续栈踪迹中的至少一个后续栈踪迹:

确定是否已生成表示所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹所包括的后续栈帧序列的后续踪迹签名;以及

至少部分地基于确定尚未生成所述后续踪迹签名,执行以下操作:

至少部分地基于所述一个或多个后续栈踪迹中的所述至少一个后续栈踪迹,确定包括第二栈帧序列的第二栈踪迹签名;

至少部分地基于比较所述第一栈帧序列与所述第二栈帧序列,生成第二节点;

生成第二元组,第二元组包括第二二叉树,其中所述第二二叉树包括所述第二节点;

生成表示第二栈帧序列的第二踪迹签名,所述第二踪迹签名对应于所述第二元组;以及

至少部分地基于所述第二踪迹签名和至少部分地基于所述第二踪迹签名生成了的其它后续踪迹签名,生成表示该后续栈帧序列的所述后续踪迹签名。

16.如权利要求15所述的非瞬态计算机可读介质,

其中,所述第一栈踪迹和所述一个或多个后续栈踪迹中的每一个与代表多线程程序执行的运行线程对应;并且

其中,响应于所述多线程程序的一个或多个线程转储,接收所述第一栈踪迹和所述一个或多个后续栈踪迹,所述一个或多个线程转储在所述多线程程序正在执行时在不同的时间点处被执行。

17.如权利要求15所述的非瞬态计算机可读介质,

其中,所述第一栈踪迹和所述一个或多个后续栈踪迹中的每一个与代表多线程程序执行的运行线程对应;

其中,从所述多线程程序的一个或多个线程转储中获得多个栈踪迹,所述一个或多个线程转储在所述多线程程序执行时在不同的时间点被执行;

其中,对所述多个栈踪迹进行重复以获得所述第一栈踪迹和所述一个或多个后续栈踪迹,所述第一栈踪迹和所述一个或多个后续栈踪迹各自包括不同的栈帧序列;

其中,确定所述第一栈踪迹和所述一个或多个后续栈踪迹中的每一个的发生的概率;以及

其中,所述第一栈踪迹和所述一个或多个后续栈踪迹是按发生的概率的顺序被接收的。

18.如权利要求15所述的非瞬态计算机可读介质,所述非瞬态计算机可读介质包括所

述一条或多条指令，所述一条或多条指令当由所述一个或多个处理器执行时使所述一个或多个处理器将所述第一踪迹签名发送到计算机，从而使得所述计算机能够重新创建具有所述第一踪迹签名的第一栈踪迹。

19.如权利要求15所述的非瞬态计算机可读介质，其中，每个栈帧包括与所述每个栈帧对应的被调用方法的方法名称和包括定义被调用方法的代码的源文件名称。

20.如权利要求19所述的非瞬态计算机可读介质，其中，一组霍夫曼编码被应用于由所述第一栈踪迹或所述一个或多个后续栈踪迹所包括的一个或多个栈帧所包括的一个或多个方法名称或一个或多个源文件名称，以压缩所述一个或多个栈帧。

# 用于编码栈踪迹信息的方法、系统和介质

[0001] 相关申请的交叉引用

[0002] 本申请是于2017年5月5日提交的标题为“Compression Techniques for Encoding Stack Trace Information”的美国非临时申请No.15/588,523、于2016年5月9日提交的标题为“Correlation of Thread Intensity and Heap Usage to Identify Heap-Hoarding Stack Traces”的美国临时申请No.62/333,786、于2016年5月9日提交的标题为“Memory Usage Determination Techniques”的美国临时申请No.62/333,798、于2016年5月9日提交的标题为“Compression Techniques for Encoding Stack Traces Information”的美国临时申请No.62/333,804、于2016年5月9日提交的标题为“Correlation of Stack Segment Intensity in Emergent Relationships”的美国临时申请No.62/333,811、于2016年5月9日提交的标题为“Systems and Methods of Stack Trace Analysis”的美国临时申请No.62/333,809、和于2016年5月23日提交的标题为“Characterization of Segments of Time-Series”的美国临时申请No.62/340,256、于2017年5月5日提交的标题为“Correlation of Thread Intensity And Heap Usage To Identify Heap-Hoarding Stack Traces”的美国非临时申请No.15/588,531、于2017年5月5日提交的标题为“Memory Usage Determination Techniques”的美国非临时申请No.15/588,526、和于2017年5月5日提交的标题为“Correlation of Stack Segment Intensity in Emergent Relationships”的美国非临时申请No.15/588,521的依据35 U.S.C.119(e)的权益和优先权的国际申请,这些申请的全部内容通过引用并入本文,用于所有目的。

[0003] 本申请涉及以下同时提交的申请,这些申请的全部内容通过引用并入本文,用于所有目的:

[0004] (1)于2017年5月8日提交的标题为“CORRELATION OF THREAD INTENSITY AND HEAP USAGE TO IDENTIFY HEAP-HOARDING STACK TRACES”的PCT非临时申请No./_,_,(代理人案卷号:088325-1047566(175200PC))。

[0005] (2)于2017年5月8日提交的标题为“MEMORY USAGE DETERMINATION TECHNIQUES”的PCT非临时申请No./_,_,(代理人案卷号:088325-1047576(175210PC))。

[0006] (3)于2017年5月8日提交的标题为“CORRELATION OF STACK SEGMENT INTENSITY IN EMERGENT RELATIONSHIPS”的PCT非临时申请No./_,_,(代理人案卷号:088325-1047582(175230PC))。

## 背景技术

[0007] 通常,云服务提供商维护操作资源以满足与客户的服务级别协议(SLA)。提供商不断监视他们提供的云服务的性能指标,以确保服务符合SLA。然而,由于可用工具可能缺乏预测或检测即将发生的SLA违规的能力,因此操作资源可能无法规避违规。此外,由于这些工具可能缺乏诊断SLA违规的根本原因的能力,因此当在此类违规确实发生时,操作可能需要较长的时间才能解决此类违规。因此,客户体验可能会受到不利影响。

[0008] 另外,这样的SLA可能要求系统地分析数据并且主动地对数据中的可操作信息采

取行动以避免SLA违规并且还确定是否满足协议。遵循服务级别协议和其它要求可能是非常繁重的,并且随着时间的推移会变得更加繁重。

[0009] 为了获得上述能力,需要的是代表使用高级别状态模型的系统的技术,其中基于系统的低级别事件和系统测量容易地更新该高级别状态模型。至于获得关于低级别事件的指标,可以工具化检测(instrument)系统底层的应用程序来收集事件的确切测量。然而,在这种方法中,工具化检测本身会影响测量。当围绕方法的工具化检测代码的执行时间在方法本身的执行时间中占主导地位时(例如,如果方法的调用计数高),该问题可能更加突出。

## 发明内容

[0010] 公开了用于基于从一系列线程转储中获得的栈踪迹对线程进行归类的某些技术。一些实施例可以通过合成和分析过程利用带标签的二叉树的元组对栈踪迹进行归类。

[0011] 一个实施例针对一种方法。该方法可以包括:由计算机接收包括栈帧序列的栈踪迹;至少部分地基于该栈帧序列生成表示该栈帧序列的踪迹签名;由计算机接收一个或多个后续栈踪迹;并且对于该一个或多个后续栈踪迹中的每一个:确定是否已生成表示后续栈踪迹所包括的后续栈帧序列的后续踪迹签名;并且如果尚未生成后续踪迹签名,则至少部分地基于踪迹签名和基于该踪迹签名生成了的其它后续踪迹签名生成表示后续栈帧序列的后续踪迹签名。

## 附图说明

[0012] 下面参考以下附图详细描述说明性实施例:

[0013] 图1描绘了以相对高频率的采样率在一段时间内的单个线程的示例性运行时剖析。

[0014] 图2描绘了示例性调用上下文树。

[0015] 图3描绘了根据一些实施例的虚拟机在一段时间内的示例性线程转储(dump)。

[0016] 图4-图6描绘了根据一些实施例的示例性线程归类签名。

[0017] 图7示出了简化的流程图,其描绘了根据一些实施例的响应于线程转储而生成和/或修改一个或多个线程归类签名。

[0018] 图8示出了简化的流程图,其描绘了响应于检测到分支点而生成或修改线程归类签名。

[0019] 图9示出了简化的流程图,其描绘了根据一些实施例的与高的堆使用量对应的代码的识别。

[0020] 图10示出了简化的流程图,其描绘了根据一些实施例的各种类的线程与高的堆使用量之间的相关程度的计算。

[0021] 图11描绘了示例曲线图,其中指派给样本测量值的权重跨示例数据集的时间范围相对于与样本测量值相关联的采样时间间隔而绘制。

[0022] 图12描绘了示出针对生产环境中的堆使用量通过不同线性回归技术导出的趋势图的示例图表。

[0023] 图13描绘了示出附加趋势图的示例图表,该附加趋势图图示了由标准鲁棒回归技术给出的不正确结果。

[0024] 图14示出了简化的流程图,其描绘了根据一些实施例的信号预测的生成。

[0025] 图15描绘了用于实现某些实施例的分布式系统的简化图。

[0026] 图16描绘了根据一些实施例的在其中可以将服务作为云服务供应的系统环境的一个或多个部件的简化框图。

[0027] 图17描绘了可以用于实现某些实施例的示例性计算机系统。

[0028] 该专利或申请文件包含至少一幅彩色附图。具有彩色附图的本专利或专利申请出版物的副本将在请求和支付必要费用后由主管局提供。

## 具体实施方式

[0029] I.概述

[0030] 在以下描述中,出于解释的目的,阐述了具体细节以便提供对本公开的实施例的透彻理解。然而,将清楚的是,可以在没有这些具体细节的情况下实践各种实施例。各图和描述不旨在是限制性的。

[0031] 本公开一般而言涉及使用堆使用量统计信息和线程强度统计信息来识别多线程进程(例如,应用程序)内的代码块以用于潜在的优化并预测未来的堆使用量和/或线程强度。线程强度统计信息可以用于跟踪进程的响应、负载和资源使用量,而无需对进程的底层代码进行工具化检测或使用代码注入。具体而言,线程的类型或栈段的类型的强度可以指示正由该线程执行或正被该栈段引用的代码块的“热度”的统计度量。代码块的热度可以通过执行量来量化(例如,代码块的调用次数乘以代码块的执行时间)。较热的代码块具有较高的调用次数和/或较长的响应时间。

[0032] 通过以规则或不规则的时间间隔分析从进程中获取的一系列线程转储,一些实施例可以提供统计采样解决方案,该解决方案(1)是低开销的,(2)是非侵入性的,(3)提供始终在线的(always-on)监视,以及(4)避免工具化检测代码在被进行工具化检测的代码的执行时间中占据主导地位的问题(即,海森堡(Heisenberg)问题)。

[0033] 一些实施例可以基于强度统计信息对线程和栈段进行归类。通过监视从软件执行环境(例如,虚拟机)接收到的线程转储中包括的各个线程的栈踪迹,监视进程可以基于其栈踪迹的内容将线程归类为一个或多个线程类。随着更多的栈踪迹被分析,一些实施例可以观察到线程类到子类的分支并最终建立线程类的层次结构。例如,如果观察到栈段(A)是栈段(A,B,D)的组成部分,则可以说线程类型(A,B,D)是线程类型(A)的子类。也可以说线程类型(A,C)是线程类型(A)的子类。在如下的意义上,线程类型(A)包括子类(A,B,D)和(A,C),即,与(A,B,D)和(A,C)对应的强度统计信息的聚合可以通过与(A)对应的强度统计信息来表示。此外,一些实施例可以沿着线程类层次结构行进(例如,遍历树或图)以观察特定线程类的强度如何可以成比例地归因于该线程类的一个或多个子类的强度。例如,(A)的线程强度可以成比例地归因于(A,B,D)和(A,C)的线程强度。在其它实施例中,每个栈踪迹可以被表示为二叉树。

[0034] 一些实施例可以提供一个或多个顺序过滤器来估计度量、变化速率、加速度、季节因子和残差。可以通过这样的实施例来执行表示多个时段(例如,工作日时段和周末时段)的单独季节索引并且对这多个时段的季节因子进行归一化的技术。具体而言,一些实施例可以针对多个时段中的每个时段表示单独的季节索引序列。例如,多个时段可以包括工作

日时段、周末时段、季末时段或各个假日时段。在估计多个时段的季节索引时，一些实施例还可以(1)将季节索引重新归一化以在所有时段上提供公共的尺度和公共的参考水平，以及(2)跨相邻时段拟合平滑样条以提供在时段的周期之间或两个相邻时段的周期之间的平滑过渡。通过重新归一化，跨多个时段的季节因子可以具有公共的尺度。

[0035] 一些实施例可以将各种线程类的强度统计信息和堆使用量统计信息之间的趋势进行相关，以识别其强度统计信息与高的堆使用量具有高度相关性的线程类。在其强度统计信息与软件执行环境中的高的堆使用量高度相关的线程类当中，很有可能发现低效的堆存储器使用。一旦识别出这些线程类，就可以调查和/或优化与这些线程类相关联的代码。

[0036] 一些实施例可以构建和维护执行进程的多线程环境(例如，虚拟机)的模型(例如，单变量、多变量)，其中模型包括每个线程类的强度的季节性趋势、线性趋势和一阶非线性趋势。这些模型可以用于获得关于系统性能趋势的季节性调整的长期预测。

[0037] 通过(1)动态地对线程进行归类并观察线程类的子类的强度如何对线程类的聚合强度做出贡献，以及(2)观察各种线程类与检测到的高的堆使用量的时段相关的紧密程度，一些实施例可以促进检测和观察云服务供给系统内的性能故障。因为即使轻微的性能故障也常常会在进程中揭示可能导致SLA违规的问题，因此使服务提供商能够检测并解决性能故障可以极大降低此类违规的风险。

[0038] II.线程的运行时剖析

[0039] 图1-图2描绘了对正在运行的线程进行剖析以确定各个栈段在该线程的调用栈上相对于彼此存在多长时间的技术。图1描绘了以相对高频率的采样率在一段时间内对单个线程100的示例性运行时剖析。在一些情况下，某些技术可以利用运行时剖析器来获取线程的多个栈踪迹样本以构造图2中所示的调用上下文树200。如果运行时剖析器采用的采样间隔与线程的执行时间相比相对较短，那么对线程的每个调用上下文的观察计数(即，调用计数)统计信息可以被用于相对于采样间隔来准确地估计和/或表示调用上下文的执行时间。

[0040] 例如，如图1所示，线程100的总执行时间可以在100毫秒和1秒之间，而采样间隔在10毫秒和100毫秒之间。在线程的执行期间，取决于线程所调用的方法，线程的栈中可能存在不同的调用上下文。线程可以通过调用与栈段A对应的一组方法开始线程的执行。

[0041] 应该注意的是，栈段与线性连接的一组一个或多个栈帧对应。线性连接的栈帧在栈踪迹内始终被一起观察，因此具有相同的强度统计信息。因此，栈段A可以与多个栈帧对应，诸如栈帧a1、a2和a3。对线程进行采样可能在栈帧列表中产生栈踪迹，该栈踪迹描述被采样的线程的整个调用上下文。如果列出的一些栈帧是线性连接的，那么这些栈帧可以在概念上被分组为栈段。因此，栈踪迹可以包括一个或多个栈段，每个栈段包括一个或多个栈帧。

[0042] 随着线程继续其执行，与栈段A相关联的代码可能导致线程调用与栈段B对应的一组方法。接下来，与栈段B相关联的代码可能导致线程调用与栈段D对应的另一组方法。在短时间段之后，运行时剖析器可以获取线程100的样本1，从而产生第一栈踪迹。从第一栈踪迹中，运行时剖析器可以确定栈段A、B和D在采样时位于栈上。在采样间隔之后，运行时剖析器可以获取线程的另一个样本2，从而产生第二栈踪迹。从第二栈踪迹中，运行时剖析器可以确定栈段A、B和D位于栈上。随着线程继续执行，与栈段D相关联的方法可能返回，从而导致与栈段D对应的栈帧从栈弹出。接下来，运行时剖析器可以获取线程的另一个样本3，从而产

生第三栈踪迹。从第三栈踪迹中，运行时剖析器可以确定栈段A和B位于栈上。

[0043] 随着线程执行，栈段B调用栈段E，栈段E调用栈段F。接下来，获取样本4产生第四栈踪迹，其指示栈段A、B、E和F在栈上。栈段F、E和B一个接一个地返回。接下来，获取样本5产生第五栈踪迹，其指示仅栈段A在栈上。栈段A导致栈段C被推入到栈上。在栈段C返回之前，样本6和7被获取，从而产生第六栈踪迹和第七栈踪迹，两者都指示栈段A和C在栈上。最终，栈段C返回，从而只留下栈段A在栈上。当与栈段A相关联的方法返回时，该线程完成执行。

[0044] 如图2所示，调用上下文树200描绘了栈段A-F相对于采样间隔的执行时间。节点202指示在所有七个样本中观察到栈段A。节点204指示在七个样本中的四个中观察到栈段B。节点206指示在七个样本中的两个中观察到栈段C。节点208指示在七个样本中的两个中观察到栈段D。节点210指示在七个样本中的一个中观察到栈段E。节点212指示在七个样本中的一个中观察到栈段F。因为线程100的总执行时间大约是采样间隔的持续时间的十倍，因此每个栈段的观察计数可以与栈段的执行时间紧密相关。例如，因为栈段B被观察到了四次，因此可以推断出栈段B的相对执行时间是采样间隔的至少四倍。

[0045] 在一些情况下，线程100执行的环境(即，软件执行环境)可以对应于虚拟机(例如，热点Java虚拟机(JVM))，其中每个采样间隔进行一次线程转储。在虚拟机进行线程转储之前，它可以发信号通知所有正在执行的线程(例如，线程100)在安全点处暂停。该安全点机制可以类似于由垃圾收集器用来在执行完全垃圾回收之前暂停线程的机制。注意的是，在内核模式下运行的线程(例如，在I/O操作上运行/阻塞)可能在线程退出内核模式(例如，返回到JVM模式)之前不会在安全点处暂停。

[0046] 然而，应该注意的是，以高频率调用安全点机制可能导致大量开销。因此，依赖于高采样率的运行时剖析技术可能更适合于开发或测试环境而不是生产环境。

[0047] 为了减少开销，一些实施例采用系统模型来补偿降低的采样率。例如，一些实施例可以跟踪多线程进程的线程的强度，并且仅对强度超过阈值的线程进行采样，这些线程决定了延迟。采用降低的采样率或自适应采样率的实施例的一个优点是在内核模式下运行的线程不太可能在安全点处暂停。降低开销的其它方法可能涉及延长采样间隔以与被采样的线程的强度相称。例如，虽然一分钟的采样间隔可能导致生产环境内的开销可忽略不计，但是一分钟的采样间隔对于导出生产环境中线程的相对执行时间及其组成部分栈段可能足够短。因此，一些实施例可以为表现出平稳的平均遍历性或周期平稳的平均遍历性以满足Little公式的假设的生产系统提供始终在线的性能监视解决方案。在这样的实施例中，始终在线的性能监视解决方案可以被实施在监视进程(即，控制系统)中，该监视进程周期性地对在生产系统的一个或多个虚拟机内执行的线程进行采样。

[0048] III.对线程进行归类

[0049] 各种实施例提供用于顺序地分析从一个或多个虚拟机(例如，JVM)获取的一系列线程转储样本以识别线程类并跟踪与线程类有关的强度统计信息的技术。例如，在虚拟机内的一个或多个多线程进程的执行期间，控制系统可以周期性地获取虚拟机的线程转储。线程转储可以产生在虚拟机中执行的每个线程的栈踪迹。对于接收到的每个栈踪迹，控制系统可以分析栈踪迹中包含的文本以对相关联的线程进行归类并基于栈踪迹更新针对所有线程类跟踪的强度统计信息。

[0050] 除了对线程进行归类之外，实施例可以在每当新栈段沿着先前归类的栈段在分支

点处出现时对新栈段进行归类。当控制系统在发现任何线程类之前观察到第一栈踪迹时，控制系统可以认为该栈踪迹内的栈帧的整个序列是线性连接的，因为到目前为止栈帧的整个序列仅一起出现。作为响应，控制系统可以初始化线程类以对整个栈踪迹(即，栈帧的整个序列)进行归类。当控制系统观察到包括不同栈帧序列的后续栈踪迹时，控制系统可以初始化附加线程类以对栈帧的每个唯一排列进行归类。在一些情况下，控制系统可能观察到不与先前观察到的栈踪迹共享任何栈帧(即，具有任何共同的栈帧)的栈踪迹。作为响应，控制系统可以初始化单独的线程类以对新的栈踪迹整体进行归类。

[0051] 然而，更常见的是，控制系统可以观察到与先前观察到的栈踪迹共享一个或多个栈帧的栈踪迹。回到图1，例如，假设控制系统观察到的第一个栈踪迹是{(A,B,D)}(即，样本1或样本2中的栈踪迹)，其中栈踪迹包含栈段A、B和D中包含的栈帧。控制系统可以初始化线程类{(A,B,D)}以对观察到的包括栈段A、B和D中所包含的栈帧的所有线程进行归类。接下来，假设控制系统观察到的第二个栈踪迹是{(A,C)}(即，样本6或样本7中的栈踪迹)。在这点上，控制系统可以确定虽然第一和第二栈踪迹不同，但是第一和第二栈踪迹共享栈段A中包含的所有栈帧，这导致栈段A处的分支点。作为响应，控制系统可以初始化线程类{(A,C})以对在其调用栈上包含栈段A和C的所有线程进行归类。

[0052] 应该注意的是，因为栈段A中的栈帧已经被观察到与栈段(B,D)中的栈帧分离，因此栈段A和(B,D)不再被控制器系统认为是线性连接的。然而，控制系统仍然认为栈段A中的栈帧是线性连接的，并且栈段(B,D)中的栈帧是线性连接的。在这点上，控制系统可以初始化线程类{(A,B,D)}和线程类{(A,C)}的若干线程段组成部分，以对由新发现的分支点形成的新栈段进行归类。具体而言，控制系统可以初始化线程段(A)、线程段(B,D)和线程段(C)，其中线程段(A)和(B,D)是线程类{(A,B,D})的组成部分并且线程段(A)和(C)是线程类{(A,C)}的组成部分。

[0053] 一些实施例可以使用归类签名来表示栈踪迹和栈段。具体而言，踪迹签名可以用于表示特定线程类的栈踪迹，并且段签名可以用于表示特定线程段的栈段。每个踪迹签名可以与经由合成和分析进程建立的标记二叉树的元组对应。同时，线程段的每个段签名可以与元组中的节点对应，该节点与线程段是其组成部分的线程类对应。稍后在分析进程中，元组可以像解析树一样使用(例如，作为产生式文法(production grammar)的一部分)，以辨别传入的栈踪迹。

[0054] 回到上面的示例，在观察第一栈踪迹之后但在观察第二栈踪迹之前，线程类{(A,B,D})可以与单个二叉树的元组对应。因为第一栈踪迹内的整个帧序列被认为是单个栈段，因此单个二叉树可以包括表示栈段(A,B,D)的单个根节点。在观察第二栈踪迹之后，元组可能仍然仅包括单个二叉树。然而，二叉树现在可以包括三个单独的节点:表示栈段(A,B,D)的根节点、根节点的表示栈段(A)的第一子节点，以及根节点的表示栈段(B,D)的第二子节点。下面参考图4-图6进一步详细讨论合成踪迹签名和段签名的过程。

[0055] 二叉树中的每个节点可以由标签或标识符唯一地识别，该标签或标识符可以被称为紧凑代码(compact code)。在一些实施例中，特定线程类的线程可以由一个或多个紧凑代码表示，该一个或多个紧凑代码识别与线程类对应的每个排名靠前的元组节点。以类似于霍夫曼编码或其它熵编码方案的方式，一些实施例可以将较短的元组与更流行(即，具有较高线程强度)和/或被首先发现的线程类相关联。因此，更常见类型的线程可以由更短的

紧凑代码序列紧凑地表示。在一些实施例中，可以通过首先在离线分析(即，离线处理)中分析栈踪迹的概率分布并且以频率的降序将栈踪迹馈送到控制系统来确保这一点。

[0056] 在不依赖于离线分析的实施例中，控制系统可以以从一个或多个虚拟机周期性地获取的线程转储的顺序来接收栈踪迹(即，在线处理)。

[0057] 观察到不同类型的栈踪迹的次序可能受到每种类型的栈踪迹的强度的影响。换句话说，具有更高强度的栈踪迹在统计上更可能在序列中更早地被观察到。因此，这样的实施例可以假设(1)特定线程类的线程强度表示相关联的栈踪迹的发生概率，以及(2)在与较低强度线程类相关联的栈踪迹之前经常观察到与较高强度线程类相关联的栈踪迹。在这点上，控制系统将自然地为最高强度线程导出最紧凑的表示。因此，通过依赖于线程强度统计信息而不是离线处理，一些实施例可以为响应于一系列线程转储而观察到的栈踪迹提供最佳压缩算法。

[0058] A.线程强度的季节性

[0059] 对于识别出的每个线程类，一些实施例可以估计线程类的强度的季节性趋势。如上所述，线程类或线程段的强度可以指由相关联的栈踪迹或栈段引用的代码块的“热度”的统计度量。代码块的热度可以通过代码块的调用次数乘以代码块的执行时间来量化。线程类的单个原始线程强度度量可以是该线程类在特定线程转储中的线程数量的计数。每个线程转储的平均线程强度度量可以与线程类型的流量强度、供应的负载或队列长度对应。对于平均遍历性进程，Little公式可以将预期强度$\hat{\rho}$(在与预期响应时间$\hat{\tau}$对应的采样间隔期间的预期到达数量)与预期响应时间$\hat{\tau}$和到达率λ相关联，如下所示：

[0060] $$\hat{\rho} = \lambda \cdot \hat{\tau}$$

[0061] 在一些实施例中，季节性趋势分析进程可以使用可变过滤器参数来考虑不规则的采样间隔(例如，对堆使用量进行采样和/或获取线程转储)并且克服柯西(Cauchy)分布问题。该进程还可以支持以不同长度(例如，1天、2天)顺序地过滤多种类型的时段(例如，工作日时段、周末时段和假日时段)。此外，该进程可以根据季节性调整进行线程转储的速率以减少开销，同时维持基于线程转储确定的线程强度统计信息的特定置信水平。在一些情况下，调整线程转储速率还可以最小化需要通过网络(例如，LAN、互联网)传输到其它机器(例如，大数据存储库)以进行离线处理的线程转储数据的体量。

[0062] 在一些实施例中，季节性趋势分析进程可以将工作日时段(即，24小时时段)划分为96个十五分钟的间隔，这为每个工作日时段产生96个季节索引(即，季节)。该进程可以将周末时段(即，48小时时段)划分为192个15分钟的间隔，这为每个周末时段产生192个季节索引。在接收到特定长度的数据集(例如，记录其中包括一个或两个周末的10天内的线程转储或堆使用量的时间系列)时，该进程可以将多时段趋势分析过滤器单独应用于工作日时段和周末时段，以便分离出在单个工作日观察到的季节模式和在整个周末观察到的季节模式，从而产生每个工作日的96个季节索引的一组96个季节因子和每个周末的192个季节索引的一组192个季节因子。然后，该进程可以将工作日季节因子和周末季节因子重新归一化，使得季节因子“1”表示工作日时段和周末时段的公共参考水平。

[0063] 应该注意的是，如果将大于1的季节因子指派给季节索引，则该季节索引与该时段的其余部分相比具有高于平均值的值。另一方面，如果将小于1的季节因子指派给季节索

引,则与该时段的其余部分相比,该季节索引具有低于平均值的值。例如,如果与9AM-9:15间隔对应的季节索引的特定线程类的线程强度的季节因子是1.3,则该特定线程类在9AM-9:15AM间隔期间的平均线程强度间隔比该特定线程类在整个工作日的平均线程强度高30%。

[0064] 在一些实施例中,季节性趋势分析进程可以将假日(例如,劳动节、圣诞节)分离出来作为以每12个月一次的频率重复的单独时段,而工作日时段以每24小时进行重复并且周末时段以每5天或7天进行重复。这样的假日时段的一组季节因子可以与工作日时段和周末时段的那些季节因子一起进行重新归一化,使得季节因子1表示所有时段的公共参考水平。根据需要,每个时段的其它频率也可能是适当的。作为示例,假日可以以每6个月(等等)的频率分开,而工作日可以是每12小时(等等)重复的时段。

[0065] 在一些实施例中,确定和跟踪强度统计信息还可以包括预测未来值和变化速率。然而,采样间隔可以是不规则的,甚至可以变得任意接近于零。在采样间隔变得任意接近于零的情况下,变化速率可能成为柯西分布的随机变量,其均值和标准差是不确定的。为了克服关于利用自适应采样间隔确定季节性趋势的柯西分布问题,一些实施例可以采用各种适配,如Holt的双指数过滤器、Winter的三指数过滤器、Wright的不规则时间间隔的扩展、Hanzak的用于时间相近间隔的调整因子、异常值检测,以及具有异常值截止的自适应缩放的削波。可以将这五组指数过滤器顺序地应用于数据集,以估计工作日时段和周末时段的季节因子的集合。

[0066] B.归类签名和压缩方案

[0067] 某些实施例可以向线程的栈踪迹指派可变长度的紧凑代码序列,其中序列的长度取决于线程的强度。示例性栈踪迹如下表示:

[0068] oracle.jdbc.driver.T4CCallableStatement.executeForRows(T4CCallableStatementjava:991)

[0069] oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1285)

[0070] …

[0071] oracle.mds.core.MetadataObject.getBaseMO(MetadataObject.java:1048)

[0072] oracle.mds.core.MDSSession.getBaseMO(MDSSession.java:2769)

[0073] oracle.mds.core.MDSSession.getMetadataObject(MDSSession.java:1188)

[0074] …

[0075] oracle.adf.model.servlet.ADFBindingFilter.doFilter(ADFBindingFilter.java:150)

[0076] …

[0077] oracle.apps.setup.taskListManager.ui.customization.CustomizationFilter.doFilter(CustomizationFilter.iava:46)

[0078] …

[0079] weblogic.servlet.internal.WebAppServletContext.securedExecute(WebAppServletContext.jana:2209)

[0080] weblogic.servlet.internal.ServletRequestImpl.run

(ServletRequestImpl.java:1457)

[0081] …

[0082] weblogic.work.ExecuteThread.execute(Execute Thread.java:250)

[0083] weblogic.work.ExecuteThread.rum(ExecuteThread.java:213)

[0084] 在示例性栈踪迹中,Java数据库连接(JDBC)驱动程序栈段(即,各自包括“oracle.jdbc.driver...并的两个栈帧)下面的栈帧“oracle mds core MetadataObject getBaseMO并指示元数据服务(MDS)库调用与该JDBC栈段对应的JDBC操作。MDS库栈段(即,各自包括“oracle.mds...并的三个栈帧)下面的栈帧“oracle adf model servlet ADFBindingFilter doFilter”指示应用开发框架(ADF)操作调用MDS操作。如栈踪迹底部的WebLogic栈段(即,各自包括“weblogic…”的四个栈帧)所示,ADF操作通过超文本传输协议(HTTP)Servlet请求来调用。

[0085] 作为示例,可以使用两级霍夫曼编码方案来编码和压缩上述栈踪迹,从而产生表示示例性栈踪迹的紧凑代码序列。在第一级,压缩工具(例如,gzip)可以检测栈踪迹中的子串,诸如“ServletRequestImpl.java”和“weblogic.servlet.internal.ServletRequestImpl.run并,并根据这些子串出现在栈踪迹中的频率来导出用于这些子串的霍夫曼编码。为了增加压缩比,可以为较频繁出现的子串指派较短的霍夫曼编码。在第一级压缩之后,压缩后的栈踪迹可以包括作为元数据的编码字典,该编码字典可以用于根据霍夫曼编码恢复子串。

[0086] 第二级可以涉及通过用段签名替换栈踪迹的栈段来将另一级压缩应用于压缩后的栈踪迹。下面关于图4-6更详细地讨论应用第二级压缩的步骤。

[0087] C.示例性数据结构

[0088] 可以经由一个或多个对象类型在存储器中表示归类签名。具体而言,一些实施例可以使用ThreadClassificationInfo对象来表示线程类的归类签名(即,踪迹签名)、使用SegmentInfo对象来表示线程段的归类签名(即,段签名)、使用StackFrameInfo对象来表示栈段内线性连接的栈帧中的每个元素,以及使用SeasonalTrendInfo对象来封装和跟踪线程类或线程段的强度统计信息。

[0089] 下面提供了定义ThreadClassificationInfo对象、SegmentInfo对象、StackFrameInfo对象和SeasonalTrendInfo对象的示例性类/接口定义:

```
public class ThreadClassificationInfo {
        long id;
        String name;
        short numOfOccur;
        short totalNumOfOccur;
        short numOfStackFrames;
        short numOfCoalescedSegments;
        List<SegmentInfo>segments;
        SeasonalTrendInfo trend;
}

public class SegmentInfo extends SegmentInfo {
```

[0090]

```
        long id;
        String name;
        String dimension;
        short numOfOccur;
        short totalNumOfOccur;
        List<StackFrameInfo>elements;
        SegmentInfo firstSegment;
        SegmentInfo secondSegment;
        StackSegmentInfo coalescingSegment;
        Set<StackSegmentInfo>predecessors;
        Set<StackSegmentInfo>successors;
        SeasonalTrendInfo trend;
        Set<ThreadClassInfo> partOfThreadClasses;
```

```
}

public class StackFrameInfo {
        long id;
        String name;
        short numOfOccur;
        short totalNumOfOccur;
        Set<StackFrameInfo>predecessors;
        Set<StackFrameInfo>successors;
        StackSegmentInfo coalescingSegment;
        String classMethodLineNumber;
}

public class SeasonalTrendInfo {
        List<long>posixTimestampOfMeasurement;

        List<short>rawMeasure;
        List<double>rawDeseasonalizedMeasure;
        List<double>smoothedMeasure;
        List<double>smoothedDeseasonalizedMeasure;
        double measureFilterConstant;
        List<double>measureWeightFactor;
        List<double>measureFilterParameter;

        List<double>rawGrowthRate;
        List<double>smoothedGrowthRate;
        double rateFilterConstant;
        List<double>rateWeightFactor;
        List<double>rateFilterParameter;

        List<double>rawGrowthRateAcceleration;
        List<double>smoothedGrowthRateAcceleration;
        double accelerationFilterConstant;
        List<double>accelerationWeightFactor;
        List<double>accelerationFilterParameter;

        List<double>rawWeekdaySeasonalFactor;
        List<double>rawWeekendSeasonalFactor;
```

[0091]

[0092]

```
List<double>smoothedWeekdaySeasonalFactor;
List<double>smoothedWeekendSeasonalFactor;
double seasonalFactorFilterConstant;
List<double>seasonalIndexWeightFactor;
List<double>seasonalIndexFilterParameter;

List<double>errorResidual;
List<double>smoothedErrorResidual;
List<double>smoothedAbsoluteErrorResidual;
List<double>normalizedResidual;
List<double>normalizedResidualCutoff;
double errorResidualFilterConstant;
List<double>errorResidualWeightFactor;
List<double>errorResidualFilterParameter;

List<double>localGrowthRateForecast;
List<double>oneStepIntensityForecast;
List<double>multiStepIntensityForecast;
short                    forecastHorizon;

double[96] weekdaySeasonalFactor;
double[192] weekendSeasonalFactor;

}
```

[0093] 从以上定义中可以看出,每个ThreadClassificationInfo对象、SegmentInfo对象和StackFrameInfo对象包括唯一标识符(即,id)、名称、跟踪相同类型的对象(例如,相同的线程类、相同的线程段、相同类型的栈帧)在最新的线程转储中被观察到的次数的计数器(即,numOfOccur),以及跟踪相同类型的对象在所有线程转储中被观察到的次数的另一个计数器。

[0094] ThreadClassificationInfo对象可以包括SegmentInfo对象的列表和SeasonalTrendInfo对象。在这点上,ThreadClassificationInfo可以与二叉树的元组对应,而SegmentInfo对象的列表与构成二叉树的节点对应。SeasonalTrendInfo对象可以记录与由ThreadClassificationInfo对象表示的线程类有关的强度统计信息(例如,过滤器状态)。

[0095] SegmentInfo对象可以包括StackFrameInfo对象的列表、第一子SegmentInfo对象(即,firstSegment)、第二子SegmentInfo对象(即,secondSegment)、合并(即,父)SegmentInfo对象(即,coalescingSegment)、前面同级SegmentInfo对象(即,前驱)的列表、后继同级SegmentInfo对象(即,后继者)的列表,以及SeasonalTrendInfo对象。在这点上,SegmentInfo对象可以与栈段对应。如果SegmentInfo对象与叶节点对应,则

StackFrameInfo对象的列表可以与栈段中包括的线性连接的栈帧对应。如果SegmentInfo对象与分支点相邻，则同级SegmentInfo对象可以与分支点的相对侧上的栈段对应，而合并SegmentInfo对象可以与包括栈段和同级栈段的父栈段对应。如果SegmentInfo对象不与叶节点对应，则子SegmentInfo对象可以与在栈段中发现分支点时创建的栈段的子段对应。SeasonalTrendInfo对象可以记录与由SegmentInfo对象表示的线程段有关的强度统计信息。

[0096] 一些实施例可以通过将与单个SegmentInfo节点一起观察到的StackFrameInfo对象的列表相关联来对栈踪迹的栈段进行归类。换句话说，SegmentInfo节点是栈段的每个StackFrameInfo对象的合并节点。每个StackFrameInfo对象可以具有单个合并的SegmentInfo节点。当沿着SegmentInfo节点的线性连接的StackFrameInfo对象的某处检测到分支点时，一些实施例可以创建两个新的SegmentInfo节点并将线性连接的StackFrameInfo对象分割成新的SegmentInfo节点当中的两组线性连接的StackFrameInfo对象。然后，它可以通过分支点重新连接两个StackFrameInfo对象。

[0097] 新SegmentInfo节点中的每一个都成为该节点的部分段中的StackFrameInfo对象的合并节点。某些实施例可以相应地更新StackFrameInfo对象的coalescingSegment，使得每个StackFrameInfo对象引用正确的合并SegmentInfo节点。两个新的SegmentInfo节点被表示为左同级节点和右同级节点。这两个新的SegmentInfo节点也成为原始SegmentInfo节点的子节点，而原始SegmentInfo节点又成为它们的父节点。父SegmentInfo节点可以成为这两个新SegmentInfo节点的合并节点。

[0098] 响应于发现的分支点而分割栈段的进程可以导致由SegmentInfo节点组成的二叉树结构。该分割进程可以被看作是线程类(即，栈踪迹类)分叉成线程子类。随着栈段中的各个栈帧的强度随时间而偏离，一些实施例可以连续地将栈段分割成更小的栈段，从而使得能够下挖线程类层次结构以观察线程类的强度如何可以被按比例归因于线程子类的强度。

[0099] 在一些实施例中，二叉树内部的SegmentInfo节点是父节点，其StackFrameInfo对象并非全部是线性连接的，因为一些栈帧通过分支点连接。作为对照，叶SegmentInfo节点的StackFrameInfo对象可以是线性连接的。在SegmentInfo节点内，线性连接或分支点连接的StackFrameInfo对象可以被定向为具有底部StackFrameInfo和顶部StackFrameInfo的栈。按照惯例，左同级SegmentInfo节点中的顶部StackFrameInfo对象可以通过分支点连接到右同级SegmentInfo节点的底部StackFrameInfo对象。

[0100] 每个SegmentInfo节点可以包括SeasonalTrendInfo对象，以跟踪由该SegmentInfo节点表示的线程(子)类的强度统计信息。当将SegmentInfo节点分割成两个新的子SegmentInfo节点时，一些实施例可以将SegmentInfo节点的SeasonalTrendInfo对象克隆到两个新的SeasonalTrendInfo对象中，并在每个子SegmentInfo节点中设置一个SeasonalTrendInfo对象。

[0101] 一些实施例提供了通过分割进程将父SegmentInfo节点的过滤器状态复制到新的子SegmentInfo节点的能力。在这样做时，一些实施例可以连续跟踪强度统计信息在该父节点和同级SegmentInfo节点之间的比率。具体而言，子SegmentInfo节点的强度统计信息初始地每个都与父SegmentInfo节点的强度统计信息相同。然而，当获得新样本时，子SegmentInfo节点的强度统计信息可能开始偏离父节点和彼此的强度统计信息。随着新栈

段的过滤器状态被分别更新，新栈段的过滤器状态开始彼此发生偏差并且与原始栈段的过滤器状态发生偏差。

[0102] 在一些情况下，父SegmentInfo节点和同级SegmentInfo节点之间的强度统计信息可以随时间收敛到某个比率。一些实施例可以应用SegmentInfo节点之间的父子关系和同级关系来定义多变量状态估计技术的相关模型。具体而言，如果进程是平稳的，则相关的SegmentInfo节点之间的强度统计信息的比率可以收敛到平稳状态。具体而言，如果进程是严格意义上或广义上平稳的，那么第一和第二时刻的强度统计信息在相关的SegmentInfo节点之间的联合概率分布(其可以包括相关的SegmentInfo节点的均值、方差、自协方差和互协方差)可能不会相对于时间而变化。因此，可以预期强度统计信息在父SegmentInfo节点和同级SegmentInfo节点之间的比率随时间收敛。因此，通过经由分支点连续跟踪同级SegmentInfo节点的强度统计信息并且确定父SegmentInfo节点和同级SegmentInfo节点之间的强度统计信息的比率随时间收敛，一些实施例可以使用这些比率来定义用于多变量状态估计技术的相关模型。结果得到的模型可以用于异常检测和生成预测。

[0103] StackFrameInfo对象可以包括一个或多个前驱StackFrameInfo对象和/或一个或多个后继StackFrameInfo对象(即，前驱和后继者)、合并SegmentInfo对象(即，coalescingSegment)，以及识别由StackFrameInfo对象引用的代码的信息(即classMethodLineNumber)。如果StackFrameInfo对象不与分支点相邻，则StackFrameInfo对象可以线性连接到单个前驱栈帧和单个后继者栈帧。StackFrameInfo对象可以通过成员变量coalescingSegment来引用进行包含的SegmentInfo对象。

[0104] 当该处理最新的线程转储时，每个ThreadClassificationInfo对象、SegmentInfo对象和StackFrameInfo对象的成员变量numOfOccur可以被重置为0。从线程转储获得的每个栈踪迹可以从栈踪迹的底部向顶部进行解析。在应用第一级别的霍夫曼编码方案以压缩栈踪迹之后，可以将栈踪迹的每一行解析为StackFrameInfo对象。在将StackFrameInfo对象的列表解析为SegmentInfo对象的列表之后，一些实施例可以尝试将SegmentInfo对象的列表匹配到包含SegmentInfo对象的匹配列表的ThreadClassificationInfo对象。如果不存在这样的ThreadClassificationInfo对象，则一些实施例可以注册新的ThreadClassificationInfo对象以表示SegmentInfo对象的列表。之后，一些实施例然后可以更新匹配的/新的ThreadClassificationInfo对象以及匹配的/新的ThreadClassificationInfo对象中的每个SegmentInfo对象和StackFrameInfo对象的numOfOccur和totalNumOfOccur成员变量。注意的是，如果SegmentInfo节点是叶级节点，则该节点的numOfOccur成员变量将等于该SegmentInfo节点中每个StackFrameInfo元素的numOfOccur成员变量。

[0105] 接下来，一些实施例可以更新封装在相关联的SeasonalTrendInfo对象中的强度统计度量。具体而言，一些实施例可以通过将rawMeasure设置为进行包含的ThreadClassificationInfo对象或SegmentInfo对象的numOfOccur成员变量来更新每个SeasonalTrendInfo对象中的rawMeasure成员变量。注意的是，在一些实施例中，rawMeasure可以仅在每N个线程转储时更新，在这种情况下，SeasonalTrendInfo对象的rawMeasure被设置为对应的numOfOccur除以N。在一些实施例中，这样的实施例可以仅当相关联的ThreadClassificationInfo对象或相关联的SegmentInfo对象的numOfOccur成员变

量不为零时才更新SeasonalTrendInfo对象的rawMeasure成员变量。如果numOfOccur成员变量不为零,则SeasonalTrendInfo对象的rawMeasure被设置为numOfOccur的值除以N,其中N是自上次更新rawMeasure以来线程转储的数量。在这样的实施例中,该方法将numOfOccur为零时的情况处理为就好像没有可用的测量值一样。在这点上,当没有可用的测量时,rawMeasure不会被更新。换句话说,这些实施例跟踪自上次更新rawMeasure以来线程转储的数量'N'。线程强度测量值可以与不规则的时间系列对应。应该注意的是,不规则时间间隔的指数过滤器(例如,上面公开的Holt的双指数过滤器和Winter的三指数过滤器)可以有效地过滤rawMeasure从而从以不规则的时间间隔获取的一组测量值中获得去季节化的度量和季节因子。

[0106] 应该注意的是,每个SeasonalTrendInfo对象可以包括由被应用于以下每个统计测量值的五组指数过滤器生成的时间系列数据:线程强度的原始度量、线程强度正在增加或减小的速率、该速率的加速度或减速度、线程强度的季节因子、以及残余分量。在SeasonalTrendInfo对象内,用于变量的五组指数过滤器的状态、过滤器常数、(用于调整样本之间的不规则时间间隔的)过滤器参数调整权重因子、以及过滤器参数可以由该时间系列数据来表示。

[0107] D.示例性归类签名的生成

[0108] 图3描绘了根据一些实施例的虚拟机300在一段时间内的示例性线程转储。与图1中的100ms至1秒采样间隔的运行时剖析相比,由图3中的控制系统采用的采样间隔可以更长(例如,在20秒和1分钟之间)以减少采样开销。如图3所示,在两到三个采样间隔内,在虚拟机300内执行的进程产生线程302、304、306、308、310和312。线程302-312中的每一个在执行时与单独的调用栈相关联,并且因此可以在获取线程转储时产生栈踪迹。图3描绘了总共三个线程转储被获取:线程转储N、线程转储N+1和线程转储N+2。

[0109] 图3示出了在三次连续的线程转储中按顺序(A,B,D)、(A,B,D)、(A,C)和(A,B,E)观察到的三种不同类型的栈踪迹。栈踪迹(A,B,D)被观察到两次。在获取线程转储N之前,线程302被产生并开始执行。当获取线程转储N时,观察到线程302的栈踪迹(A,B,D)。应该注意的是,即使栈段A、栈段B和栈段D尚未被识别出,但是为了方便起见,在图3中描绘的整个示例中,将使用这些栈段的名称。随着在获取线程转储N之后经过一个采样间隔,线程302结束,线程304被产生并且线程304在从未在线程306和308被产生的同时被采样的情况下结束。当获取线程转储N+1时,线程308产生栈踪迹(A,B,D),而线程310产生栈踪迹(A,C)。随着在获取线程转储N+1之后经过另一个采样间隔,线程306和308结束,线程310被产生且在从未被采样的情况下结束,并且线程312被产生。当获取线程转储N+2时,线程312产生栈踪迹(A,B,E)。如从图3中可以看到的,(A,B,D)线程类型是将被观察到的第一种线程类型,并且(A,B,D)线程类型具有高于(A,C)或(A,B,E)线程类型的强度。

[0110] 在线程转储N之后,控制系统可以将单个SegmentInfo(A,B,D)节点注册为栈踪迹(A,B,D)的归类签名。然后,控制系统可以将SeasonalTrendInfo(A,B,D)对象与SegmentInfo(A,B,D)节点相关联,并更新由该节点封装的状态:

[0111] SegmentInfo(A,B,D).numOfOccur=1.

[0112] SegmentInfo(A,B,D).totalNumOfOccur=1.

[0113] 图4描绘了一组归类签名400,其包括响应于栈踪迹(A,B,D)而已注册的单个归类

签名450。如图4中可以看到的，归类签名450包括与SegmentInfo(A,B,D)对应的单个节点402，其中SegmentInfo(A,B,D)被示出为是栈踪迹的所有栈帧a1-d3的合并节点。

[0114] 当在线程转储N+1中再次观察到栈踪迹(A,B,D)时，控制系统可以如下更新SegmentInfo(A,B,D)节点：

[0115] SegmentInfo(A,B,D).numOfOccur＝1.

[0116] SegmentInfo(A,B,D).totalNumOfOccur＝2.

[0117] 当在线程转储N+1中第一次观察到栈踪迹(A,C)时，控制系统确定栈段(A,B,D)内的整个栈帧集合不再是线性连接的。现在，在栈帧集合的由'A'表示的(例如，从栈踪迹的顶部到底部)最后一个栈帧和该栈帧集合的由'B,D'表示的第一栈帧之间存在分支点，因为，在任何给定的栈踪迹中，最后一个栈帧之后的下一个栈帧可以是(1)(B,D)的第一个栈帧，或者(2)栈帧集合中的由'C'表示的第一个栈帧。因此，控制系统可以通过创建节点SegmentInfo(A)和SegmentInfo(B,D)并将这两个节点指派为SegmentInfo(A,B,D)的子节点来将栈段(A,B,D)分割成栈段(A)和栈段(B,D)。对于栈踪迹(A,C)，控制系统可以通过创建节点SegmentInfo(C)来初始化栈段(C)并且注册包括SegmentInfo(A)和SegmentInfo(C)的有序元组作为栈踪迹(A,C)的归类签名。

[0118] 在一些实施例中，控制系统可以将SeasonalTrendInfo(A,B,D)对象分别克隆到节点SegmentInfo(A)和SegmentInfo(B,D)的SeasonalTrendInfo(A)和SeasonalTrendInfo(B,D)对象中，并且为SegmentInfo(C)创建新的SeasonalTrendInfo(C)，如下所示：

[0119] SeasonalTrendInfo(A)←SeasonalTrendInfo(A,B,D)

[0120] SeasonalTrendInfo(B,D)←SeasonalTrendInfo(A,B,D)

[0121] SeasonalTrendInfo(C)←new SeasonalTrendInfo

[0122] 控制系统还可以如下更新上述SegmentInfo节点：

[0123] SegmentInfo(A).numOfOccur＝2

[0124] SegmentInfo(A).totalNumOfOccur＝3

[0125] SegmentInfo(C).numOfOccur＝1

[0126] SegmentInfo(C).totalNumOfOccur＝1

[0127] 图5描绘了一组归类签名500，其包括归类签名450和响应于第一次观察到栈踪迹(A,C)而生成的新归类签名550。如从图5中可以看到的，归类签名450现在包括三个节点：节点402、节点502和节点504。节点402与SegmentInfo(A,B,D)对应，它是节点502和节点的合并节点。节点502与SegmentInfo(A)对应，其合并栈帧a1-a3。节点504与SegmentInfo(B,D)对应，其合并栈帧b1-d3。归类签名550包括两个节点：与被示出为合并栈帧a1-a3的SegmentInfo(A)对应的节点506，以及与被示出为合并栈帧c1-c3的SegmentInfo(C)对应的节点508。

[0128] 当在线程转储N+2中第一次观察到栈踪迹(A,B,E)时，控制系统确定栈段(B,D)内的整个栈帧集合不再是线性连接的。现在，在栈帧集合的由'B'表示的最后一个栈帧和栈帧集合的由'D'表示的第一个栈帧之间存在分支点，因为，在任何给定的栈踪迹中，最后一个栈帧之后的下一个栈帧可以是(1)(D)的第一个栈帧，或者(2)栈帧集合的由'E'表示的第一个栈帧。因此，控制系统可以通过创建节点SegmentInfo(B)和SegmentInfo(D)并将这两个节点指派为SegmentInfo(B,D)的子节点来将栈段(B,D)分割成栈段(B)和栈段(D)。对于栈

踪迹(A,B,E),控制系统可以通过创建节点SegmentInfo(E)来初始化栈段'E'并且注册包括SegmentInfo(A),SegmentInfo(B)和SegmentInfo(E)的有序元组作为栈踪迹(A,B,E)的归类签名。

[0129] 在一些实施例中,控制系统可以将SeasonalTrendInfo(B,D)对象分别克隆到节点SegmentInfo(B)和SegmentInfo(D)的SeasonalTrendInfo(B)和SeasonalTrendInfo(D)对象中,并为SegmentInfo(E)创建新的SeasonalTrendInfo(E),如下:

[0130] SeasonalTrendInfo(B)←SeasonalTrendInfo(B,D)

[0131] SeasonalTrendInfo(D)←SeasonalTrendInfo(B,D)

[0132] SeasonalTrendInfo(E)←new SeasonalTrendInfo

[0133] 控制系统还可以如下更新上述SegmentInfo节点:

[0134] SegmentInfo(A).numOfOccur=1

[0135] SegmentInfo(A).totalNumOfOccur=4

[0136] SegmentInfo(B).numOfOccur=1

[0137] SegmentInfo(B).totalNumOfOccur=3

[0138] SegmentInfo(E).numOfOccur=1

[0139] SegmentInfo(E).totalNumOfOccur=1

[0140] 图6描绘了一组归类签名600,其包括归类签名450和550以及响应于栈踪迹(A,B,E)而生成的新归类签名650。如图6中可以看到的,归类签名450现在包括五个节点:节点402、节点502、节点504、节点602和节点604。节点504与SegmentInfo(B,D)对应,它是节点602和节点604的合并节点。节点602与SegmentInfo(B)对应,它合并栈帧b1-b3。节点604与SegmentInfo(D)对应,它是栈帧d1-d3的合并节点。归类签名550没有改变。归类签名650包括三个节点:与被示出为合并栈帧a1-a3的SegmentInfo(A)对应的节点606、与被示出为合并栈帧b1-b3的SegmentInfo(B)对应的节点608,以及与被示出为合并栈帧e1-e3的SegmentInfo(E)对应的节点610。

[0141] 如图6所示,栈踪迹(A,B,D)的归类签名可以由归类签名450的根处的单个SegmentInfo节点组成。换句话说,作为最高强度栈踪迹的栈踪迹(A,B,D)具有最紧凑的表示。同时,栈踪迹(A,C)被指派具有两个有序节点(A)和(C)的第二短的归类签名。最后检测到的栈踪迹(A,B,E)被指派具有三个有序节点(A)、(B)和(E)的第三短的归类签名。如图4-图6所示,ThreadClassificationInfo对象可以与SegmentInfo节点的元组对应,并且SegmentInfo节点可以引用其它SegmentInfo节点和/或StackFrameInfo对象集合的二叉树(或二进制的子树)。ThreadClassificationInfo对象、SegmentInfo节点和StackFrameInfo对象可以一起构成产生式文法:

[0142] Thread1->(A,B,D)

[0143] Thread2->(A)(C)

[0144] Thread3->(A)(B)(E)

[0145] (A,B,D)->(A)(B,D)

[0146] (B,D)->(B)(D)

[0147] A->a1,a2,a3

[0148] B->b1,b2,b3

[0149] C->c1,c2,c3

[0150] D->d1,d2,d3

[0151] E->e1,e2,e3

[0152] 如上所述,各个栈帧ai,bi,ci,di,ei是终端,而SegmentInfo节点是文法的非终端。一些实施例可以从栈踪迹的底部向栈踪迹的顶部(在以下表示中定向为从左到右)来解析栈踪迹的栈帧。

**a1,a2,a3,b1,b2,b3,d1,d2,d3**

**(A),b1,b2,b3,d1,d2,d3 use production (A) -> a1,a2,a3**

**(A),(B),d1,d2,d3 use production (B) -> b1,b2,b3**

[0153] **(A),(B),(D) use production (D) -> d1,d2,d3**

**(A),(B,D) use production (B,D) -> (B)(D)**

**(A,B,D) use production (A,B,D) -> (A),(B,D)**

**Thread1 use production Thread1 -> (A,B,D)**

[0154] 如以上可以看到的,一些实施例可以经由自下而上的语法分析来分析栈帧,其可以类似于移位减少解析(shift-reduce parsing)或从左到右的"LR"解析。该分析可以涉及移位和减少栈帧和SegmentInfo节点,以通过从树的叶子到根进行操作来为栈踪迹构造解析树。一些实施例可以针对线程的栈踪迹的较早的出现(occurrence)来合成解析树,并通过减少(即,移位减少解析、从左到右的"LR"解析)来对同一解析树分析线程的另一出现的栈踪迹。归类树的每个节点可以是用于栈踪迹类的紧凑标签,并且归类树的根可以是用于线程类的紧凑标签。

[0155] 图7图示了根据一些实施例的用于响应于线程转储而生成和/或修改一个或多个线程归类签名的过程的流程图700。在一些实施例中,流程图700中描绘的过程可以由具有一个或多个处理器(例如,图17的计算机系统1700)的计算机系统来实现,其中该一个或多个处理器可以基于存储在计算机可读介质中的计算机代码来执行这些步骤。图7中描述的步骤可以以任何顺序执行,并且可以有或没有任何其它步骤。

[0156] 流程图700开始于步骤702,其中实施例在多线程程序的执行期间执行线程转储。具体而言,一些实施例可以与监视多线程程序在其中执行的软件执行环境的一个或多个监视进程对应。该软件执行环境可以支持包括多线程程序的多个多线程进程。在一些情况下,软件执行环境可以是支持线程转储的获取的虚拟机。在一些实施例中,一个或多个监视进程可以在多线程程序旁边的虚拟机内执行。在一些实施例中,一个或多个监视进程可以在同一组机器上或在不同组机器上与虚拟机分开执行。一个或多个监视进程可以周期性地发起虚拟机的线程转储。对于特定的线程转储,可以为在获取该特定的线程转储时代表多线程程序执行(例如,由多线程程序产生)的每个线程获得栈踪迹。

[0157] 在步骤704处,实施例接收在线程转储期间正在执行的每个线程的栈踪迹。特定线

程的栈踪迹可以与描述该线程的调用栈的一行或多行文本对应。栈踪迹内的每一行与线程的调用栈上的特定栈帧对应，并且可以描述与该栈帧相关联的代码块。在一些实施例中，栈帧可以包括源代码文件和指向代码块的行号以及与代码块相关联的类名和/或方法名。

[0158] 在决策706处，实施例确定是否需要分析另一个栈踪迹。如果不需要，则流程图在步骤716处结束。具体而言，一旦已通过一个或多个监视进程分析了线程转储的所有栈踪迹，一些实施例就可以更新由存储器中的一个或多个对象封装的强度统计信息。例如，可以基于从线程转储获得哪种栈踪迹来更新一个或多个SeasonalTrendInfo对象的成员变量(例如，rawMeasure、rawDeseasonalizedMeasure、smoothedWeekdaySeasonalFactor和/或smoothedWeekendSeasonalFactor)。

[0159] 否则，在步骤708处，实施例确定现有的踪迹签名是否表示栈踪迹所包括的栈帧序列。具体而言，一些实施例可以使用已基于从先前的线程转储接收到的栈帧建立的现有的一组归类签名作为产生式文法，以确定栈帧的序列是否可以由现有签名之一来表示。这可能涉及一个或多个移位减少操作，其中栈踪迹的部分被折叠到叶子SegmentInfo节点中，并且SegmentInfo节点本身被折叠到合并节点中。如果移位减少操作导致被注册为归类签名的有序元组，则该归类签名表示栈踪迹所包括的栈帧序列。

[0160] 在决策710处，如果存在这样的踪迹(即，归类)签名，则流程图前进到步骤714。否则，在步骤712处，实施例生成新的踪迹签名，该新的踪迹签名表示由栈踪迹包括的栈帧的序列。换句话说，已经发现了曾被认为是线性连接的一组栈帧内的分支点。然后，一些实施例可以生成一个或多个SegmentInfo节点、修改一个或多个二叉树和/或修改一个或多个有序元组以生成新的归类签名，该新的归类签名表示由栈踪迹包括的一组(先前)线性连接的栈帧。下面关于图8更详细地描述生成新的归类签名的技术。

[0161] 在步骤714处，实施例在返回到决策706之前递增与踪迹签名相关联的计数器。具体而言，作为ThreadClassificationInfo对象、SegmentInfo对象和/或StackFrameInfo对象的成员的某些计数器(例如，numOfOccur和/或totalNumOfOccur)可以被递增以在接收和发现它们时按类型跟踪栈踪迹、栈段和栈帧的数量。

[0162] 图8图示了根据一些实施例的用于响应于检测到分支点而生成或修改线程归类签名的过程的流程图800。在一些实施例中，流程图800中描绘的过程可以由具有一个或多个处理器(例如，图17的计算机系统1700)的计算机系统来实现，其中该一个或多个处理器可以基于存储在计算机可读介质中的计算机代码来执行步骤。图8中描述的步骤可以以任何顺序执行，并且可以有或没有任何其它步骤。

[0163] 流程图800开始于步骤802，其中实施例确定先前是否已生成一个或多个SegmentInfo节点。如果是这样，则流程图前进到步骤804。否则，流程图前进到步骤814。除非当前正在分析的栈踪迹是为数据集接收到的第一个栈踪迹，否则该组归类签名将可能包含为之前的栈踪迹先前生成的一个或多个归类签名，其中这些归类签名包括SegmentInfo节点。由于从同一进程接收到的栈踪迹类型可能彼此共享栈段，因此第一次接收到的任何类型的栈踪迹都将可能导致发现分支点。

[0164] 在步骤804处，实施例确定栈踪迹所包括的栈帧序列中包括的、不由任何先前生成的节点表示的栈帧的一个或多个子序列。具体而言，一些实施例可以在尝试通过一系列移位减少操作来压缩由栈踪迹包含的栈帧序列时查阅现有的归类签名和SegmentInfo节点。

可以将不能被减少的序列的栈帧的任何子序列确定为新类型的栈段。在这种情况下，一些实施例可以确定需要生成表示该新类型的栈段的SegmentInfo节点。

[0165] 在步骤806，实施例生成一个或多个附加节点以表示栈帧的一个或多个子序列。具体而言，可以为新类型的栈段中包括的每个栈帧生成新的StackFrameInfo对象。可以生成与新类型的栈段对应的新的SegmentInfo节点，其中该新的SegmentInfo节点引用新的StackFrameInfo对象中的每一个。

[0166] 在步骤808处，实施例将一个或多个附加节点中的至少一个并入到一个或多个先前生成的元组的一个或多个先前生成的二叉树中。可以修改和/或扩展一个或多个现有归类签名的一个或多个二叉树以考虑新发现的分支点。在由新分支点分割由现有二叉树的叶SegmentInfo节点表示的栈段的情况下，该叶节点可以成为两个新叶SegmentInfo节点的合并节点。

[0167] 在步骤810处，实施例生成一个或多个附加二叉树，其中一个或多个二叉树中的至少一个或多个包括一个或多个附加节点中的至少一个。在许多情况下，该一个或多个附加二叉树可以是具有单个节点的单级树。新生成的二叉树之一可以包括在步骤806中生成的新SegmentInfo节点。

[0168] 在步骤812处，实施例生成包括一个或多个附加二叉树的附加元组以表示栈踪迹。该附加元组可以与表示新发现的栈踪迹类型的归类签名对应。一些元组可以是有序的单级二叉树集合，每个二叉树包含单个节点，并且可以看起来类似于节点列表。其它元组可以与单个多级二叉树对应。还有的其它元组可以包括组合的单级二叉树和多级二叉树。通常，随着发现越来越多类型的栈踪迹，生成的每个后续归类签名可能与越来越长的有序元组对应。然而，由于更可能首先遇到常见类型的栈踪迹，因此较长的归类签名更可能表示不常发生的栈踪迹。这可以确保将更高百分比的栈踪迹压缩成更短的归类签名。在步骤812之后，流程图在步骤820处结束。

[0169] 在步骤814处，实施例生成包括单个二叉树的元组，该单个二叉树包括表示栈踪迹的单个节点。由于未找到SegmentInfo节点，因此当前分析的栈踪迹可能是第一个。因此，一些实施例可以生成与仅具有一个SegmentInfo节点的单个二叉树对应的归类签名。在步骤814之后，流程图在步骤820处结束。将来一遇到不同类型的栈踪迹，就可以用新的SegmentInfo节点扩展二叉树以表示新遇到的分支点。

[0170] IV.以不规则的时间间隔进行堆使用量测量

[0171] 一些实施例可以使控制系统监视堆分配的时间系列数据(即，堆使用量)以估计趋势并预测虚拟机内的将来存储器使用量。通过检测季节性趋势并预测存储器容量要求，一些实施例可以在虚拟机之间动态地重新分配共享的系统存储器，从而实现资源分配的弹性。容量要求的预测可能涉及对堆增长率的估计。为了确保样本准确性，可以在完全垃圾收集(GC)周期期间进行堆分配测量，这以不规则时间间隔发生。对堆增长率的估计可能涉及除以随机时间间隔，这由于间歇性地任意接近于零的不规则时间间隔而复杂化。增长率测量中的噪声是产生柯西分布的两个高斯(Gaussian)分布的比率，其可能难以过滤。该柯西分布的均值和标准差是不确定的，从某种意义上说，大量数据点不比单个数据点产生更准确的均值和标准差的估计。增加样本池可以增加遇到具有与除以时间相近的间隔对应的大绝对值的样本点的可能性。

[0172] 应该注意的是,与由于完全GC周期的不规则性而导致其采样间隔不规则的堆大小测量不同,可以以规则间隔对线程强度测量进行采样以避免时间相近的间隔。即使如此,本文描述的用于堆分配的趋势分析的相同技术也可以应用于线程和栈段强度测量的季节性趋势分析和预测。在一些实施例中,该技术可以针对由于线程的CPU调度和完全GC周期的干扰所造成的可变延迟而调整。这些技术还可以针对由于对栈段进行归类所需的可变计算时间所造成的可变采样间隔进行调整。在线程转储中尚未观察到特定线程或栈段的情况下,一些实施例可能会将相关联的ThreadClassificationInfo对象或相关联的SegmentInfo对象的numOfOccur成员变量保留为零,这可以指示没有针对该特定线程或栈段的可用的测量。这样的实施例可能不更新SeasonalTrendInfo对象的rawMeasure变量。这样的实施例可以仅当相关联的ThreadClassificationInfo对象或相关联的SegmentInfo对象的numOfOccur成员变量不为零时,才更新SeasonalTrendInfo对象的rawMeasure成员变量。这些实施例可以跟踪自上次更新rawMeasure以来的线程转储的数量'N'。线程强度测量值可以与具有不规则时间间隔的系列对应。

[0173] 1957年和1960年出版的Holt-Winter三指数过滤器可以用于季节性趋势分析和预测。发表于Office of Naval Research Memorandum,no.52(1957)由C.C.Holt所著的"Forecasting Trends and Seasonal by Exponentially Weighted Averages"通过引用并入本文。发表于Management Science,vol.6,no.3,p.324-342(1960)由P.R.Wright所著的"Forecasting Sales by Exponentially Weighted Moving Averages"通过引用并入本文。Wright在1986年扩展了Holt-Winter公式,以支持不规则的时间间隔。发表于Management Science,vol.32,no.4,pp.499-510(1986)由D.J.Wright所著的"Forecasting data published at irregular time intervals using an extension of Holt's method"通过引用并入本文。在2008年,Hanzak提出了用于时间接近的间隔的调整因子。发表于WDS'08Proceedings Part I,pp.62-67(2008)由T.Hanzak所著的"Improved Holt Method for Irregular Time Series"通过引用并入本文。

[0174] 如果时间间隔在由存储器泄漏或死锁引起的拥塞期间单调减小,那么用于时间接近的间隔的调整因子(其意在补偿由于速率估计中的随机的时间接近的间隔引起的较高相对噪声强度)可能无意中抑制变化速率估计。完全GC算法的非线性或多项式时间复杂度会随着拥塞恶化而导致线程运行时间隔减少。在存储器泄漏的情况下,随着时间间隔减小,运行时间可能减少,但是测量时间可以增加,这是因为由于更频繁地执行完全GC,虚拟机可以被更长时间地冻结。如果虚拟机在完全GC期间被冻结,那么新请求可以在虚拟机外排队。这种积压(backlog)可以加速后续运行时期间堆使用量的变化速率。在一些实施例中,Hanzak对时间接近的间隔的调整被用于堆分配的趋势分析和预测以及跟踪加速的堆增长率。

[0175] 在本发明的实施例中,Holt-Winter三指数过滤器可以应用于堆使用量的季节性趋势分析和预测,以有效地实现存储器分配的弹性。可以应用于规则时间系列的需求预测的标准Holt-Winter三指数过滤器可以进行特殊调整,以针对具有不规则的时间接近的间隔的随机时间间隔工作。本发明的实施例可以应用用于不规则时间间隔的Wright公式和针对时间接近的间隔的Hanzak调整以进行堆分配的趋势分析和预测。可以执行适合于由完全GC产生的不规则时间间隔的过滤器结构的非平凡(non-trivial)选择。Holt-Winter-Wright-Hanzak过滤器的结构可以从第一原理推导出来,以系统地设计适应性以匹配由完

全GC周期生成的时间系列。

[0176] 在一些实施例中,应用指数移动平均的公式来平滑时间系列数据、局部线性趋势、季节性趋势、预测的误差残差,以及预测的绝对偏差,以便监视和预测资源利用率度量,诸如堆存储器使用量和线程强度。在一些实施例中,该公式可以基于1956年提出的Brown指数过滤器、1957年提出的Holt双指数过滤器、1960年提出的Winters三指数过滤器、1986年提出的针对不规则时间间隔的Wright扩展、2008年提出的针对时间接近的间隔的Hanzak调整因子,以及异常值检测和削波。以下出版物通过引用被包括在本文中:R.G.Brown,"Exponential Smoothing for Predicting Demand,"Cambridge,Arthur D.Little Inc.(1956),p.15;C.C.Holt,"Forecasting Trends and Seasonal by Exponentially Weighted Averages,"Office of Naval Research Memorandum,no.52,(1957);P.R.Winters,"Forecasting Sales by Exponentially Weighted Moving Averages,"Management Science,vol.6,no.3,p.324-342,(1960);D.J.Wright,"Forecasting data published at irregular time intervals using an extension of Holt's method,"Management Science,vol.32,no.4,pp.499-510(1986);T.Hanzak,"Improved Holt Method for Irregular Time Series,"WDS'08 Proceedings Part I,pp.62-67(2008);以及S.Maung,S.W.Butler and S.A.Henck,"Method and Apparatus for process Endpoint Prediction based on Actual Thickness Measurements,"United States Patent 5503707(1996)。

[0177] V.使线程强度和堆使用量相关

[0178] 各种实施例通过使由应用产生的各种线程类的强度统计信息与堆使用量统计信息之间的趋势相关来提供用于识别多线程应用内的堆囤积的栈踪迹(即,各类线程)的技术。在这样做时,一些实施例可以基于堆使用量统计信息来识别正在软件执行环境内执行一个或多个多线程应用的时间段内高的堆使用量趋向于高的季节(即,高的堆使用量季节)。如上所述,一些实施例然后可以通过分析在高的堆使用量季节的相同时间段内从软件执行环境获得的线程转储来识别和收集多个线程类的强度统计信息。然后,一些实施例可以通过按照线程类的强度统计信息和高的堆使用量趋势之间的相关程度对线程类进行排名来从识别出的线程类中识别"堆囤积的"线程类(即,堆囤积的栈踪迹)。

[0179] 一些实施例可以将这样的线程类称为堆囤积,因为这样的线程执行的代码很可能在堆存储器使用方面是低效的。换句话说,由这些线程执行的错误编写的代码和/或未优化的代码可能导致线程囤积大量的堆存储器,从而显著地导致高的堆使用量趋势。

[0180] 应该注意的是,从在生产环境中长时间操作基于云的服务的角度来看,这样的存储器热点是重要的。因此,通过使得能够连续检测和缓解这样的热点,一些实施例可以直接影响云服务的操作效率。还应该注意的是,这样的实施例可能优于使用存储器剖析器工具来剖析这样的应用,因为这样的工具可能给应用增加太多的开销。因此,存储器剖析器工具对于连续剖析正在生产环境中执行的应用可能是不切实际的。

[0181] A.代码中的低效的堆使用

[0182] 低效的存储器使用的一个常见原因是由于在线程的栈帧中定义的局部变量。通常,当正在运行的线程实例化对象时,该对象占用堆存储器,直到(直接或间接)引用该对象的栈帧的数量降到零,此时该堆存储器在下一次垃圾回收时被释放。因此,从长时间保持活

动的栈帧引用大对象的局部变量可能会无意中对堆存储器使用产生很大影响，因为它们不允许对这些对象进行垃圾回收。

[0183] 一些实施例假设总堆使用量'G'字节的一部分'p'可以归因于一类线程'C'。另外，一些实施例还可能假设这类线程'C'中的平均堆使用量(即，每线程的堆使用量)是'M'字节。在这个实例中，令'$T_C$'表示线程类'C'的预期线程数。以下关系给出'$T_C$'，其被定义为统计模型中的线程强度：

[0184] $$T_C = \frac{p\,G}{M}$$

[0185] 响应于识别堆囤积的线程类，某些实施例可以向开发人员、性能工程师和其它相关人员报告(例如，经由通知或警告)这些线程类。因此，与这种类型的线程相关联的代码可能需要进行详细的代码审查和代码剖析。在一些情况下，可以检查某些相关联的栈帧。例如，调查可能涉及获取在堆使用量接近季节性峰值期间的堆转储，以检查堆囤积的线程的栈踪迹中包含的栈帧。这些栈帧可能包含引用对高的堆使用量有贡献的对象(例如，占用大量堆存储器的对象)的局部变量。这种代码检查和优化可以通过可视代码审查、自动代码审查、识别出的线程的剖析、即时编译器优化、动态字节代码注入或这些技术的组合来完成。在一些实施例中，可以向其它自动代码优化工具报告堆囤积的线程类以利用它们的代码优化功能。

[0186] 一些实施例可以自动重新设计或重写应用代码以使其对存储器的使用更高效。例如，一些实施例可以自动重写代码，使得局部变量尽快释放大对象而不改变应用的行为或正确性。在一些情况下，这可能涉及深度分析堆囤积的线程中涉及的代码路径。

[0187] 例如，考虑以下代码：

[0188] fileOS.write(buffer.toString().getBytes());

[0189] 一些实施例可以确定上述代码在存储器使用方面是低效的，因为三个对象：buffer、buffer.toString()和buffer.toString().getBytes()由堆囤积线程的栈帧中的局部变量保持。具体而言，当线程正在文件系统调用中阻塞时，这些局部变量会阻止这三个对象被垃圾收集。

[0190] 一些实施例可以如下所示修改代码，使得当线程正在文件系统调用中阻塞时，至少两个对象:buffer和buffer.toString()可以被垃圾收集：

[0191] String temp1＝buffer.toString();

[0192] buffer＝new StringBuffer();//allow garbage collection of the old buffer

[0193] byte[]temp2＝temp1.getBytes();

[0194] temp1＝null;//allow garbage collection of the string

[0195] fileOS.write(temp2);//this is a blocking call

[0196] temp2＝null;//allow garbage collection of the bytes array

[0197] 一些实施例可以使用非侵入式方式来检查堆囤积的栈踪迹的栈帧。

[0198] B.初始化工作日和周末时段的季节因子

[0199] 为了识别堆囤积的栈踪迹，一些实施例可以(1)通过估计执行环境的堆使用量统计信息的季节性趋势来识别高的堆使用量季节，以及(2)针对一个或多个线程类中的每一

个估计线程类的线程强度统计信息的季节性趋势。在专利申请14/109,578、14/109,546和14/705,304中公开了针对规则或不规则的时间间隔用于确定堆使用量统计信息的季节性趋势和线程强度统计信息的季节性趋势的一些技术,这些申请通过引用并入本文,用于所有目的。

[0200] 为了确定统计的季节性趋势,可以定义季节性趋势被映射到的时段和间隔。具体而言,可以将时段划分为多个非重叠间隔。时段的每个间隔可以与季节索引相关联。例如,如果时段是一天并且间隔是一小时,那么应该有24个季节索引来涵盖该时段。作为另一个示例,如果时段是一年并且间隔是一个月,那么应该有12个季节索引。

[0201] 一些实施例可以将工作日、周末和假日建模为单独的时段。如果工作日和周末时段是分开的,那么可以有5个周期的工作日时段与1个周期的周末时段交错,使得在处理5个连续工作日时段之后,处理单个周末时段。因此,连续工作日时段的频率将是每24小时一个工作日时段,而周末时段的频率将是每7天一个周末时段。在各个假日(例如,圣诞节和新年假日)被建模为单独的时段的实施例中,特定假日时段的频率是一年一次。

[0202] 季节索引可以是应用于与季节索引相关联的间隔的乘性季节因子或加性季节项。例如,在使用乘性季节因子表示季节索引的实施例中,如果间隔'9-10AM'与季节因子1.3相关联,那么在9-10AM间隔期间采样的任何测量值可以被调高30%(即,乘以1.3)。在季节索引由加性季节项表示的实施例中,该加性季节项被加到测量值上。

[0203] 季节按一些标准对一组间隔进行归类。例如,给定一年的时间段,1月、2月、3月、4月、5月、6月、7月、8月、9月、10月、11月和12月的12个时段可以被归类为如下四个北方气象季节:

[0204] 12月、1月和2月被归类为冬季季节

[0205] 3月、4月和5月被归类为春季季节

[0206] 6月、7月和8月被归类为夏季季节

[0207] 9月、10月和11月被归类为秋季季节

[0208] 一些实施例可以将工作日时段划分为96个15分钟的间隔。在这点上,得到96个季节索引,其中96个工作日季节索引(即,工作日因子)中的每一个都映射到96个工作日间隔中的不同的一个工作日间隔。类似地,一些实施例可以将周末时段划分为192个15分钟的间隔,从而得到192个季节索引,其中192个周末季节索引(即,周末因子)中的每一个映射到192个周末间隔中的不同的一个周末间隔。

[0209] 为了分离出工作日时段的季节模式和周末时段的季节模式,某些实施例可以与将多时段趋势分析过滤器应用于周末时段分开地将多时段趋势分析过滤器应用于工作日时段。然后,一些实施例可以将工作日因子和周末因子重新归一化,使得季节因子1表示工作日时段和周末时段二者的公共参考水平。因此,大于1的季节因子可以表示在该季节因子所应用的间隔期间高于平均值的堆使用量。同时,小于1的另一个季节因子可以表示在该另一个季节因子所应用的另一个时间间隔期间低于平均值的堆使用量。

[0210] 在一些实施例中,可以扩展用于多时段趋势分析的技术以将假日(例如,劳动节、圣诞节、元旦等)分离为单独的时段,其中假日时段以每12个月一次的频率重复。同时,工作日时段以每24小时一次的频率重复,并且周末时段以每7天一次的频率重复。在这样的实施例中,假日时段的季节因子、工作日时段的季节因子和周末时段的季节因子可以全部被一

起重新归一化，使得季节因子1表示工作日时段、周末时段和假日时段的公共参考水平。

[0211] 给定时段(例如，工作日时段、周末时段或假日/一年时段等)，令P表示由给定测量数据集所覆盖的时段的周期数(例如，跨特定时间段的堆使用量测量的时间系列)并且令K表示由该给定数据集所覆盖的时段数内的间隔数。如果L表示时段中的季节索引的数量，则K＝P*L。例如，如果数据集内至少有3年的数据，一个时段对应于一年，并且一个间隔对应于一个月，那么该时段的可用周期的数量P为3，并且可用的月间隔的数量为36。

[0212] 一些实施例可以基于跨时段的多个周期的数据来计算该时段的每个间隔的平均堆使用量。具体而言，一些实施例可以枚举从0到(K-1)的间隔，并使用以下公式计算每个枚举间隔的平均堆使用量$\bar{x}_k$：

[0213] $$\bar{x}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{t_i},$$

[0214] k＝0,1,...,K-1;$N_k$是间隔k中的样本的数量;并且$t_i$是间隔k中的样本编号i的时间

[0215] 一些实施例还可以基于跨时段的多个周期的数据来计算该时段的每个周期的平均堆使用量。具体而言，一些实施例可以枚举从0到(P-1)的时段的周期，并使用以下公式计算该时段的每个枚举周期的平均堆使用量$D_p$：

[0216] $$D_p = \frac{1}{N_p}\sum_{i=1}^{N_p} x_{t_i}, \qquad p = 0,1,\ldots,P-1\ ;$$

[0217] $N_p$是时段的周期p中的样本的数量;并且$t_i$是时段的周期p中的样本编号i的时间

[0218] 为了初始化时段的季节因子，一些实施例可以使用以下公式计算时段中的每个季节索引$\bar{S}_l$的季节因子：

[0219] $$\bar{S}_l = \frac{1}{P}\left(\bar{x}_l/D_0 + \bar{x}_{l+L}/D_1 + \bar{x}_{l+2*L}/D_m + \cdots + \bar{x}_{l+(P-1)*L}/D_{P-1}\right), l = 0,1,\ldots,L-1$$

[0220] 具体而言，特定间隔的季节因子可以等于该间隔跨整个数据集的平均堆使用量(通过平均整个数据集(例如，跨越整个周的数据集)中所有相同间隔(例如，所有9-10AM间隔)的平均堆使用量来计算)与该时段跨整个数据集的平均堆使用量的比率。

[0221] C.重新归一化

[0222] 如上所述，一些实施例可以将工作日季节因子和周末季节因子重新归一化，使得季节因子“1”表示工作日时段和周末时段的公共参考水平。

[0223] 通常，某些实施例可以通过计算跨所有时段的季节因子的加权平均值并将每个季节因子除以加权平均值来执行重新归一化。考虑以下涉及不同长度的多个时段的季节索引的示例，其中每个时段被划分为15分钟的间隔：

[0224] 工作日的季节索引:$D_i$,i＝1,2,…96

[0225] 周末的季节索引:$E_i$,i＝1,2,…,192

[0226] 各个假日的季节索引:$H_{k,i}$,i＝1,2,…,96;k＝1,2,…10

[0227] 假设在特定年份，有253个工作日(假日除外)、50.5个周末和10个假日，其中253+

50.5*2+10＝364天。在该示例中，一些实施例可以使用以下公式来计算季节因子的加权平均值“A”，其中权重与一年中的每个时段(例如，工作日时段、周末时段和10个各个假日时段)的周期的数量成比例。

[0228] $$A = \frac{1}{364}\left(253\sum_{i=1}^{86} D_i + 50.5\sum_{i=1}^{192} E_i + 10\sum_{k=1}^{10}\sum_{i=1}^{86} H_{k,i}\right)$$

[0229] 一些实施例可以通过将每个季节因子$D_i$，$E_i$和$H_{k,i}$除以A得到每个时段的新的重新归一化的季节因子。

[0230] 返回到用于识别堆囤积的栈踪迹的步骤，在使用上述公式初始化季节索引$\bar{S}_l$之后，一些实施例可以通过将每个周末因子$\bar{B}_k$和每个工作日因子$\bar{C}_l$除以归一化因子来重新归一化工作日因子和周末因子，如下所示：

[0231] $$\frac{1}{K+5L}\left(\sum_{k=0}^{K-1}\bar{B}_k + 5\sum_{l=0}^{L-1}\bar{C}_l\right)$$

[0232] 在工作日因子和周末季节因子的重新归一化之后，季节因子1应该表示工作日因子和周末因子两者的公共参考水平。

[0233] D.平滑样条拟合

[0234] 如上所述，一些实施例可以跨多个时段拟合平滑样条以在时段的周期之间(例如，两个工作日时段之间)或两个相邻时段的周期之间(例如，工作日时段和周末之间)提供平滑过渡。具体而言，拟合样条可以涉及将一个或多个时段的季节索引连接以平滑这些时段之间的过渡。

[0235] 通常，当某些实施例(例如，过滤器)到达时段$A_i$的周期的结束并且开始时段$A_i$的新的周期时，诸如当在从星期一到星期二、星期二到星期三、星期三到星期四和星期四到星期五的过渡处重复工作日周期时，这样的实施例可以连接季节索引$A_i$的三个序列并且跨整个序列拟合平滑样条。然后，一些实施例可以采用经平滑的序列的中间段来表示新的经平滑的季节索引$A_i$。

[0236] 当某些实施例(例如，过滤器)到达时段$A_i$的周期的结束并且开始相邻时段$B_i$的新的周期时，诸如当从星期五过渡到星期六时，一些实施例可以连接季节索引$A_i$的一个序列、季节索引$B_i$的一个序列，以及在时段$B_i$之后的时段的季节索引$C_i$的一个序列，并且跨整个序列拟合平滑样条。然后，一些实施例可以采用经平滑的序列的中间段来表示新的经平滑的季节索引$B_i$。一些实施例也可以采用经平滑的序列的第一段来表示经平滑的季节索引$A_i$。

[0237] 当某些实施例(例如，过滤器)到达时段$B_i$的周期的结束并且开始相邻时段$C_i$的新的周期时，诸如当从星期日过渡到星期一时，一些实施例可以连接时段$B_i$之前的时段的季节索引$A_i$的一个序列、季节索引$B_i$的一个序列和季节索引$C_i$的一个序列，并且跨整个序列拟合平滑样条。然后，一些实施例可以采用经平滑的序列的中间段来表示新的经平滑的季节索引$B_i$。一些实施例也可以采用经平滑的序列的第三段来表示新的经平滑的季节索引$C_i$。

[0238] 关于云服务，周末和假日期间的负载周期通常不同于工作日期间的负载周期。常规的季节趋势分析解决方案通常只可以表示一个时段的季节索引。为了将周末的季节索引

与常规工作日的季节索引分离，这种常规的解决方案可能依赖于将时段的范围延长到整个周或整个月。此外，这种常规解决方案可能单独地处理假日。

[0239] 返回到用于识别堆囤积的栈踪迹的步骤，为了平滑工作日季节因子，一些实施例可以通过连接工作日因子的三个序列来组成季节因子阵列。例如，一些实施例可以通过执行以R编程语言的以下代码来生成阵列：

[0240] factors<-c(smoothedWeekdaySeasonalFactor,

[0241] smoothedWeekdaySeasonalFactor,

[0242] smoothedWeekdaySeasonalFactor)

[0243] 接下来，一些实施例可以应用样条来平滑工作日因子的阵列。例如，一些实施例可以用0.3的平滑参数来调用R的smooth.spline函数以平滑因子：

[0244] extendedWeekdayIndices<-1:(3*96)

[0245] f<-smooth.spline(extendedWeekdayIndices,factors,spar＝0.3)

[0246] 然后，一些实施例可以将阵列内的中间序列(即，中间96个工作日因子)指定为经平滑的工作日因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的工作日因子：

[0247] sandwichWeekdayIndices<-(96+1):(96*2)

[0248] smoothedWeekdaySeasonalFactor<-predict(f,sandwichWeekdayIndices)$y

[0249] 以类似于平滑工作日因子的方式，一些实施例可以应用样条来平滑周末因子。具体而言，一些实施例可以通过在两个工作日因子序列之间连接周末因子序列来组成季节因子的阵列。例如，一些实施例可以通过执行以R编程语言的以下代码来生成阵列：

[0250] factors<-c(smoothedWeekdaySeasonalFactor,

[0251] smoothedWeekendSeasonalFactor,

[0252] smoothedWeekdaySeasonalFactor)

[0253] 接下来，一些实施例可以应用样条来平滑工作日和周末因子的阵列。例如，一些实施例可以用0.3的平滑参数来调用R的smooth.spline函数以平滑因子：

[0254] extendedWeekendIndices<-1:(2*96+192)

[0255] f<-smooth.spline(extendedWeekendIndices,factors,spar＝0.3)

[0256] 然后，一些实施例可以将阵列内的中间序列(即，阵列内的中间192个季节因子，其为周末因子)指定为平滑的周末因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的周末因子：

[0257] sandwichWeekendIndices<-(96+1):(96+192)

[0258] smoothedWeekendSeasonalFactor<-predict(f,sandwichWeekendIndices)$y

[0259] 应该注意的是，一些实施例可以分别表示96个工作日季节索引和192个周末季节索引，以便将工作日期间观察到的季节模式与周末期间观察到的季节模式分开。在一些实施例中，顺序过滤堆使用量统计信息的时间系列可以涉及五组指数过滤器，包括一组用于堆使用量测量、一组用于季节因子、一组用于线性趋势、一组用于加速趋势以及一组用于残差。

[0260] 如上所述，为了确保样本准确性，可以在以不规则时间间隔发生的完全垃圾收集(GC)周期期间进行堆分配测量。在堆使用量特别高的情况下，由于持续的垃圾收集，采样间

隔可能变得任意接近于零。由于预测涉及变化速率的估计，因此如果不规则时间间隔变得任意接近于零，那么变化速率可能成为柯西分布的随机变量，其均值和标准差是不确定的。因此，一些实施例可以采用Holt的双指数过滤器、Winters的三指数过滤器、Wright的针对不规则时间间隔的扩展、Hanzak的针对时间相近的间隔的调整因子、以及具有异常值截止的自适应缩放的异常值检测和削波，来克服用于确定与完全GC相关联地确定的统计信息的季节性趋势的柯西分布问题。在一些实施例中，可以将五组指数过滤器顺序地应用于时间系列以估计工作日因子和周末因子。

[0261] 当某些实施例(例如，过滤器)到达工作日和周末时段的处理周期的结束时，在处理该时段的下一个周期或过渡到不同时段(例如，从工作日时段过渡到周末时段)之前，这样的实施例可以如下将每个周末因子$\bar{B}_k$和工作日因子$\bar{C}_l$除以归一化因子

[0262] $$\frac{1}{K+5L}\left(\sum_{k=0}^{K-1}\bar{B}_k+5\sum_{l=0}^{L-1}\bar{C}_l\right)$$

[0263] 在每个时段结束之后，一些实施例可以应用样条来平滑季节因子。例如，当到达另一个工作日时段之前的工作日时段的结束时(即，当从星期一到星期二、从星期二到星期三、从星期三到星期四、或从星期四到星期五过渡时)，一些实施例可以通过连接工作日因子的三个序列来组成季节因子阵列。例如，一些实施例可以通过执行以R编程语言的以下代码来生成阵列：

[0264] factors<-c(smoothedWeekdaySeasonalFactor,

[0265] smoothedWeekdaySeasonalFactor,

[0266] smoothedWeekdaySeasonalFactor)

[0267] 接下来，一些实施例可以应用样条来平滑工作日因子的阵列。例如，一些实施例可以用0.3的平滑参数来调用R的smooth.spline函数以平滑因子：

[0268] extendedWeekdayIndices<-1:(3*96)

[0269] f<-smooth.spline(extendedWeekdayIndices,factors,spar＝0.3)

[0270] 然后，一些实施例可以将阵列内的中间序列(即，中间96个工作日因子)指定为经平滑的工作日因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的工作日因子：

[0271] sandwichWeekdayIndices<-(96+1):(96*2)

[0272] smoothedWeekdaySeasonalFactor<-predict(f,sandwichWeekdayIndices)$y

[0273] 在不同的实例中，当到达周末时段之前的工作日时段的结束时(即，当从星期五过渡换到星期六时)，一些实施例可以通过在工作日季节因子的两个序列之间连接周末季节因子序列来组成季节因子阵列。例如，一些实施例可以通过执行以R编程语言的以下代码来生成阵列：

[0274] factors<-c(smoothedWeekdaySeasonalFactor,

[0275] smoothedWeekendSeasonalFactor,

[0276] smoothedWeekdaySeasonalFactor)

[0277] 接下来，一些实施例可以应用样条来平滑工作日和周末因子的阵列。例如，一些实

施例可以用0.3的平滑参数来调用R的smooth.spline函数以平滑因子：

[0278] extendedWeekendIndices<-1:(2*96+192)

[0279] f<-smooth.spline(extendedWeekendIndices,factors,spar＝0.3)

[0280] 然后，一些实施例可以将阵列内的左序列(即，阵列内的前96个季节因子，其为工作日因子)指定为经平滑的工作日因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的工作日因子：

[0281] leftsideWeekendIndices<-1:96

[0282] smoothedWeekdaySeasonalFactor<-predict(f,leftsideWeekendIndices)$y

[0283] 在不同的实例中，当到达周末时段的结束时(即，从星期日过渡到星期一)，一些实施例可以通过在工作日季节因子的两个序列之间连接周末季节因子序列来组成季节因子的阵列。例如，一些实施例可以通过执行以R编程语言的以下代码来生成阵列：

[0284] factors<-c(smoothedWeekdaySeasonalFactor,

[0285] smoothedWeekendSeasonalFactor,

[0286] smoothedWeekdaySeasonalFactor)

[0287] 接下来，一些实施例可以应用样条来平滑工作日和周末因子的阵列。例如，一些实施例可以用0.3的平滑参数来调用R的smooth.spline函数以平滑因子：

[0288] extendedWeekendIndices<-1:(2*96+192)

[0289] f<-smooth.spline(extendedWeekendIndices,factors,spar＝0.3)

[0290] 然后，一些实施例可以将阵列内的中间序列(即，阵列内的中间192个季节因子，其为周末因子)指定为经平滑的周末因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的周末因子：

[0291] sandwichWeekendIndices<-(96+1):(96+192)

[0292] smoothedWeekendSeasonalFactor<-predict(f,sandwichWeekendIndices)$y

[0293] 一些实施例还可以将阵列内的右序列(即，阵列内的最后96个季节因子，其为工作日因子)指定为经平滑的工作日因子。例如，一些实施例可以通过执行以R编程语言的以下代码来获得经平滑的工作日因子：

[0294] rightsideWeekendIndices<-(96+192+1):(2*96+192)

[0295] smoothedWeekdaySeasonalFactor<-predict(f,rightsideWeekendIndices)$y

[0296] 应该注意的是，一些实施例可以在每次顺序过滤器到达(1)时段的周期的结束并且开始相同时段的新周期时(例如，顺序过滤器到达星期一的结束)或者(2)时段的周期的结束并且开始相邻时段的新周期时(例如，顺序过滤器到达星期五的结束)执行上述的重新归一化和平滑样条拟合。

[0297] E.针对季节性周期的测试

[0298] 一些实施例可以针对数据集的一个或多个候选时段来测试季节性周期的存在，以确定是否应该表示该时段的单独的季节索引序列。通常，为了确定数据集是否表现出特定时段的季节性周期，一些实施例可以执行以下步骤。

[0299] 令Q表示时段中的季节索引的数量，P表示时段的可用周期的数量，并且K表示跨时段的这些周期的可用间隔的数量，其中K＝P*Q。

[0300] 一些实施例可以计算时段的周期的每个间隔中的平均度量。为此，一些实施例可

以枚举从0到(K-1)的间隔,并使用下面的公式计算该时段的每个间隔的平均度量:

[0301] $$\bar{x}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{t_i} \quad ,$$

[0302] k=0,1,...,K-1;$N_k$是间隔k中的样本的数量;并且$t_i$是间隔k中的样本编号i的时间

[0303] 然后,一些实施例可以计算时段的每个周期的平均度量。为此,一些实施例可以枚举从0到(P-1)的时段的周期,并使用下面的公式计算时段的每个周期的平均度量:

[0304] $$D_p = \frac{1}{N_p}\sum_{i=1}^{N_p} x_{t_i}, \qquad p = 0,1,\dots,P-1 \;;$$

[0305] $N_p$是时段的周期p中的样本的数量;

[0306] 并且$t_i$是时段的周期p中的样本编号i的时间

[0307] 然后,一些实施例可以使用以下公式计算该时段的每个周期p的季节索引的序列:

$$\bar{Y}_p = < \bar{Y}_{p,0}, \bar{Y}_{p,1}, \dots, \bar{Y}_{p,Q-1} >$$

[0308] $$\bar{Y}_{p,j} = \frac{\bar{x}_j}{D_p}, j = k \; modulo \; Q \;;\; k = 0,1,\dots,K-1$$

$$p = 0,1,\dots,P-1$$

[0309] 然后,一些实施例可以应用零(null)假设测试来检测时段中是否存在季节性周期。在这点上,待测试的零假设可以对应于最近周期'u'的季节索引与前一周期'v'的季节索引之间的相关系数$r_{u,v}$为零的假设。具体而言,一些实施例可以使用以下公式确定相关系数$r_{u,v}$:

[0310] $$Mean(\bar{Y}_p) = \frac{1}{Q}\sum_{j=0}^{Q-1} \bar{Y}_{p,j}$$

[0311] $$Variance(\bar{Y}_p) = \frac{1}{11}\sum_{j=0}^{Q-1} \left(\bar{Y}_{p,j} - Mean(\bar{Y}_p)\right)^2$$

[0312] $$r_{u,v} = \frac{\sum_{j=0}^{Q-1}\left(\bar{Y}_{u,j} - Mean(\bar{Y}_u)\right)\left(\bar{Y}_{v,j} - Mean(\bar{Y}_v)\right)}{11\sqrt{Variacle(\bar{Y}_u)\,Variacle(\bar{Y}_v)}}$$

[0313] 一些实施例可以采用各种技术来确定相关系数$r_{u,v}$是否足够大以指示在显著性水平之上在周期'u'和'v'之间存在共同的季节性周期。例如,一些实施例可以采用Student-t测试、置换测试或Fisher变换。

[0314] 为了测试该假设,一些实施例可以定义一个或多个测试统计,其可以是参数的函数。在这种情况下,待测试的是相关系数$r_{u,v}$。以下测试统计t具有Student's t-分布,具有'

n-2'的自由度并且是$r_{u,v}$的函数。一些实施例定义零假设$r_{u,e}=0$,其假定季节索引在时段的周期之间不相关。一些实施例可以搜索证据以通过接受替代假设来拒绝零假设(即,$r_{u,e}=0$)。

[0315] $$t = r_{u,v}\sqrt{\frac{n-2}{1-r_{u,v}^2}}$$

[0316] 令F(t)表示概率分布。给定0.1的显著性水平,令$t_{0.9,(n-2)}$表示随机变量t的值,使得F(t)=0.9。替代假设是单方面条件:

[0317] $$r_{u,v}\sqrt{\frac{n-2}{1-r_{u,v}^2}} > t_{0.9,(n-2)}$$

[0318] 如果该条件为真,则接受替代假设,这表明在周期年份'u'和'v'之间存在共同的季节性周期。如果在最近的周期和前一周期之间存在共同的季节性周期,则一些实施例可以继续计算周期的每个季节索引的季节因子。一些实施例应用上述公式来检测软件执行环境的堆使用量的年度季节性周期的存在,如下所述。

[0319] F.检测堆使用量的年度季节性周期

[0320] 当分析软件执行环境的多年堆使用量统计信息时,一些实施例可以以不同时间尺度检测多于一个的季节性趋势。例如,这样的实施例可以检测多年时间系列的堆使用量统计信息、年度季节性趋势和每日季节性趋势,它们都被叠加到多季节性趋势上。因此,一些实施例可以采用适当的时间尺度来分析年度季节性趋势,其中该时间尺度具有与1年对应的时段和与1个月对应的间隔。因此,可以将一年长的时段划分为12个一月长的间隔。

[0321] 为了确定数据集是否表现出年度季节性周期,一些实施例可以首先确定数据集中的月度索引的乘性因子。

[0322] 在特定实例中,令P表示数据集中的可用年数(即,一年时段的周期数)。此外,令Q表示数据集中的可用月数(即,周期数内的间隔数)。因此,Q=12*P。令K表示数据集中可用的工作日或周末的数量。令索引k的范围从0到(K-1),以表示可用工作日或周末的枚举。令$N_k$表示第k个工作日或周末中的样本的数量。使用以下公式,一些实施例可以应用以下公式来计算数据集中每个工作日或周末的平均堆使用量:

[0323] $$\bar{x}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{t_i}, k = 0,1,\ldots,K-1$$并且$N_k$是天k中的样本的数量

[0324] 一些实施例可以定义函数H,H:(Year×Integer)→Index,其映射包括年份的索引和与该年内的工作日或周末的索引对应的整数的有序对。使用以下公式,一些实施例然后可以计算根据该年内的工作日或周末的平均堆使用量计算每年的平均堆使用量:

[0325] $$\bar{Z}_p = \frac{1}{N_p}\sum_{i=1}^{N_p} \bar{x}_{H(p,i)}, \quad p = 0,1,\ldots,P-1 \ ;$$

[0326] $N_p$是一年时段的周期p中的工作日的数量;

[0327] 并且H(p,i)是一年时段的周期p中的第i个工作日的索引。

[0328] 一些实施例定义函数G,G:(Month×Integer)→Index,其映射包括月份的索引和与该月内的工作日或周末的索引对应的整数的有序对。使用以下公式,一些实施例可以根据该月内的工作日或周末的平均堆使用量计算该时段的每个月份间隔的平均堆使用量:

[0329] $$\bar{Y}_{p,m}=\frac{1}{M_m\bar{Z}_p}\sum_{i=1}^{M_m}\bar{x}_{G(m,i)},m=0,1,\ldots,Q-1\ ;$$

[0330] $M_m$是月份m中的工作日的数量;

[0331] G(m,i)是月份m中的第i个工作日的索引;

[0332] 并且p是年时段的周期的索引。

[0333] 具体而言,上述公式产生$\bar{\boldsymbol{Y}}_{\boldsymbol{p}}=<\bar{\boldsymbol{Y}}_{\boldsymbol{p,0}},\bar{\boldsymbol{Y}}_{\boldsymbol{p,1}},\ldots,\bar{\boldsymbol{Y}}_{\boldsymbol{p,Q}}>$,其对应于一年长时段p的每个周期的12个月度平均值。一个月的平均堆使用量可以除以年度平均堆使用量,以获得与该月对应的月度索引的乘性因子。在这样的实施例中,如果确定特定月份的乘性因子大于1,则该月份中的堆使用量高于平均值。另一方面,如果确定特定月份的乘性因子小于1,则该月份中的堆使用量低于平均值。

[0334] 在确定月度索引的乘性因子之后,一些实施例可以应用零假设测试来检测是否存在年度季节性周期。在这点上,待测试的零假设可以对应于最近年'u'的月索引与前一年'v'的月索引之间的相关系数$r_{u,e}$为零的假设。具体而言,一些实施例可以使用下面的公式来确定相关系数$r_{u,v}$:

[0335] $$Mean(\bar{Y}_p)=\frac{1}{12}\sum_{j=1}^{12}\bar{Y}_{p,j}$$

[0336] $$Variance(\bar{Y}_p)=\frac{1}{11}\sum_{j=1}^{12}\left(\bar{Y}_{p,j}-Mean(\bar{Y}_p)\right)^2$$

[0337] $$r_{u,v}=\frac{\sum_{j=1}^{12}\left(\bar{Y}_{u,j}-Mean(\bar{Y}_u)\right)\left(\bar{Y}_{v,j}-Mean(\bar{Y}_v)\right)}{11\sqrt{Variacle(\bar{Y}_u)\,Variacle(\bar{Y}_v)}}$$

[0338] 一些实施例可以采用各种技术来确定相关系数$r_{u,v}$是否足够大以指示在显著性水平之上在年份'u'和'v'之间存在共同的季节性周期。例如,一些实施例可以采用Student-t测试、置换测试或Fisher变换。

[0339] 如果零假设为真(即,$r_{u,v}=0$),则以下测试统计t具有Student's t-分布,具有'n-2'的自由度

[0340] $$t=r_{u,v}\sqrt{\frac{n-2}{1-r_{u,v}^2}}$$

[0341] 令F(t)表示概率分布。给定0.1的显著性水平,令$t_{0.9,(n-2)}$表示随机变量t的值,使得F(t)=0.9。替代假设是单方面条件:

[0342] $$r_{u,v}\sqrt{\frac{n-2}{1-r_{u,v}^2}} > t_{0.9,(n-2)}$$

[0343] 接受替代假设的条件表明在年份'u'和'v'之间存在共同的季节性周期。

[0344] G.确定年度高的堆使用量季节

[0345] 如果确定在最近年份和前几年之间存在共同的季节性周期,则一些实施例可以通过采用以下公式来计算通过月度季节索引0至11枚举的每个月的季节因子:

[0346] $$\bar{S}_n = \frac{1}{P}\left(\bar{Y}_n + \bar{Y}_{n+12} + \bar{Y}_{n+2*12} + \cdots + \bar{Y}_{n+(P-1)*12}\right), n = 0,1,\ldots,11$$

[0347] 在替代实施例中,最近年份(即,周期)的月度索引$\bar{Y}_p$可以用作月度季节索引,如以下公式所指示的:

[0348] $$\bar{S}_n = \bar{Y}_{p,n}, n = 0,1,\ldots,11$$

[0349] 为了对年度高的堆使用量季节进行归类,一些实施例可以识别与在一年长的时段中具有最大季节因子的月份对应的季节索引N。然后,这些实施例可以使用索引N作为种子。从N开始,这样的实施例可以扫描具有大于阈值T的季节因子的小于或大于N的季节索引(即,季节索引0、1、2...N-1、N+1、N+2)。在一些实施例中,T大于1。如果存在多于一个N使得$\bar{S}_N = MAX(\bar{S}_n)$,则一些实施例可以对多于一个不相交的高的堆使用量季节进行归类。函数$MAX(\bar{S}_n, s)$选择索引序列的第s个元素N,$\bar{S}_N = MAX(\bar{S}_n)$。参数s用于在存在多于一个N使得$\bar{S}_N = MAX(\bar{S}_n)$的情况下打破平局。一些实施例可以对每个不相交的高的堆使用量季节进行归类并且针对每个季节重复相关性分析。在一些实施例中,通过以下递归来定义针对高的堆使用量季节对月度季节索引的集合进行归类的方法:

[0350] $$V_0 = \{ N \mid \bar{S}_N = \mathbf{MAX}(\bar{S}_n, s)_{n=0,\ldots,(P-1)} \}$$

[0351] $$V_{W+1} = V_W \cup \{ J \mid |\bar{S}_J| > T \text{ 且 } [[J = (K+1) \bmod P, K = \mathbf{MAX}(V_W)] \text{ or } [J = (L-1+P) \bmod P, L = \mathbf{MIN}(V_W)]] \}$$

[0352] $$V = \bigcup_{W=0\ldots(P-1)} V_W$$

[0353] 应该注意的是,上面的递归涉及可以用于打破平局的无约束变量s。在一些实施例中,默认s=1。

[0354] 在某些实施例中,季节索引的闭包V对年度高的堆使用量季节进行归类。阈值T可以被设置为百分比,诸如季节因子的范围的85%。例如,假设一年长时段中的12个月度季节索引的季节因子如下表所给出。

[0355]

| 1月 | 2月 | 3月 | 4月 | 5月 | 6月 | 7月 | 8月 | 9月 | 10月 | 11月 | 12月 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.76 | 0.82 | 1.0 | 1.2 | 1.29 | 1.34 | 1.26 | 1.12 | 1.01 | 0.99 | 0.95 | 0.9 |

[0356] 乘性季节因子的范围是(1.34-0.76),即0.58。因此,季节因子的范围的85%是(0.76+0.85*0.58),即1.253。给定85%阈值T,T=1.25。因此,这样的实施例可以将5月、6月

和7月归类为年度高的堆使用量季节。

[0357] 一些实施例可以选择数据集的跨一年长时段的最近周期的段。例如，在2013年、2014年、2015年和2016年的周期当中，这些实施例可以选择覆盖2015年至2016年的数据的段。所选择的数据段可以跨越堆使用量统计信息中年度高的堆使用量季节内的2个或更多个星期。例如，如果季节因子如下表所给出，则数据段可以选自2015年11月、2015年12月和2016年1月。

[0358]

| 1月 | 2月 | 3月 | 4月 | 5月 | 6月 | 7月 | 8月 | 9月 | 10月 | 11月 | 12月 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.26 | 1.12 | 1.01 | 0.99 | 0.95 | 0.9 | 0.76 | 0.82 | 1.0 | 1.2 | 1.29 | 1.34 |

[0359] H.过滤常数和时区偏移的回归

[0360] 一些实施例可以包含对时区偏移的估计。如果时区偏移不可用，则一些实施例可以对数据集的段执行非线性回归以估计时区偏移并使用它来过滤数据。通过提供对时区偏移的估计，一些实施例可以改进对在时段之间的过渡中的季节索引的估计。

[0361] 具体而言，一些实施例可以用以下过滤器常数(即，作为独立变量的回归参数)执行非线性回归：measureFilterConstantα，rateFilterConstantβ，accelerationFilterConstantκ，seasonalFactorFilterConstantγ，errorResidualFilterConstantδ和timeZoneOffset tz，以最小化1-步预测的残差的均方误差(MSE)和/或平均绝对偏差(MAD)。在一些实施例中，时间戳可以在回归中以时区偏移tz被移位。一些实施例可以使用优化例程(例如，由R编程语言提供的优化例程)来应用非线性多元回归。一些实施例可以使用α,β,κ,γ,δ和tz的最佳值导出工作日和周末季节因子，如这些实施例所采用的以下公式中所指示的：

[0362] $$MSE = f(\alpha,\beta,\kappa,\gamma,\delta,tz,x_t) = \left(\frac{1}{N}\sum_{n=1}^{N}\left(e_{h,t_n}\right)^2\right)^{1/2}$$

[0363] $$MAD = f(\alpha,\beta,\kappa,\gamma,\delta,tz,x_t) = \frac{1}{N}\sum_{n=1}^{N}\left|e_{h,t_n}\right|$$

[0364] 一些实施例包括时区偏移作为回归参数，使得时段的周期之间或两个相邻时段之间的过渡可以尽可能地准确。

[0365] I.按相关程度对线程类进行排名

[0366] 一旦确定年度高的堆使用量季节，一些实施例就可以计算和/或获得表示由近来(例如，最近)年度高的堆使用量季节所覆盖的每日/每周季节周期的工作日/周末因子。应该注意的是，数据集的该段中(即，在年度高的堆使用量季节期间)的每日/每周季节性周期可能比在其它时间(即，在年度高的堆使用量季节之外)更加明显。因此，确定堆使用量中的季节性趋势与一个或多个线程类的强度统计信息中的季节性趋势之间的相关程度可以基于数据集的该段。换句话说，对于相关性分析，一些实施例可以使用与由最近的年度高的堆使用量季节所覆盖的时间间隔相同的时间间隔来导出各个线程类的季节性趋势。

[0367] 应该注意的是，为了确定特定线程类的强度统计信息的季节性趋势，一些实施例

可以采用如上所述的用于确定堆使用量的季节性趋势的技术。换句话说，线程强度统计信息和堆使用量统计信息的季节性趋势两者都可能涉及对工作日和周末时段使用相同数量的季节索引(例如，用于工作日时段的96个季节索引和用于周末时段的192个季节索引)。

[0368] 在确定堆使用量的季节性趋势和一个或多个线程类的强度统计信息的季节性趋势后，一些实施例然后可以针对这一个或多个线程类中的每一个来计算堆使用量的季节性趋势和线程类的强度统计信息的季节性趋势之间的相关程度。具体而言，可以针对96个季节因子或192个季节因子的序列计算相关程度。应该注意的是，计算季节性趋势之间的相关程度可能比计算堆使用量度量序列与线程强度度量序列之间的相关程度更高效，因为度量的序列可能长得多。

[0369] 令H表示用于堆使用量的有N个季节因子的序列。令T表示一类线程的线程强度的有N个季节因子的序列。季节因子的两个序列的相关系数由CorrelationCoefficient(H,T)给出，如下面所定义的：

[0370] $$Mean(H) = \frac{1}{N}\sum_{j=1}^{N} H_j$$

[0371] $$Mean(T) = \frac{1}{N}\sum_{j=1}^{N} T_j$$

[0372] $$Variance(H) = \frac{1}{N-1}\sum_{i=1}^{N}(H_i - Mean(H))^2$$

[0373] $$Variance(T) = \frac{1}{N-1}\sum_{i=1}^{N}(T_i - Mean(T))^2$$

[0374] $$CorrelationCoefficient(H,T) = \frac{\sum_{i=1}^{N}(H_i - Mean(H))(T_i - Mean(T))}{(N-1)\sqrt{Variacle(H)\,Variacle(T)}}$$

[0375] 一些实施例可以通过对最近年度高的堆使用量季节中包括的堆使用量统计信息进行回归来导出堆使用量的工作日和周末季节因子。让我们用(t1,t2)来表示数据集的该段的时间间隔。为了分析一类线程的强度统计信息的季节因子与堆使用量的季节因子之间的相关性，一些实施例可以采用来自在与该线程类相关联的SeasonalTrendInfo中的季节因子时间系列中的相同时间间隔(t1,t2)的季节因子。具体而言，该季节因子时间系列可以被存储在相关联的SeasonalTrendingInfo对象中的smoothedWeekdaySeasonalFactor成员变量和smoothedWeekendSeasonalFactor成员变量中。

[0376] 一些实施例可以遍历所有线程类的ThreadClassificationInfo对象，并递归地遍历每个ThreadClassificationInfo对象中的SegmentInfo对象，以收集ThreadClassificationInfo对象和SegmentInfo对象中包含的SeasonalTrendInfo对象。在使用上面识别出的公式计算堆使用量和每个线程类之间的CorrelationCoefficient(H,T)时，一些实施例可以检索每个SeasonalTrendInfo对象中的工作日因子或周末因子。一旦已经为每线程类计算相关程度，一些实施例就可以通过线程类与堆使用量季节性趋势的相关

程度来对线程类进行排名。然后，高排名的线程类可以被归类为堆囤积的线程类。然后，一些实施例可以分析与堆囤积的线程类相关联的栈踪迹和代码，以识别可以被手动或自动地纠正和/或改进的低效存储器使用。

[0377] 应该注意的是，可以扩展一些实施例以基于除工作日和周末时段之外的时段(例如，季度末时段)来确定相关系数。

[0378] 图9图示了根据一些实施例的用于识别可能对软件执行环境内的高的堆使用量有贡献的代码的过程的流程图900。在一些实施例中，流程图900中描绘的过程可以由具有一个或多个处理器(例如，图17的计算机系统1700)的计算机系统来实现，其中该一个或多个处理器可以基于存储在计算机可读介质中的计算机代码来执行步骤。图9中描述的步骤可以以任何顺序执行，并且可以有或没有任何其它步骤。

[0379] 流程图900开始于步骤902，其中实施例确定一个或多个进程的堆使用量超过阈值的时间长度。该时间长度可以与年度高的堆使用量季节对应，而阈值可以与指派给跨一个或多个时段(例如，工作日时段和周末时段)的间隔(例如，15分钟间隔)的季节因子的范围的百分比对应。在一些实施例中，可以通过选择百分比来设置阈值。一旦选择了百分比，就可以由季节因子的范围和百分比的乘积与最小季节因子的和给出阈值。例如，如果所选的百分比为85％，最小季节因子为0.76，并且最大季节因子为1.34，则阈值可以被给出为(0.76+0.85*(1.34-0.76))，即1.253。因此，可以将具有超过1.25的乘性季节因子的任何间隔确定为堆使用量超过阈值的时间长度的一部分。

[0380] 在步骤904处，实施例确定该一个或多个进程在该时间长度期间的堆信息。堆信息可以与在该时间长度期间在不同点处由软件执行环境内的该一个或多个进程使用的堆存储器的量对应。例如，堆信息可以基于以不规则间隔(例如，在完全GC期间)从软件执行环境获得的堆使用量度量。此外，软件执行环境可以与包括一个或多个虚拟机(例如，JVM)的生产环境对应，并且一个或多个进程可以支持一个或多个云服务。

[0381] 在步骤906处，实施例确定该一个或多个进程在该时间长度期间的线程信息。在一些实施例中，对于从经分析的线程转储确定的一个或多个线程类中的每一个，线程信息可以包括针对多个间隔中的每个间隔的线程强度季节因子。

[0382] 在一些实施例中，堆信息可以包括针对多个间隔中的每个间隔的堆使用量季节因子。具体而言，时间长度可以跨越具有第一长度的第一时段(例如，工作日时段)的一个或多个周期和具有第二长度的第二时段(例如，周末时段)的一个或多个周期。每个时段可以被分割成多个间隔。例如，工作日时段可以被分割成96个15分钟的间隔，而周末时段可以被分割成192个15分钟的间隔。

[0383] 应该注意的是，多个间隔中的每个间隔可以被映射到时段之一的特定季节(即，季节索引)。对于每个季节索引，一些实施例可以确定堆使用量季节因子，并且对于所确定的每个线程类，确定线程强度季节因子，其可以导致每个间隔与堆使用量季节因子和多个线程强度季节因子(每个线程类一个线程强度季节因子)相关联。例如，假设发现了三个不同的线程类，那么工作日时段可以具有96个堆使用量季节因子和288个线程强度季节因子(三个线程类中每个线程类有96个线程强度季节因子)，而周末时段可以具有192个堆使用量季节因子和576个线程强度季节因子。

[0384] 在步骤908处，实施例将堆信息与线程信息相关，以识别该一个或多个进程的与超

过阈值的堆使用量对应的一行或多行代码。下面关于图10更详细地讨论将堆信息与线程信息相关的步骤。

[0385] 在步骤910处，响应于识别出该一行或多行代码，实施例发起与该一行或多行代码相关联的一个或多个动作。例如，实施例可以生成与该一行或多行代码相关联的警报，其被发送给相关人员或代码优化工具。作为响应，可以调查和/或优化识别出的代码行。替代地，一些实施例可以优化这一行或多行代码从而以更高效的方式使用堆存储器。

[0386] 图10图示了根据一些实施例的用于计算各线程类和高的堆使用量之间的相关程度的过程的流程图1000。在一些实施例中，流程图1000中描绘的过程可以由具有一个或多个处理器(例如，图17的计算机系统1700)的计算机系统来实现，其中该一个或多个处理器可以基于存储在计算机可读介质中的计算机代码来执行步骤。图10中描述的步骤可以以任何顺序执行，并且可以有或没有任何其它步骤。

[0387] 流程图1000开始于步骤1002，其中实施例获得一个或多个进程的一个或多个线程转储。如上所述，控制系统可以周期性地使软件执行环境进行线程转储，其中每个线程转储包括由在软件执行环境内执行的一个或多个进程产生的线程的一个或多个栈踪迹。

[0388] 在步骤1004处，实施例通过从一个或多个线程转储接收一个或多个线程并基于与接收到的线程对应的栈踪迹对每个接收到的线程进行归类来获得一个或多个线程类。一旦已经接收并处理所有线程转储，实施例就可以分析一个或多个线程类中的每一个，以在步骤1006-1016中确定每个线程类和高的堆使用量之间的相关程度。

[0389] 在决策1006处，实施例确定是否存在一个或多个线程类的用于确定与高的堆使用量的相关程度的另一个线程类。如果是，则实施例可以进行到步骤1008。否则，实施例可以进行到步骤1018。

[0390] 在可选步骤1008处，实施例计算多个间隔的堆使用量季节因子的平均值。在步骤1010处，实施例计算线程类和多个间隔的线程强度季节因子的平均值。在可选的步骤1012处，实施例计算多个间隔的堆使用量季节因子的方差。在步骤1014处，实施例计算线程类和多个间隔的线程强度季节因子的方差。在步骤1016处，实施例计算线程类和超过阈值的堆使用量之间的相关程度。

[0391] 在步骤1018处，实施例从一个或多个线程类中选择与超过阈值的堆使用量具有最高相关程度的给定的线程类。具体而言，一旦针对每个线程类计算出相关程度，一些实施例就可以通过线程类与堆使用量季节性趋势的相关程度来对线程类进行排名。然后，可以选择高排名的线程类作为给定的线程类。

[0392] 在步骤1020处，实施例基于这些给定的线程类识别可能对高的堆使用量有显著贡献的一行或多行代码。具体而言，一些实施例然后可以分析由栈踪迹指定的文件名和行，以定位与堆囤积的线程类相关联的代码行。应该注意的是，一个或多个进程的属于给定线程类的每个线程执行该一行或多行代码。

[0393] VI.克服预测中的弱外生性和异方差性

[0394] 如上所述，为了确保样本准确性，可以在以不规则时间间隔发生的完全垃圾收集(GC)周期期间进行堆分配测量。在堆使用量特别高的情况下，由于持续的垃圾收集，采样间隔可能变得任意接近于零。因此，基于堆分配测量的时间系列数据可能表现出弱外生性(其中生成残差的过程在某种程度上取决于生成完全GC样本的时间间隔的过程)，以及异方差

性(其中残差的方差随时间不是恒定的)。

[0395] 常规上,生成线性趋势的普通最小二乘回归假定预测器变量和响应变量是由既外生(exogenous)又同方差(homoscedastic)的过程生成的。然而,关于基于在完全GC期间进行的测量的数据集,预测器变量(即,不规则时间间隔)和响应变量(即,在完全GC期间进行的堆使用量测量)不是独立的,因为进行完全GC的频率在堆使用量增加时可能增加。一些实施例可以使用鲁棒和抗性回归方法来克服数据集的弱外生性和异方差性。

[0396] 某些实施例可以利用鲁棒最小二乘回归来克服在这些数据集中表现出的弱外生性和异方差性。具体而言,一些实施例可以(1)将测量的时间系列分解为去季节化度量分量(即,去季节化分量)和季节因子分量(即,季节性效应因子),(2)对去季节化度量分量应用鲁棒线性回归,(3)对季节因子分量应用平滑样条过滤器,以及(4)将线性回归线和经平滑的季节因子重构为季节性和线性趋势模型。

[0397] 最小修剪方块(LTS)估计器是抵抗异常值的影响的鲁棒的回归技术。给定一组N个样本,LTS估计器通过将与最大平方残差对应的50%样本修剪为异常值来最小化最小50%平方残差的总和。LTS估计器运行所有N个样本的普通最小二乘回归的一次迭代以对残差进行排序来选择最小的N/2个残差(即,修剪的样本)。然后,LTS估计器通过更新修剪的样本来迭代地重新运行回归,以减小平方残差的平均值。然而,与下面描述的某些实施例相比,LTS算法的时间复杂度可能相对高。

[0398] 广义加权最小二乘(WLS)估计器是将每个样本的平方误差残差乘以与样本的方差成反比的权重的鲁棒的回归技术。采用WLS估计器可以取决于根据数据的先验知识确定的权重。例如,先验知识可以指定(1)用于测量不同采样点的不同工具的准确度,(2)与相同时刻对应的冗余测量值之间的方差,或(3)测量的最近邻居组之间的方差。如果权重不能通过先验知识来确定,则WLS估计器可以运行普通最小二乘回归的一次迭代来估计残差并使用残差的逆作为权重来迭代地重新运行回归以产生对线性模型的稳定估计。然而,与下面描述的某些实施例相比,WLS算法的时间复杂度相对高。

[0399] 在专利申请14/109,546(该专利通过引用并入本文,用于所有目的)中,公开了用于过滤度量的变化速率$r_{t_n}$的一组等式。该过滤器监视度量的趋势:

[0400] $$r_{t_n} = \frac{x_{t_n} - x_{t_{n-1}}}{t_n - t'_{n-1}}$$

[0401] $$\bar{R}_{t_n} = v_{t_n} r_{t_n} + (1 - v_{t_n}) \bar{G}_{t_n}$$

[0402] 由于变化速率$r_{t_n} = \frac{x_{t_n} - x_{t_{n-1}}}{t_n - t'_{n-1}}$涉及除以时间间隔的长度($t_n - t'_{n-1}$),因此一些实施例可以调整过滤器参数以在时间间隔的长度($t_n - t'_{n-1}$)相对短时给予样本相对小的权重。

[0403] 过滤器参数$v_{t_n}$在以下等式中通过调整因子$\sigma_n^{n-1}$进行调整:

[0404] $$v_{t_n} = \frac{v_{t_{n-1}}}{v_{t_{n-1}} + \sigma_n^{n-1} b_n}$$

[0405] $$b_n = (1 - \beta)^{(t_n - t_{n-1})}$$

[0406] $$\sigma_n^{n-1} = \left(\frac{t_{n-1} - t'_{n-2}}{t_n - t'_{n-1}}\right)$$

[0407] 速率过滤器参数用于过滤经平滑的变化速率，如下所示。如果不采用季节性趋势，那么一些实施例可以使用值$r'_{t_n}$来更新平均值，如下面的公式中所示：

[0408] $$\bar{R}_{t_n} = v_{t_n} r'_{t_n} + (1 - v_{t_n})\bar{G}_{t_n}$$

[0409] 另一方面，如果采用季节性趋势，那么一些实施例可以取决于时间落在周末还是落在工作日时段来使用以下公式之一，其中$\bar{B}_{\tau_n}$和$\bar{C}_{\tau_n}$分别是周末和工作日时段的季节因子。

[0410] $$\Delta x'_{t_n} = \begin{cases} \frac{x'_{t_n}}{\bar{B}_{\tau_n}} - \frac{x_{t_{n-1}}}{\bar{B}_{\tau_{n-1}}}, & t_{n-1} \text{和} t_n \text{落在周末季节} \\ \frac{x'_{t_n}}{\bar{C}_{\tau_n}} - \frac{x_{t_{n-1}}}{\bar{C}_{\tau_{n-1}}}, & t_{n-1} \text{和} t_n \text{落在工作日季节} \\ \frac{x'_{t_n}}{\bar{B}_{\tau_n}} - \frac{x_{t_{n-1}}}{\bar{C}_{\tau_{n-1}}}, & t_{n-1} \text{落在周末并且} t_n \text{落在周末} \\ \frac{x'_{t_n}}{\bar{C}_{\tau_n}} - \frac{x_{t_{n-1}}}{\bar{B}_{\tau_{n-1}}}, & t_{n-1} \text{落在周末并且} t_n \text{落在工作日} \end{cases}$$

[0411] 接下来，一些实施例可以使用以下公式确定去季节化的原始增长率：

[0412] $$r'_{t_n} = \frac{\Delta x'_{t_n}}{t_n - t'_{n-1}}$$

[0413] 然后，一些实施例可以使用以下公式来更新移动平均值：

[0414] $$\bar{R}_{t_n} = v_{t_n} r'_{t_n} + (1 - v_{t_n})\bar{G}_{t_n}$$

[0415] 具体而言，由上述等式生成的速率过滤器参数$v_{t_n}$表示权重，该权重基于在特定样本与紧接在该特定样本之前的另一个样本之间发生的时间间隔的长度。速率过滤器参数与时间系列中的测量数据之间存在一对一的对应关系。图11描绘了针对示例数据集跨整个时间范围对采样时间间隔绘制过滤器参数$v_{t_n}$的图表。虽然时间范围被划分为6个重叠的子范围，但是每个子范围中的图表显示样本时间间隔和过滤器参数之间存在线性关系。如从图中可以看到的，当采样时间间隔小时，过滤器参数(即，样本的权重)小。该调整取决于当前采样点周围的采样点的密度动态地减少过滤器中的样本的权重。

[0416] 一些实施例使用速率过滤器参数来修剪数据点。修剪数据点可以有助于跨整个时间范围均匀采样点的密度，从而提高线性回归算法的鲁棒性。关于表示在完全GC周期期间软件执行环境中的堆使用量的测量的数据点，靠近在一起的数据点可以与其中更经常执行完全GC的较高的堆使用量的时段(例如，负载尖峰期间)对应。

[0417] 一些实施例将速率过滤器参数与阈值进行比较，并且如果速率过滤器参数小于阈值，则从鲁棒线性回归中排除(即，修剪)对应的数据点。一些实施例可以使用速率过滤器参

数的中值或均值作为阈值。具体而言，一些实施例可以修剪靠近在一起的数据点，因为这样的数据点可能表示负载浪涌或异常值。因此，一些实施例可以通过沿着时间轴均匀数据点的密度来减轻弱外生性状况，其减小了不规则时间间隔和残差之间的相关性。

[0418] 在专利申请14/109,546中公开了通过以下等式生成的针对预测误差残差的时间系列$\bar{D}_{t_n}$和针对预测度量的时间系列$\bar{F}_{t_n}$，该专利申请通过引用并入本文，用于所有目的。

[0419] $$\bar{F}_{t_n} = \begin{cases} \bar{X}_{t_{n-1}} + \bar{M}_{t_{n-1}}(t_n - t_{n-1}) \\ \left[\bar{X}_{t_{n-1}} + \bar{M}_{t_{n-1}}(t_n - t_{n-1})\right]\bar{C}_{\tau_n - L} \\ x_{t_{n-1}} + \bar{R}_{t_{n-1}}(t_n - t_{n-1}) \end{cases}$$

[0420] 一些实施例可以使用以下公式生成预测度量的误差残差：

[0421] $$e_{t_n} = \bar{F}_{t_n} - x_{t_n}$$

[0422] $$\bar{E}_{t_n} = \psi_{t_n} e_{t_n} + (1 - \psi_{t_n})\bar{E}_{t_{n-1}}$$

[0423] $$\bar{D}_{t_n} = \psi_{t_n}|e_{t_n}| + (1 - \psi_{t_n})\bar{D}_{t_{n-1}}$$

[0424] 因为过滤器生成的经平滑的绝对误差残差$\bar{D}_{t_n}$与最小二乘回归的残差的方差之间存在相关性，因此一些实施例可以使用经平滑的绝对误差残差$\mathbf{1}/\bar{D}_{t_n}$的逆作为广义加权最小二乘回归的权重。在这样做时，一些实施例可以通过向具有与期望值的相对大的偏差的样本值赋予相对小的权重来减轻异方差性状况。该期望值可以表示近邻组样本的卷积。

[0425] 以下示例代码(用R编程语言编写)示出了如何可以计算修剪的样本子集和样本的权重。如下面的示例代码所示，一些实施例可以使用R函数“rlm”，该函数使得某些实施例能够指定修剪的样本子集和样本的权重以用于生成加权最小二乘回归。应该注意的是，示例代码中的rateFilterParameter、seasonalFactor、absoluteErrorResidual、度量和时间向量是具有相同时间范围的时间系列。

[0426]
```
trimmingParameter <- c(rateFilterParameter, which(normSeasonalFactor < 1.0))

threshold1 <- median(trimmingParameter, na.rm = TRUE)
threshold2 <- median(rateFilterParameter, na.rm = TRUE)
threshold3 <- mean(rateFilterParameter, na.rm = TRUE)
trimmingThreshold <- max(threshold1, threshold2, threshold3)

lengthOfTimmingParameter <- length(trimmingParameter)

# can set up a list of graduated thresholds to give less weight to older data
```

```
listOfTrimmingThresholds <- c(trimmingThreshold * 1.1, trimmingThreshold * 1.05,
                    trimmingThreshold,
                    trimmingThreshold * 0.95, trimmingThreshold * 0.9)
numberOfTrimmingThresholds <- length(listOfTrimmingThresholds)

# select the exclusion set of the data points which are time close together
# may use graduated thresholds to give less weight to older data for what is
# known as discounted least-squares regression
prevSplitPoint <- 0
for (num in 1:numberOfTrimmingThresholds) {
  splitPoint <- trunc(lengthOfTimmingParameter * num / numberOfTrimmingThresholds)
  excludeIndices <- c(excludeIndices,
          which(rateFilterParameter[(prevSplitPoint + 1):splitPoint] <
              listOfTrimmingThresholds[num]) + prevSplitPoint)
  prevSplitPoint <- splitPoint
}
```

[0427]

```
includeIndices <- 1:length(rateFilterParameter)
includeIndices <- includeIndices[-excludeIndices]

# use the inverse absolute error residual for weights
minErrorResidual <- min(absoluteErrorResidual)
if (minErrorResidual == 0) {
  absoluteErrorResidual[absoluteErrorResidual == 0] <- 1.0
  minErrorResidual <- 1.0
}
weights <- (minErrorResidual / absoluteErrorResidual)

# use regression methods robust to heteroscedasticity and weak-exogeneity
# use trimmed indices "includeIndices" to compensate for the weak-exogeneity
# use weights to compensate for the heteroscedasticity
# in generalized weighted least-squares
linear <- tryCatch(rlm(measure ~ time, weights = weights,
                method = "M", subset = includeIndices),
                error = function(e) return(null))
```

[0428] 针对与数据点的时间戳对应的每个时间戳$t_n$，速率过滤器参数被给出为由$v_{t_n}$表示的值的时间系列。如果$v_{t_n} < z$，其中z是阈值，则从线性回归中排除对应的数据点$x_{t_n}$。通常，一些实施例可以使用速率过滤器参数的N百分位数处的任何值(例如，作为50百分位数的中值)作为阈值z。

[0429] 在一些实施例中，针对与数据点的时间戳对应的每个时间戳$t_n$，绝对误差残差被给出为时间系列$\overline{D}_{t_n}$。时间戳$t_n$处的样本的权重$W_{t_n}$可以与$\overline{D}_{t_n}$成反比。一些实施例可以补偿表示短期负载浪涌或异常值的数据点之间的方差变化。

[0430] 为了减少堆使用量中的异常值和短期浪涌对线性回归的影响，一些实施例可以将均匀数据点的密度的技术与指派较小权重给偏离样本值的技术相组合。在这样做时，一些实施例可以增加线性回归的鲁棒性，这可以有助于捕获(例如，堆使用量的)长期趋势。应该注意的是，一起使用这两种技术可以提供线性回归线与数据的更好拟合，并且可以比使用通常涉及几次回归迭代的常规的LTS估计器或WLS估计器更高效。

[0431] 为了进一步提高回归的鲁棒性，一些实施例可以附加地识别瞬态状态并移除落入瞬态状态的采样点以及移除作为异常值的运行到运行(run-to-run)段(例如，与经历存储器泄漏、存储器不足事件或非常高的增长率的软件执行环境对应的数据段)。

[0432] 图12显示了三个趋势图，每个趋势图通过针对生产环境中的堆使用量的不同线性回归技术导出。蓝色趋势线1205可以通过标准线性回归算法导出，该算法为每个采样点指派相等的权重。棕色趋势线1210可以通过常规的鲁棒回归算法导出。红色线1215表示由上述本实施例提供的回归，其靠近棕色趋势线。

[0433] 图13显示了附加图表，其图示了常规回归技术如何会提供不正确的结果。如该图中所示，表示常规回归技术的棕色趋势线1305与两个高密度样本点簇紧密拟合。作为对照，红色线1215正确地跟踪样本点中的趋势，以提供软件执行环境中的堆使用量的长期预测。

[0434] 图14图示了根据一些实施例的用于生成信号的预测的过程的流程图1400。在一些实施例中，流程图1400中描绘的过程可以由具有一个或多个处理器(例如，图17的计算机系统1700)的计算机系统来实现，其中该一个或多个处理器可以基于存储在计算机可读介质中的计算机代码来执行步骤。图14中描述的步骤可以以任何顺序执行，并且可以有或没有任何其它步骤。

[0435] 流程图1400开始于步骤1402，其中实施例接收包括在一时间跨度内从正在其中执行一个或多个进程的环境采样的多个度量的信号。在一些实施例中，该多个度量可以是由正在监视软件执行环境(例如，生产环境)内的堆使用量的控制系统获取的堆使用量测量值，其中软件执行环境包括一个或多个执行进程。

[0436] 在步骤1404处，实施例从信号1404提取季节性效应因子和去季节化分量。在一些实施例中，季节性效应因子可以与针对指派给数据集的时段的每个间隔确定的季节因子对应。在一些实施例中，可以通过将季节因子应用于信号来获得去季节化分量。

[0437] 在步骤1406处，实施例将一个或多个样条函数应用于季节性效应因子以生成第一模型。在这点上，一些实施例可以对与预期值急剧偏离的样本值赋予相对小的权重，其中预期值表示近邻样本组的卷积。

[0438] 在步骤1408处，实施例将线性回归技术应用于去季节化分量以生成第二模型。具体而言，为了补偿在高的堆使用量期间经历的相对短的时间间隔，一些实施例可以调整过滤器参数以赋予在短间隔期间获取的样本相对小的权重。一些实施例可以使用速率过滤器参数来修剪数据集中包括的数据点。修剪数据点可以有助于跨整个时间范围均匀采样点的密度，从而提高线性回归算法的鲁棒性。

[0439] 在步骤1410处，实施例基于第一模型和第二模型生成信号的预测。在一些实施例中，信号的预测可以与使用步骤1406和1408中描述的技术生成的回归线对应。具体而言，所生成的预测可以更好地拟合信号。

[0440] 在步骤1412处，实施例至少部分地基于预测来发起与环境相关联的一个或多个动作。例如，如果预测指示堆使用量将来会增加，则一些实施例可以将附加资源(例如，存储器、RAM)分配给软件执行环境。

[0441] 图15描绘了用于实现实施例的分布式系统1500的简化图。在所示实施例中，分布式系统1500包括一个或多个客户端计算设备1502、1504、1506和1508，这些客户端计算设备被配置为通过(一个或多个)网络1510执行和操作客户端应用，诸如web浏览器、专有客户端

(例如,Oracle Forms)等等。服务器1512可以经由网络1510与远程客户端计算设备1502、1504、1506和1508通信地耦合。

[0442] 在各种实施例中,服务器1512可以适于运行一个或多个服务或软件应用。在某些实施例中,服务器1512还可以提供其它服务,或者软件应用可以包括非虚拟和虚拟环境。在一些实施例中,这些服务可以作为基于web的或云服务或者在软件即服务(SaaS)模型下提供给客户端计算设备1502、1504、1506和/或1508的用户。操作客户端计算设备1502、1504、1506和/或1508的用户又可以利用一个或多个客户端应用与服务器1512交互以利用由这些部件提供的服务。

[0443] 在图15中描绘的配置中,系统1500的软件部件1518、1520和1522被示出为在服务器1512上实现。在其它实施例中,系统1500的一个或多个部件和/或由这些部件提供的服务可以也可以由客户端计算设备1502、1504、1506和/或1508中的一个或多个来实现。然后,操作客户端计算设备的用户可以利用一个或多个客户端应用来使用由这些部件提供的服务。这些部件可以用硬件、固件、软件或其组合来实现。应该理解的是,各种不同的系统配置是可能的,其可以与分布式系统1500不同。因此,图15中所示的实施例是用于实现实施例系统的分布式系统的一个示例,并且不旨在是限制性的。

[0444] 客户端计算设备1502、1504、1506和/或1508可以包括各种类型的计算系统。例如,客户端计算设备可以包括便携式手持设备(例如,**iPhone®**、蜂窝电话、**iPad®**、计算平板、个人数字助理(PDA))或可穿戴设备(例如,Google**Glass®**头戴式显示器),其运行诸如Microsoft Windows**Mobile®**之类的软件和/或诸如iOS、Windows Phone、Android、BlackBerry 10,Palm OS之类的各种移动操作系统。设备可以支持各种应用(诸如各种互联网相关的应用、电子邮件、短消息服务(SMS)应用),并且可以使用各种其它通信协议。客户端计算设备还可以包括通用个人计算机,作为示例,运行各种版本的Microsoft **Windows®**、Apple**Macintosh®**和/或Linux操作系统的个人计算机和/或膝上型计算机。客户端计算设备可以是运行任何各种商用的**UNIX®**或类UNIX操作系统(包括但不限于诸如像Google Chrome OS的各种GNU/Linux操作系统)的工作站计算机。客户端计算设备还可以包括能够提供(一个或多个)网络1510通信的电子设备(诸如瘦客户端计算机、启用互联网的游戏系统(例如,具有或不具有**Kinect®**手势输入设备的Microsoft**Xbox®**游戏控制台)和/或个人消息传送设备)。

[0445] 虽然图15中的分布式系统1500被示出具有四个客户端计算设备,但是可以支持任何数量的客户端计算设备。其它设备(例如具有传感器的设备等)可以与服务器1512交互。

[0446] 分布式系统1500中的(一个或多个)网络1510可以是对本领域技术人员熟悉的可以利用任何各种可用协议支持数据通信的任何类型的网络,其中各种协议包括但不限于TCP/IP(传输控制协议/互联网协议)、SNA(系统网络体系架构)、IPX(互联网分组交换)、AppleTalk等。仅仅作为示例,(一个或多个)网络1510可以是局域网(LAN)、基于以太网的网络、令牌环、广域网、互联网、虚拟网络、虚拟专用网络(VPN)、内联网、外联网、公共交换电话网络(PSTN)、红外网络、无线网络(例如,在任何电气和电子协会(IEEE)802.11协议套件、**Bluetooth®**、和/或任何其它无线协议下操作的网络)和/或这些和/或其它网络的任意

组合。

[0447] 服务器1512可以由一个或多个通用计算机、专用服务器计算机(作为示例,包括PC(个人计算机)服务器、UNIX®服务器、中档服务器、大型计算机、机架安装的服务器等)、服务器场、服务器集群或任何其它适当的布置和/或组合组成。服务器1512可以包括运行虚拟操作系统的一个或多个虚拟机,或涉及虚拟化的其它计算体系架构。一个或多个灵活的逻辑存储设备池可以被虚拟化,以维护用于服务器的虚拟存储设备。虚拟网络可以由服务器1512利用软件定义的联网来控制。在各种实施例中,服务器1512可以适于运行在前述公开内容中描述的一个或多个服务或软件应用。例如,服务器1512可以与根据本公开的实施例的用于如上所述执行处理的服务器对应。

[0448] 服务器1512可以运行包括以上讨论的任何操作系统的操作系统,以及任何商用的服务器操作系统。服务器1512还可以运行任何各种附加的服务器应用和/或中间层应用,包括HTTP(超文本传输协议)服务器、FTP(文件传输协议)服务器、CGI(公共网关接口)服务器、JAVA®服务器、数据库服务器等。示例性数据库服务器包括但不限于可从Oracle、Microsoft、Sybase、IBM(国际商业机器)等商业获得的数据库服务器。

[0449] 在一些实现中,服务器1512可以包括一个或多个应用,以分析和整合从客户端计算设备1502、1504、1506和1508的用户接收到的数据馈送和/或事件更新。作为示例,数据馈送和/或事件更新可以包括但不限于从一个或多个第三方信息源和持续数据流接收到的Twitter®馈送、Facebook®更新或实时更新,其可以包括与传感器数据应用、金融报价机、网络性能测量工具(例如,网络监视和流量管理应用)、点击流分析工具、汽车流量监视等相关的实时事件。服务器1512还可以包括经由客户端计算设备1502、1504、1506和1508的一个或多个显示设备显示数据馈送和/或实时事件的一个或多个应用。

[0450] 分布式系统1500还可以包括一个或多个数据库1514和1516。这些数据库可以提供用于存储信息(诸如用户交互信息、使用模式信息、适应规则信息和由本公开的实施例使用的其它信息)的机制。数据库1514和1516可以驻留在各种位置中。举例来说,数据库1514和1516中的一个或多个可以驻留在服务器1512本地(和/或驻留在服务器1512中)的非瞬态存储介质上。可替代地,数据库1514和1516可以远离服务器1512并经由基于网络的或专用的连接与服务器1512通信。在一组实施例中,数据库1514和1516可以驻留在存储区域网络(SAN)中。类似地,用于执行归属于服务器1512的功能的任何必要文件可以适当地本地存储在服务器1512上和/或远程存储。在一组实施例中,数据库1514和1516可以包括适于响应于SQL格式的命令来存储、更新和检索数据的关系数据库,诸如由Oracle提供的数据库。

[0451] 在一些实施例中,云环境可以提供一个或多个服务。图16是根据本公开的实施例的、其中服务可以作为云服务被提供的系统环境1600的一个或多个部件的简化框图。在图16中所示的实施例中,系统环境1600包括可以被用户用来与提供云服务的云基础设施系统1602交互的一个或多个客户端计算设备1604、1606和1608。云基础设施系统1602可以包括一个或多个计算机和/或服务器,其可以包括上面针对服务器1612所描述的那些。

[0452] 应该认识到的是,图16中描绘的云基础设施系统1602可以具有除了所描绘的那些之外的其它部件。另外,图16中所示的实施例仅仅是可以结合本公开的实施例的云基础设施系统的一个示例。在一些其它实施例中,云基础设施系统1602可以具有比图中所示更多

或更少的部件、可以组合两个或更多个部件，或者可以具有不同的部件配置或布置。

[0453] 客户端计算设备1604、1606和1608可以是与上述类似的设备。客户端计算设备1604、1606和1608可以被配置为操作客户端应用，诸如web浏览器、专有客户端应用(例如，Oracle Forms)或者可以由客户端计算设备的用户使用以与云基础设施系统1602交互来使用由云基础设施系统1602提供的服务的一些其它应用。虽然示例性系统环境1600被示为具有三个客户端计算设备，但是可以支持任何数量的客户端计算设备。诸如具有传感器的设备等的其它设备可以与云基础设施系统1602交互。

[0454] (一个或多个)网络1610可以促进客户端计算设备1604、1606和1608与云基础设施系统1602之间的数据通信和交换。每个网络可以是对本领域技术人员熟悉的可以利用任何各种商用的协议支持数据通信的任何类型的网络，其中协议包括以上针对(一个或多个)网络1610所描述的协议。

[0455] 在某些实施例中，由云基础设施系统1602提供的服务可以包括按需对云基础设施系统的用户可用的一系列服务。也可以提供各种其它服务，包括但不限于在线数据存储和备份解决方案、基于Web的电子邮件服务、托管的办公套件和文档协作服务、数据库处理、受管理的技术支持服务等。由云基础架构系统提供的服务可以动态扩展以满足其用户的需求。

[0456] 在某些实施例中，由云基础设施系统1602提供的服务的具体实例化在本文中可以被称为“服务实例”。一般而言，经由通信网络(诸如互联网)从云服务提供者的系统使得对用户可用的任何服务被称为“云服务”。通常，在公共云环境中，构成云服务提供者的系统的服务器和系统与客户自己的本地服务器和系统不同。例如，云服务提供者的系统可以托管应用，并且用户可以经由诸如互联网的通信网络按需订购和使用应用。

[0457] 在一些示例中，计算机网络云基础设施中的服务可以包括对存储装置、托管的数据库、托管的web服务器、软件应用或者由云供应商向用户提供的其它服务的受保护的计算机网络访问，或者如本领域中另外已知的。例如，服务可以包括通过互联网对云上的远程存储的受密码保护的访问。作为另一个示例，服务可以包括基于web服务的托管的关系数据库和脚本语言中间件引擎，用于由联网的开发人员私人使用。作为另一个示例，服务可以包括对在云供应商的网站上托管的电子邮件软件应用的访问。

[0458] 在某些实施例中，云基础设施系统1602可以包括以自助服务、基于订阅、弹性可扩展、可靠、高度可用和安全的方式交付给客户的应用套件、中间件和数据库服务产品。这种云基础设施系统的示例是由本受让人提供的Oracle Public Cloud(Oracle公共云)。

[0459] 云基础设施系统1602还可以提供与“大数据”相关的计算和分析服务。术语“大数据”一般用来指可由分析员和研究者存储和操纵以可视化大量数据、检测趋势和/或以其它方式与数据交互的极大数据集。这种大数据和相关应用可以在许多级别和不同规模上由基础设施系统托管和/或操纵。并行链接的数十个、数百个或数千个处理器可以作用于这种数据，以便呈现其或者模拟对数据或其所表示的内容的外力。这些数据集可以涉及结构化数据，诸如在数据库中组织或以其它方式根据结构化模型组织的数据，和/或者非结构化数据(例如，电子邮件、图像、数据blob(二进制大对象)、网页、复杂事件处理)。通过利用实施例相对快速地将更多(或更少)的计算资源聚焦在目标上的能力，云基础设施系统可以更好地用于基于来自企业、政府机构、研究组织、私人个人、一群志同道合的个人或组织或其它实

体的需求在大数据集上执行任务。

[0460] 在各种实施例中，云基础设施系统1602可以适于自动地供应、管理和跟踪客户对由云基础设施系统1602提供的服务的订阅。云基础设施系统1602可以经由不同的部署模型提供云服务。例如，服务可以在公共云模型下提供，其中云基础设施系统1602由销售云服务的组织拥有(例如，由Oracle公司拥有)并且使服务对一般公众或不同的工业企业可用。作为另一个示例，服务可以在私有云模型下提供，其中云基础设施系统1602仅针对单个组织操作，并且可以为组织内的一个或多个实体提供服务。云服务还可以在社区云模型下提供，其中云基础设施系统1602和由云基础设施系统1602提供的服务由相关社区中的若干个组织共享。云服务还可以在混合云模型下提供，混合云模型是两个或更多个不同模型的组合。

[0461] 在一些实施例中，由云基础设施系统1602提供的服务可以包括在软件即服务(SaaS)类别、平台即服务(PaaS)类别、基础设施即服务(IaaS)类别、或包括混合服务的服务的其它类别下提供的一个或多个服务。客户经由订阅订单可以订购由云基础设施系统1602提供的一个或多个服务。云基础设施系统1602然后执行处理，以提供客户的订阅订单中的服务。

[0462] 在一些实施例中，由云基础设施系统1602提供的服务可以包括但不限于应用服务、平台服务和基础设施服务。在一些示例中，应用服务可以由云基础设施系统经由SaaS平台提供。SaaS平台可以被配置为提供属于SaaS类别的云服务。例如，SaaS平台可以提供在集成的开发和部署平台上构建和交付点播应用套件的能力。SaaS平台可以管理和控制用于提供SaaS服务的底层软件和基础设施。通过利用由SaaS平台提供的服务，客户可以利用在云基础设施系统上执行的应用。客户可以获取应用服务，而无需客户单独购买许可证和支持。可以提供各种不同的SaaS服务。示例包括但不限于为大型组织提供用于销售绩效管理、企业集成和业务灵活性的解决方案的服务。

[0463] 在一些实施例中，平台服务可以由云基础设施系统1602经由PaaS平台提供。PaaS平台可以被配置为提供属于PaaS类别的云服务。平台服务的示例可以包括但不限于使组织(诸如Oracle)能够在共享的共同体系架构上整合现有应用的服务，以及利用由平台提供的共享服务构建新应用的能力。PaaS平台可以管理和控制用于提供PaaS服务的底层软件和基础设施。客户可以获取由云基础设施系统1602提供的PaaS服务，而无需客户购买单独的许可证和支持。平台服务的示例包括但不限于Oracle Java云服务(JCS)、Oracle数据库云服务(DBCS)以及其它。

[0464] 通过利用由PaaS平台提供的服务，客户可以采用由云基础设施系统支持的编程语言和工具，并且还控制所部署的服务。在一些实施例中，由云基础设施系统提供的平台服务可以包括数据库云服务、中间件云服务(例如，Oracle Fusion Middleware服务)和Java云服务。在一个实施例中，数据库云服务可以支持共享服务部署模型，其使得组织能够汇集数据库资源并且以数据库云的形式向客户提供数据库即服务。中间件云服务可以为客户提供开发和部署各种业务应用的平台，以及Java云服务可以在云基础设施系统中为客户提供部署Java应用的平台。

[0465] 可以由云基础设施系统中的IaaS平台提供各种不同的基础设施服务。基础设施服务促进底层计算资源(诸如存储装置、网络和其它基本计算资源)的管理和控制，以便客户利用由SaaS平台和PaaS平台提供的服务。

[0466] 在某些实施例中，云基础设施系统1602还可以包括基础设施资源1630，用于提供用来向云基础设施系统的客户提供各种服务的资源。在一个实施例中，基础设施资源1630可以包括执行由PaaS平台和SaaS平台提供的服务的硬件(诸如服务器、存储装置和联网资源)的预先集成和优化的组合，以及其它资源。

[0467] 在一些实施例中，云基础设施系统1602中的资源可以由多个用户共享并且按需动态地重新分配。此外，资源可以分配给在不同时区中的用户。例如，云基础设施系统1602可以使第一时区内的第一用户集合能够利用云基础设施系统的资源指定的小时数，然后使得能够将相同资源重新分配给位于不同时区中的另一用户集合，从而最大化资源的利用率。

[0468] 在某些实施例中，可以提供由云基础设施系统1602的不同部件或模块共享，以使得能够由云基础设施系统1602供应服务的多个内部共享服务1632。这些内部共享服务可以包括但不限于安全和身份服务、集成服务、企业储存库服务、企业管理器服务、病毒扫描和白名单服务、高可用性、备份和恢复服务、用于启用云支持的服务、电子邮件服务、通知服务、文件传输服务等。

[0469] 在某些实施例中，云基础设施系统1602可以在云基础设施系统中提供云服务(例如，SaaS、PaaS和IaaS服务)的综合管理。在一个实施例中，云管理功能可以包括用于供应、管理和跟踪由云基础设施系统1602等接收到的客户的订阅的能力。

[0470] 在一个实施例中，如图16中所绘出的，云管理功能可以由诸如订单管理模块1620、订单编排模块1622、订单供应模块1624、订单管理和监视模块1626以及身份管理模块1628的一个或多个模块提供。这些模块可以包括或可以利用一个或多个计算机和/或服务器提供，该一个或多个计算机和/或服务器可以是通用计算机、专用服务器计算机、服务器场，服务器集群或任何其它适当的布置和/或组合。

[0471] 在示例性操作中，在步骤1634处，使用客户端设备(诸如客户端计算设备1604、1606或1608)的客户可以通过请求由云基础设施系统1602提供的一个或多个服务并且对由云基础设施系统1602提供的一个或多个服务的订阅下订单来与云基础设施系统1602交互。在某些实施例中，客户可以访问诸如云UI 1612、云UI 1614和/或云UI 1616的云用户界面(UI)并经由这些UI下订阅订单。响应于客户下订单而由云基础设施系统1602接收到的订单信息可以包括识别客户和客户打算订阅的由云基础设施系统1602提供的一个或多个服务的信息。

[0472] 在步骤1636处，从客户接收到的订单信息可以存储在订单数据库1618中。如果这是新的订单，则可以为该订单创建新的记录。在一个实施例中，订单数据库1618可以是由云基础设施系统1618操作以及与其它系统元素结合操作的若干数据库当中的一个。

[0473] 在步骤1638处，订单信息可以被转发到订单管理模块1620，订单管理模块1620可以被配置为执行与订单相关的计费和记帐功能，诸如验证订单，并且在通过验证时预订订单。

[0474] 在步骤1640处，关于订单的信息可以被传送到订单编排模块1622，订单编排模块1622被配置为编排用于由客户下的订单的服务和资源的供应。在一些情况下，订单编排模块1622可以使用订单供应模块1624的服务用于供应。在某些实施例中，订单编排模块1622使得能够管理与每个订单相关联的业务过程，并且应用业务逻辑来确定订单是否应当继续供应。

[0475] 如图16中绘出的实施例所示，在步骤1642处，在接收到新订阅的订单时，订单编排模块1622向订单供应模块1624发送分配资源和配置履行订购订单所需的资源的请求。订单供应模块1624使得能够为由客户订购的服务分配资源。订单供应模块1624提供由云基础设施系统1600提供的云服务和用来供应用于提供所请求的服务的资源的物理实现层之间的抽象级别。这使得订单编排模块1622能够与实现细节隔离，诸如服务和资源是否实际上实时供应，或者预先供应并且仅在请求时才进行分配/指定。

[0476] 在步骤1644处，一旦供应了服务和资源，就可以向订阅的客户发送指示所请求的服务现在已准备好用于使用的通知。在一些情况下，可以向客户发送使得客户能够开始使用所请求的服务的信息(例如，链接)。

[0477] 在步骤1646处，可以由订单管理和监视模块1626来管理和跟踪客户的订阅订单。在一些情况下，订单管理和监视模块1626可以被配置为收集关于客户使用所订阅的服务的使用统计。例如，可以针对所使用的存储量、所传送的数据量、用户的数量以及系统启动时间和系统停机时间的量等来收集统计数据。

[0478] 在某些实施例中，云基础设施系统1600可以包括身份管理模块1628，其被配置为提供身份服务，诸如云基础设施系统1600中的访问管理和授权服务。在一些实施例中，身份管理模块1628可以控制关于希望利用由云基础设施系统1602提供的服务的客户的信息。这种信息可以包括认证这些客户的身份的信息和描述那些客户被授权相对于各种系统资源(例如，文件、目录、应用、通信端口、存储器段等)执行的动作的信息。身份管理模块1628还可以包括关于每个客户的描述性信息以及关于如何和由谁来访问和修改描述性信息的管理。

[0479] 图17图示了可以用于实现本公开的实施例的示例性计算机系统1700。在一些实施例中，计算机系统1700可以用于实现上述各种服务器和计算机系统中的任何一种。如图17所示，计算机系统1700包括各种子系统，包括经由总线子系统1702与多个外围子系统通信的处理单元1704。这些外围子系统可以包括处理加速单元1706、I/O子系统1708、存储子系统1718和通信子系统1724。存储子系统1718可以包括有形计算机可读存储介质1722和系统存储器1710。

[0480] 总线子系统1702提供用于使计算机系统1700的各种部件和子系统按照期望彼此通信的机制。虽然总线子系统1702被示意性地示为单条总线，但是总线子系统的可替代实施例可以利用多条总线。总线子系统1702可以是若干种类型的总线结构中的任何一种，包括存储器总线或存储器控制器、外围总线和利用任何各种总线体系架构的局部总线。例如，此类体系架构可以包括工业标准体系架构(ISA)总线、微通道体系架构(MCA)总线、增强型ISA(EISA)总线、视频电子标准协会(VESA)局部总线和外围部件互连(PCI)总线，其可以实现为根据IEEE P1386.1标准制造的夹层(Mezzanine)总线，等等。

[0481] 处理子系统1704控制计算机系统1700的操作并且可以包括一个或多个处理单元1732、1734等。处理单元可以包括一个或多个处理器，其中包括单核或多核处理器、处理器的一个或多个核、或其组合。在一些实施例中，处理子系统1704可以包括一个或多个专用协处理器，诸如图形处理器、数字信号处理器(DSP)等。在一些实施例中，处理子系统1704的处理单元中的一些或全部可以利用定制电路来实现，诸如专用集成电路(ASIC)或现场可编程门阵列(FPGA)。

[0482] 在一些实施例中，处理子系统1704中的处理单元可以执行存储在系统存储器1710中或计算机可读存储介质1722上的指令。在各种实施例中，处理单元可以执行各种程序或代码指令，并且可以维护多个并发执行的程序或进程。在任何给定的时间，要执行的程序代码中的一些或全部可以驻留在系统存储器1710中和/或计算机可读存储介质1722上，潜在地包括在一个或多个存储设备上。通过适当的编程，处理子系统1704可以提供各种功能。

[0483] 在某些实施例中，可以提供处理加速单元1706，用于执行定制的处理或用于卸载由处理子系统1704执行的一些处理，以便加速由计算机系统1700执行的整体处理。

[0484] I/O子系统1708可以包括用于向计算机系统1700输入信息和/或用于从或经由计算机系统1700输出信息的设备和机制。一般而言，术语“输入设备”的使用旨在包括用于向计算机系统1700输入信息的所有可能类型的设备和机制。用户接口输入设备可以包括，例如，键盘、诸如鼠标或轨迹球的指示设备、结合到显示器中的触摸板或触摸屏、滚轮、点拨轮、拨盘、按钮、开关、键板、具有语音命令识别系统的音频输入设备、麦克风以及其它类型的输入设备。用户接口输入设备也可以包括使用户能够控制输入设备并与其交互的诸如Microsoft **Kinect®**运动传感器的运动感测和/或姿势识别设备、Microsoft **Xbox®** 360游戏控制器、提供用于接收利用姿势和口语命令的输入的接口的设备。用户接口输入设备也可以包括眼睛姿势识别设备，诸如从用户检测眼睛活动(例如，当拍摄图片和/或进行菜单选择时的“眨眼”)并将眼睛姿势转换为到输入设备(例如，Google **Glass®**)中的输入的Google **Glass®**眨眼检测器。此外，用户接口输入设备可以包括使用户能够通过语音命令与语音识别系统(例如，**Siri®**导航器)交互的语音识别感测设备。

[0485] 用户接口输入设备的其它示例包括但不限于三维(3D)鼠标、操纵杆或指示杆、游戏板和图形平板、以及音频/视频设备，诸如扬声器、数字相机、数字摄像机、便携式媒体播放器、网络摄像机、图像扫描仪、指纹扫描仪、条形码读取器3D扫描仪、3D打印机、激光测距仪、以及眼睛注视跟踪设备。此外，用户接口输入设备可以包括，例如，医疗成像输入设备，诸如计算机断层摄影、磁共振成像、位置发射断层摄影、医疗超声检查设备。用户接口输入设备也可以包括，例如，音频输入设备，诸如MIDI键盘、数字乐器等。

[0486] 用户接口输出设备可以包括显示子系统、指示器灯或诸如音频输出设备的非可视显示器等。显示子系统可以是阴极射线管(CRT)、诸如利用液晶显示器(LCD)或等离子体显示器的平板设备、投影设备、触摸屏等。一般而言，术语“输出设备”的使用旨在包括用于从计算机系统1700向用户或其它计算机输出信息的所有可能类型的设备和机制。例如，用户接口输出设备可以包括但不限于，可视地传达文本、图形和音频/视频信息的各种显示设备，诸如监视器、打印机、扬声器、耳机、汽车导航系统、绘图仪、语音输出设备和调制解调器。

[0487] 存储子系统1718提供用于存储由计算机系统1700使用的信息的储存库或数据存储。存储子系统1718提供有形非瞬态计算机可读存储介质，用于存储提供一些实施例的功能的基本编程和数据结构。当由处理子系统1704执行时提供上述功能的软件(程序、代码模块、指令)可以存储在存储子系统1718中。软件可以由处理子系统1704的一个或多个处理单元执行。存储子系统1718也可以提供用于存储根据本公开使用的数据的储存库。

[0488] 存储子系统1718可以包括一个或多个非瞬态存储器设备，包括易失性和非易失性

存储器设备。如图17所示，存储子系统1718包括系统存储器1710和计算机可读存储介质1722。系统存储器1710可以包括多个存储器，包括用于在程序执行期间存储指令和数据的易失性主随机存取存储器(RAM)和其中存储固定指令的非易失性只读存储器(ROM)或闪存存储器。在一些实现中，包含帮助在诸如启动期间在计算机系统1700内的元件之间传送信息的基本例程的基本输入/输出系统(BIOS)通常可以存储在ROM中。RAM通常包含当前由处理子系统1704操作和执行的数据和/或程序模块。在一些实现中，系统存储器1710可以包括多个不同类型的存储器，诸如静态随机存取存储器(SRAM)或动态随机存取存储器(DRAM)。

[0489] 作为示例而非限制，如在图17中所绘出的，系统存储器1710可以存储应用程序1712，其可以包括客户端应用、Web浏览器、中间层应用、关系数据库管理系统(RDBMS)等、程序数据1714和操作系统1716。作为示例，操作系统1716可以包括各种版本的Microsoft **Windows®**、Apple **Macintosh®**和/或Linux操作系统、各种商用 **UNIX®**或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统、Google **Chrome®** OS等)和/或诸如iOS、**Windows®** Phone、**Android®** OS、**BlackBerry®** 10OS和 **Palm®** OS操作系统的移动操作系统。

[0490] 计算机可读存储介质1722可以存储提供一些实施例的功能的编程和数据结构。当由处理子系统1704执行时使处理器提供上述功能的软件(程序、代码模块、指令)可以存储在存储子系统1718中。作为示例，计算机可读存储介质1722可以包括非易失性存储器，诸如硬盘驱动器、磁盘驱动器、诸如CD ROM、DVD、**Blu-Ray®** (蓝光)盘或其它光学介质的光盘驱动器。计算机可读存储介质1722可以包括但不限于，**Zip®** 驱动器、闪存存储器卡、通用串行总线(USB)闪存驱动器、安全数字(SD)卡、DVD盘、数字视频带等。计算机可读存储介质1722也可以包括基于非易失性存储器的固态驱动器(SSD)(诸如基于闪存存储器的SSD、企业闪存驱动器、固态ROM等)、基于易失性存储器的SSD(诸如基于固态RAM、动态RAM、静态RAM、DRAM的SSD、磁阻RAM(MRAM)SSD)，以及使用基于DRAM和基于闪存存储器的SSD的组合的混合SSD。计算机可读介质1722可以为计算机系统1700提供计算机可读指令、数据结构、程序模块和其它数据的存储。

[0491] 在某些实施例中，存储子系统1700也可以包括计算机可读存储介质读取器1720，其可以进一步连接到计算机可读存储介质1722。可选地，与系统存储器1710一起和组合，计算机可读存储介质1722可以全面地表示远程、本地、固定和/或可移除存储设备加上用于存储计算机可读信息的存储介质。

[0492] 在某些实施例中，计算机系统1700可以提供对执行一个或多个虚拟机的支持。计算机系统1700可以执行诸如管理程序之类的程序，以便促进虚拟机的配置和管理。每个虚拟机可以被分配存储器、计算(例如，处理器、内核)、I/O和联网资源。每个虚拟机通常运行其自己的操作系统，其可以与由计算机系统1700执行的其它虚拟机执行的操作系统相同或不同。相应地，多个操作系统可以潜在地由计算机系统1700并发地运行。每个虚拟机一般独立于其它虚拟机运行。

[0493] 通信子系统1724提供到其它计算机系统和网络的接口。通信子系统1724用作用于从计算机系统1700的其它系统接收数据和向其发送数据的接口。例如，通信子系统1724可以使计算机系统1700能够经由互联网建立到一个或多个客户端计算设备的通信信道，用于

从客户端设备接收信息和发送信息到客户端计算设备。

[0494] 通信子系统1724可以支持有线和/或无线通信协议两者。例如，在某些实施例中，通信子系统1724可以包括用于(例如，使用蜂窝电话技术、高级数据网络技术(诸如3G、4G或EDGE(全球演进的增强数据速率)、WiFi(IEEE 802.11族标准)、或其它移动通信技术、或其任意组合)接入无线语音和/或数据网络的射频(RF)收发器部件、全球定位系统(GPS)接收器部件和/或其它部件。在一些实施例中，作为无线接口的附加或替代，通信子系统1724可以提供有线网络连接(例如，以太网)。

[0495] 通信子系统1724可以以各种形式接收和发送数据。例如，在一些实施例中，通信子系统1724可以以结构化和/或非结构化的数据馈送1726、事件流1728、事件更新1730等形式接收输入通信。例如，通信子系统1724可以被配置为实时地从社交媒体网络的用户和/或诸如**Twitter®**馈送、**Facebook®**更新、诸如丰富站点摘要(RSS)馈送的web馈送的其它通信服务接收(或发送)数据馈送1726，和/或来自一个或多个第三方信息源的实时更新。

[0496] 在某些实施例中，通信子系统1724可以被配置为以连续数据流的形式接收本质上可能是连续的或无界的没有明确结束的数据，其中连续数据流可以包括实时事件的事件流1728和/或事件更新1730。生成连续数据的应用的示例可以包括例如传感器数据应用、金融报价机、网络性能测量工具(例如网络监视和流量管理应用)、点击流分析工具、汽车流量监视等。

[0497] 通信子系统1724也可以被配置为向一个或多个数据库输出结构化和/或非结构化的数据馈送1726、事件流1728、事件更新1730等，其中所述一个或多个数据库可以与耦合到计算机系统1700的一个或多个流数据源计算机通信。

[0498] 计算机系统1700可以是各种类型中的一种，包括手持便携式设备(例如，**iPhone®**蜂窝电话、**iPad®**计算平板、PDA)、可穿戴设备(例如，Google **Glass®**头戴式显示器)、个人计算机、工作站、大型机、信息站、服务器机架或任何其它数据处理系统。

[0499] 由于计算机和网络不断变化的性质，对图17中绘出的计算机系统1700的描述旨在仅仅作为具体示例。具有比图17中所绘出的系统更多或更少部件的许多其它配置是可能的。基于本文所提供的公开内容和教导，本领域普通技术人员将理解实现各种实施例的其它方式和/或方法。

[0500] 虽然已经描述了本公开的具体实施例，但是各种修改、变更、替代构造和等同物也包含在本公开的范围内。修改包括所公开特征的任何相关组合。本公开的实施例不限于在某些特定数据处理环境内的操作，而是可以在多个数据处理环境内自由操作。此外，虽然已使用特定系列的事务和步骤描述了本公开的实施例，然而，对本领域技术人员应当清楚的是，本公开的范围不限于所描述系列的事务和步骤。上述实施例的各种特征和方面可以被单独或结合使用。

[0501] 此外，虽然已经使用硬件和软件的特定组合描述了本公开的实施例，但是应该认识到的是，硬件和软件的其它组合也在本公开的范围内。本公开的实施例可以只用硬件、或只用软件、或利用其组合来实现。本文描述的各种进程可以在同一处理器或以任何组合的不同处理器上实现。相应地，在部件或模块被描述为被配置为执行某些操作的情况下，这种配置可以例如通过设计电子电路来执行操作、通过对可编程电子电路(诸如微处理器)进行编程来执行操作、或其任意组合来实现。进程可以利用各种技术来通信，包括但不限于用于

进程间通信的常规技术，并且不同的进程对可以使用不同的技术，或者同一对进程可以在不同时间使用不同的技术。

[0502] 因而，说明书和附图应当在说明性而不是限制性的意义上考虑。然而，将清楚的是，在不背离权利要求中阐述的更广泛精神和范围的情况下，可以对其进行添加、减少、删除和其它修改和改变。因此，虽然已描述了具体的实施例，但是这些实施例不旨在进行限制。各种修改和等效物都在以下权利要求的范围之内。

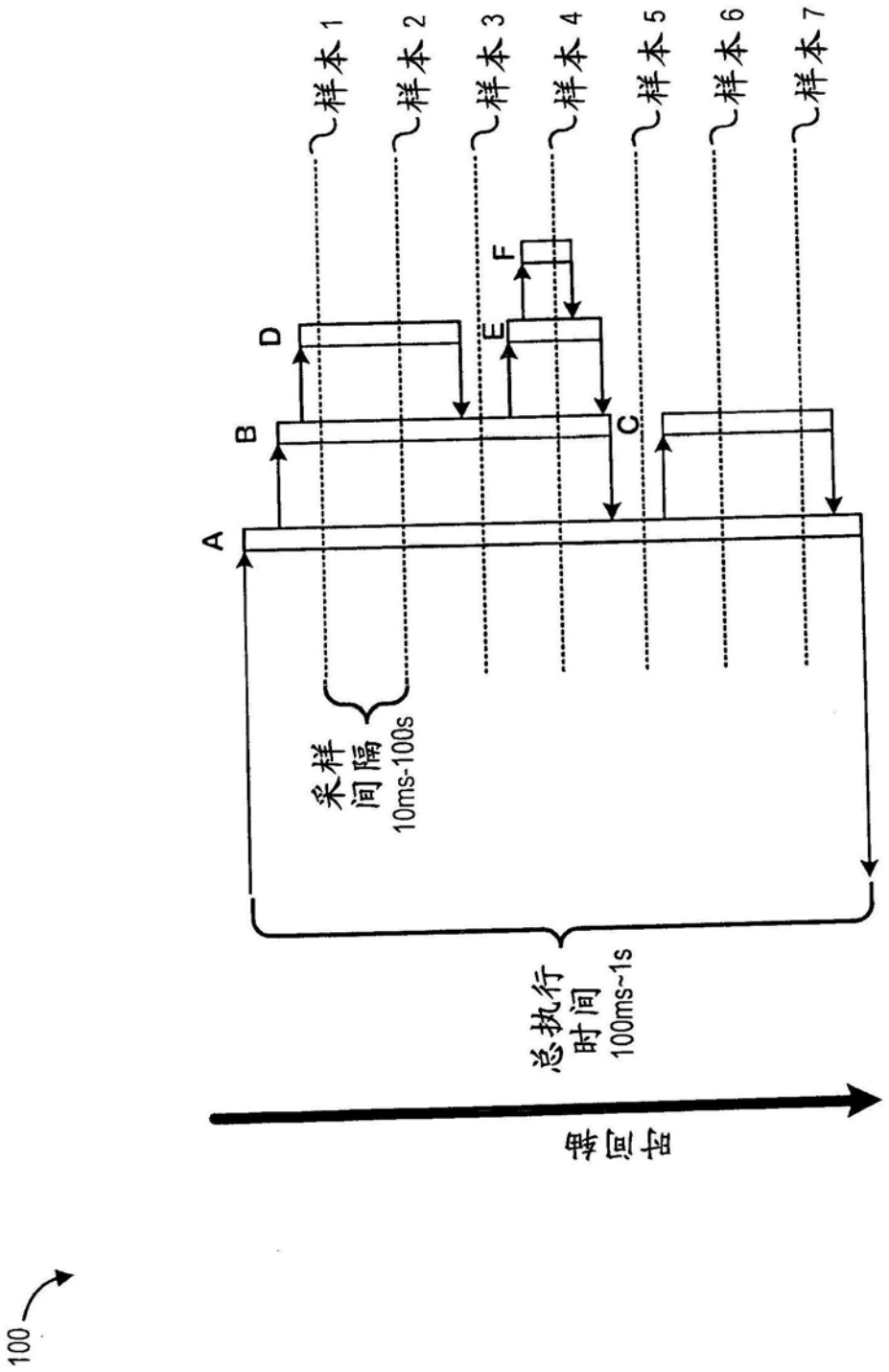

样本 1
样本 2
样本 3
样本 4
样本 5
样本 6
样本 7
A
B
C
D
E
F
采样间隔 10ms-100s
总执行时间 100ms~1s
时间轴
100

图1

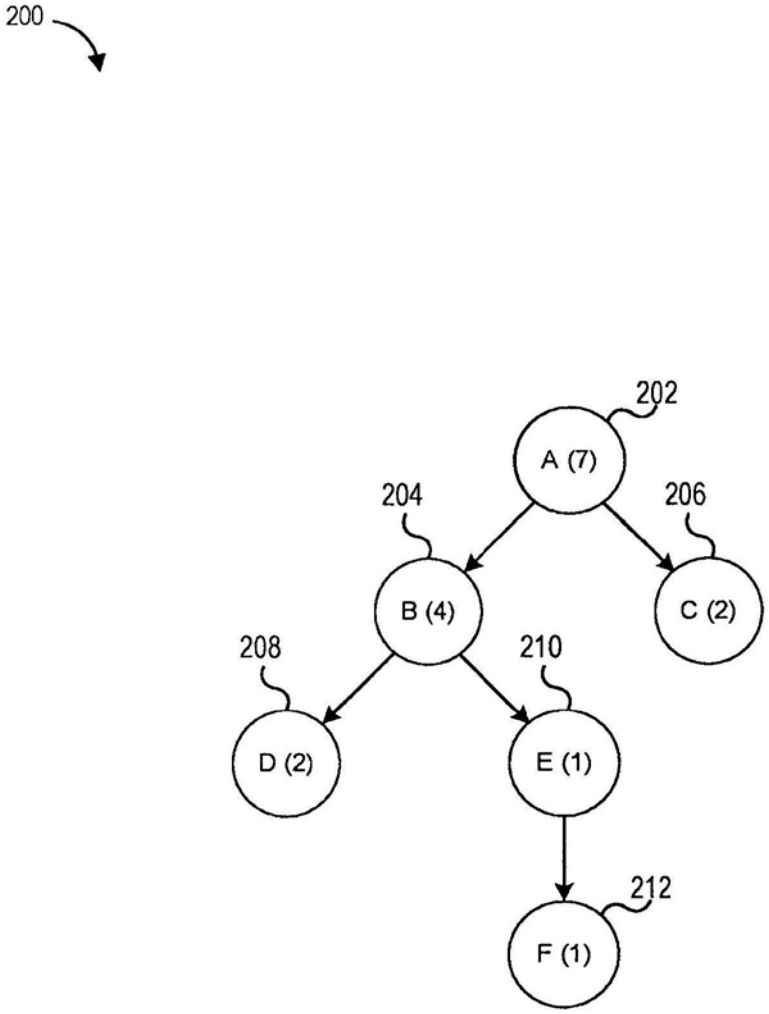



图2

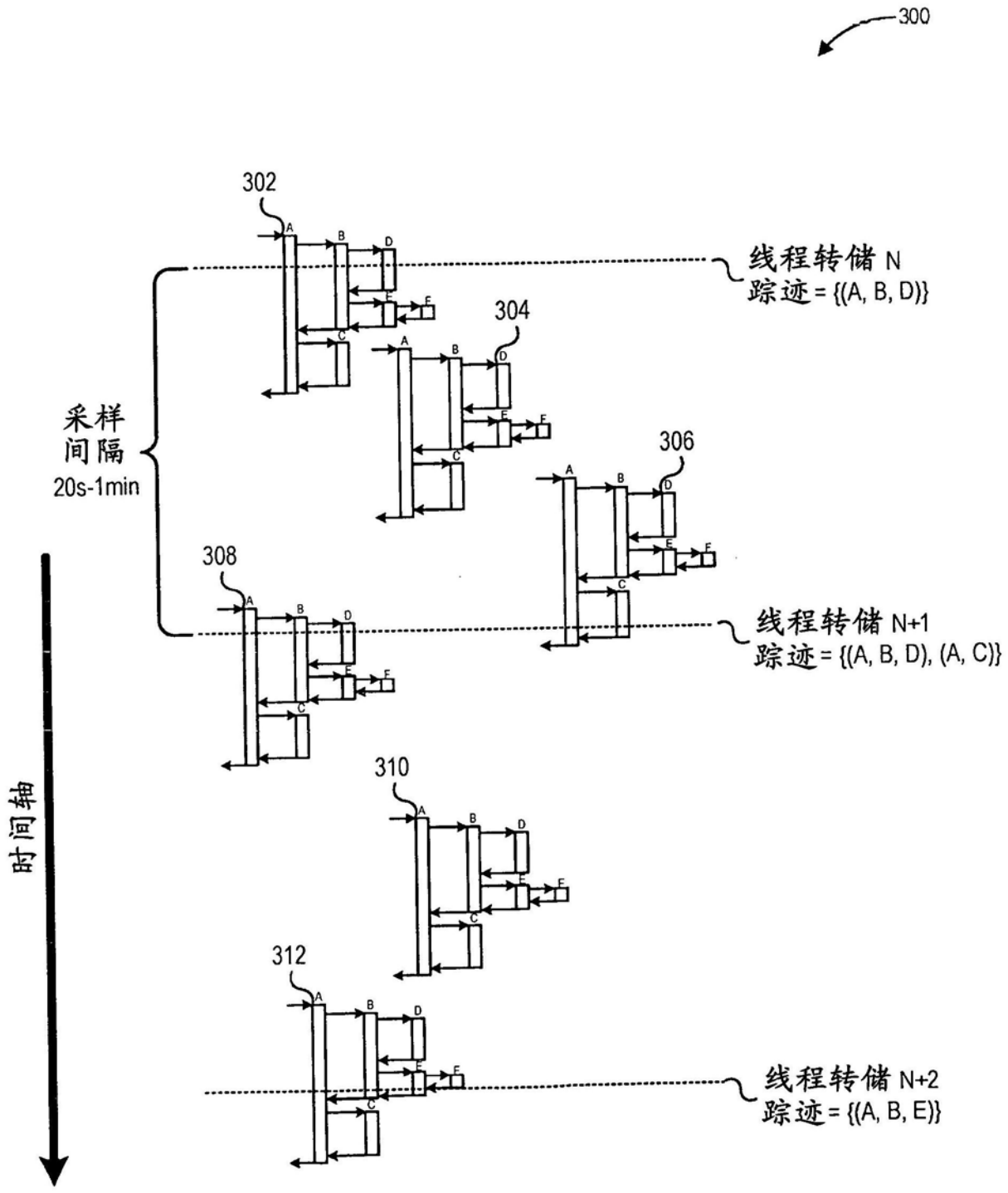

300
302
304
306
308
310
312
采样
间隔
20s-1min
时间轴
线程转储 N
踪迹 = {(A, B, D)}
线程转储 N+1
踪迹 = {(A, B, D), (A, C)}
线程转储 N+2
踪迹 = {(A, B, E)}

图3

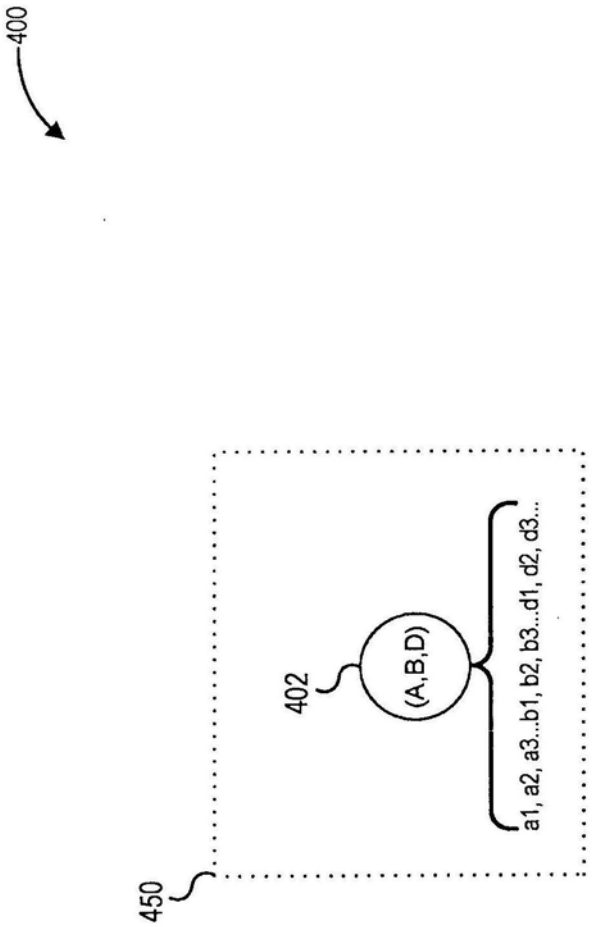



图4

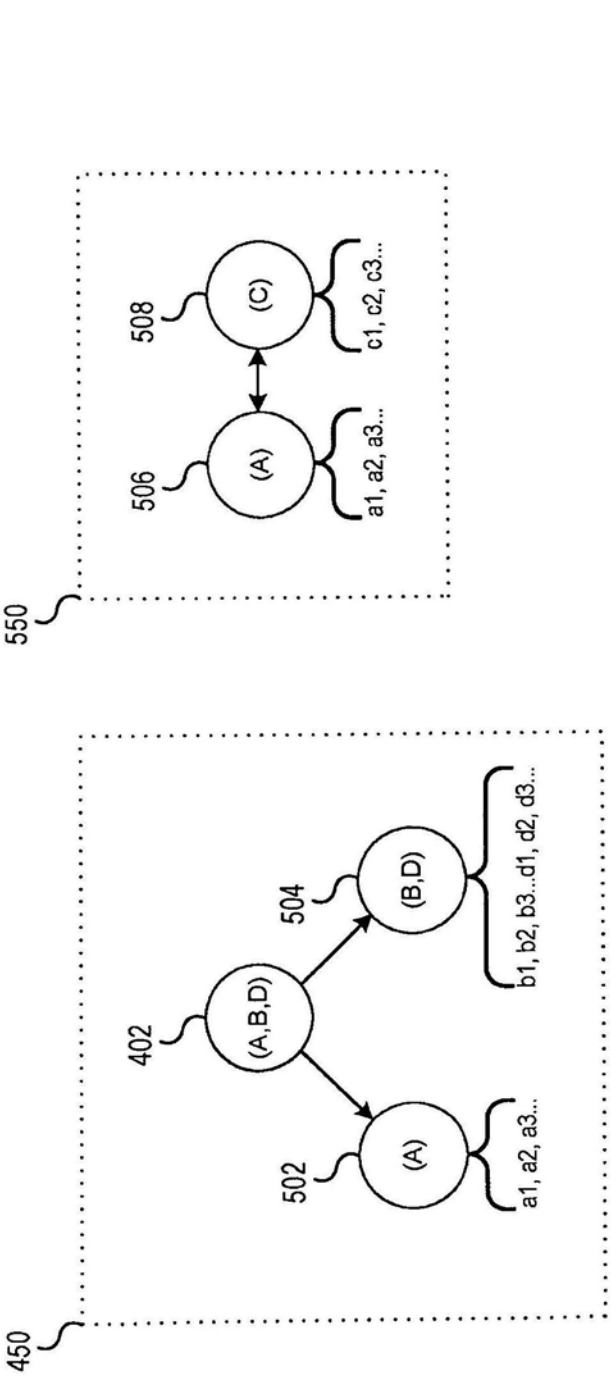

500
550
508
(C)
c1, c2, c3...
506
(A)
a1, a2, a3...
450
402
(A,B,D)
504
(B,D)
b1, b2, b3...d1, d2, d3...
502
(A)
a1, a2, a3...

图5

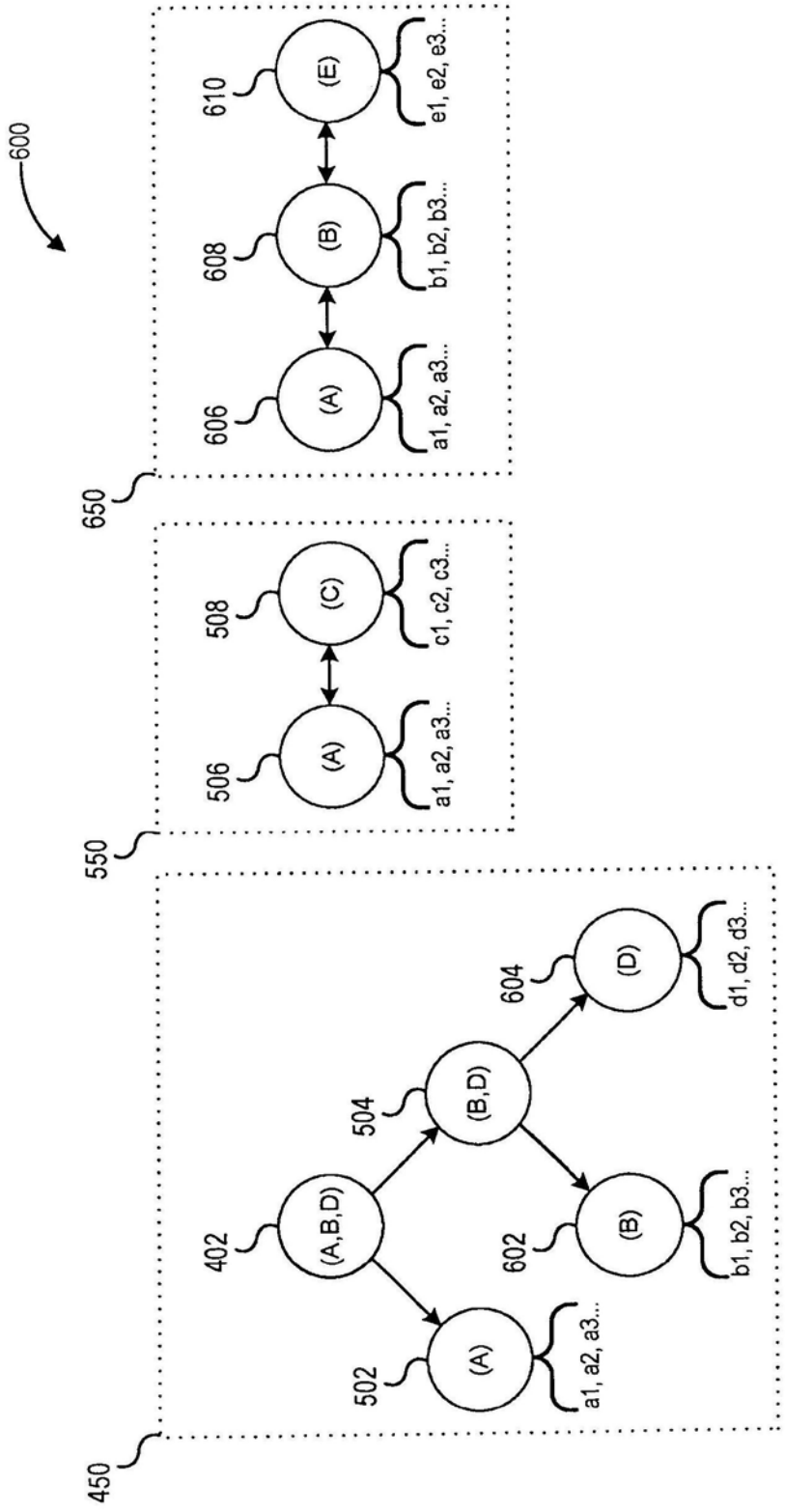

600
650
610
(E)
e1, e2, e3...
608
(B)
b1, b2, b3...
606
(A)
a1, a2, a3...
550
508
(C)
c1, c2, c3...
506
(A)
a1, a2, a3...
450
402
(A,B,D)
504
(B,D)
604
(D)
d1, d2, d3...
602
(B)
b1, b2, b3...
502
(A)
a1, a2, a3...

图6

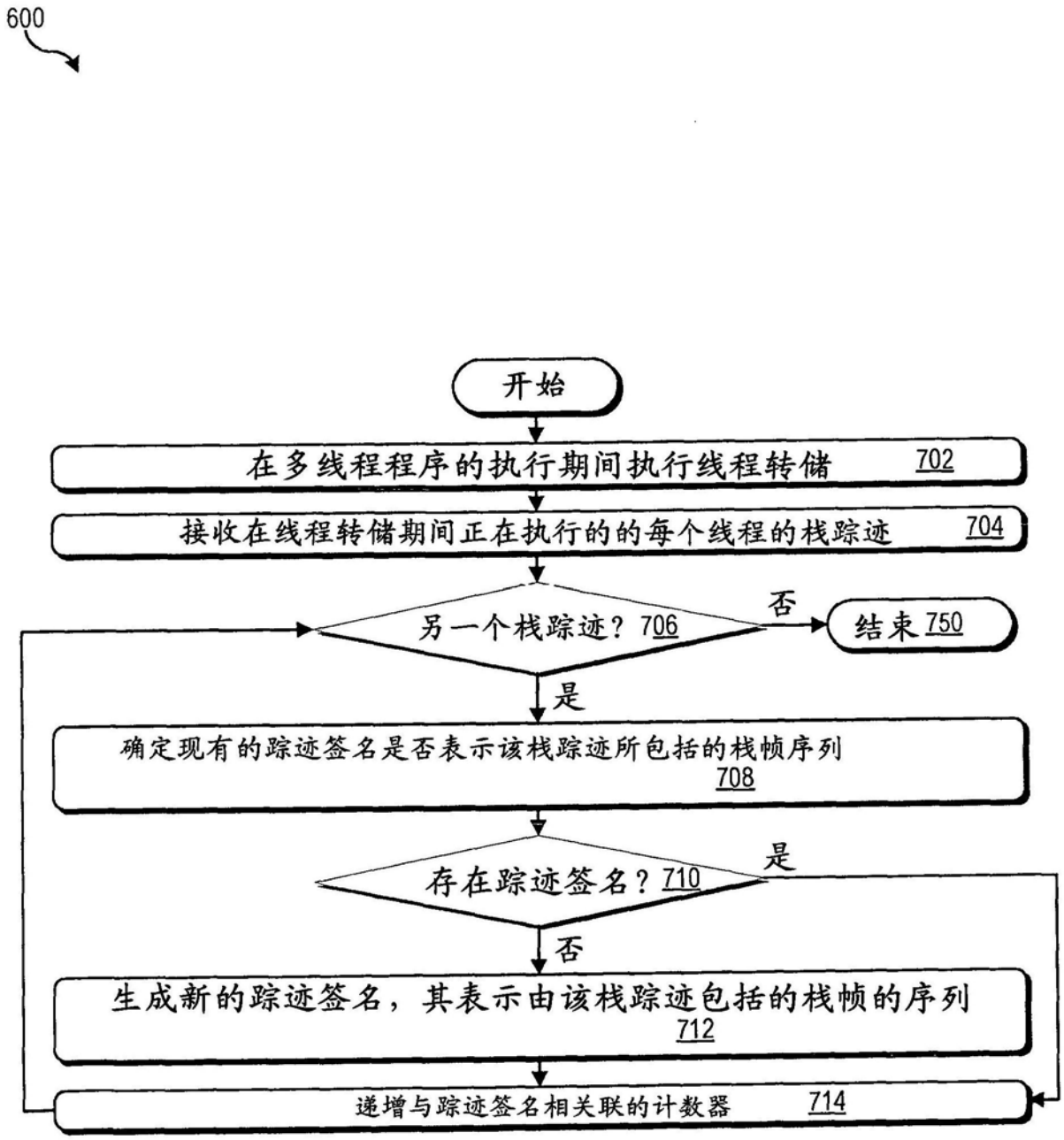

600
开始
在多线程程序的执行期间执行线程转储 702
接收在线程转储期间正在执行的的每个线程的栈踪迹 704
另一个栈踪迹？706
否
结束 750
是
确定现有的踪迹签名是否表示该栈踪迹所包括的栈帧序列 708
存在踪迹签名？710
是
否
生成新的踪迹签名，其表示由该栈踪迹包括的栈帧的序列 712
递增与踪迹签名相关联的计数器 714

图7

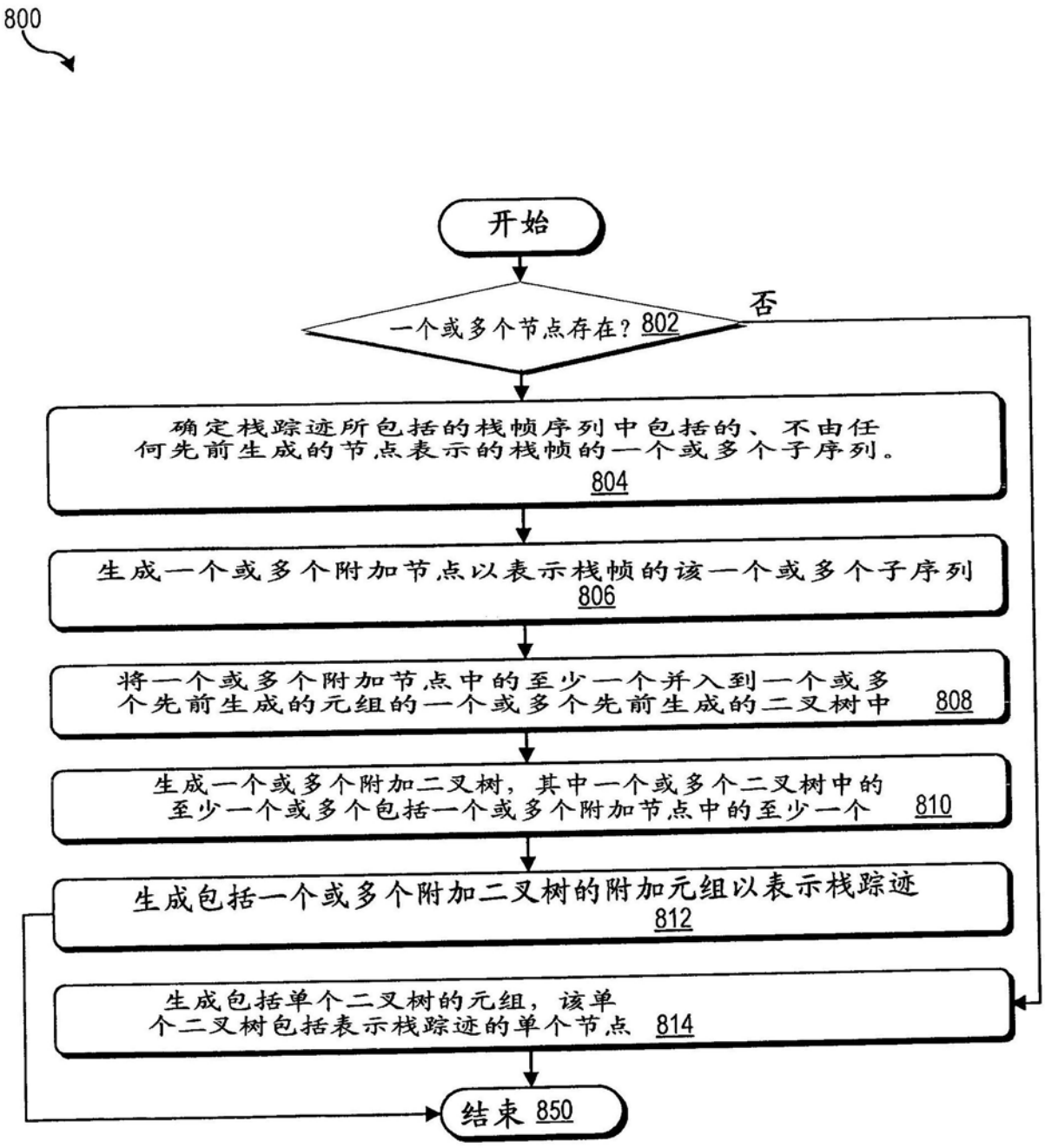

800
开始
一个或多个节点存在？802
否
确定栈踪迹所包括的栈帧序列中包括的、不由任何先前生成的节点表示的栈帧的一个或多个子序列。
804
生成一个或多个附加节点以表示栈帧的该一个或多个子序列
806
将一个或多个附加节点中的至少一个并入到一个或多个先前生成的元组的一个或多个先前生成的二叉树中
808
生成一个或多个附加二叉树，其中一个或多个二叉树中的至少一个或多个包括一个或多个附加节点中的至少一个
810
生成包括一个或多个附加二叉树的附加元组以表示栈踪迹
812
生成包括单个二叉树的元组，该单个二叉树包括表示栈踪迹的单个节点
814
结束 850

图8

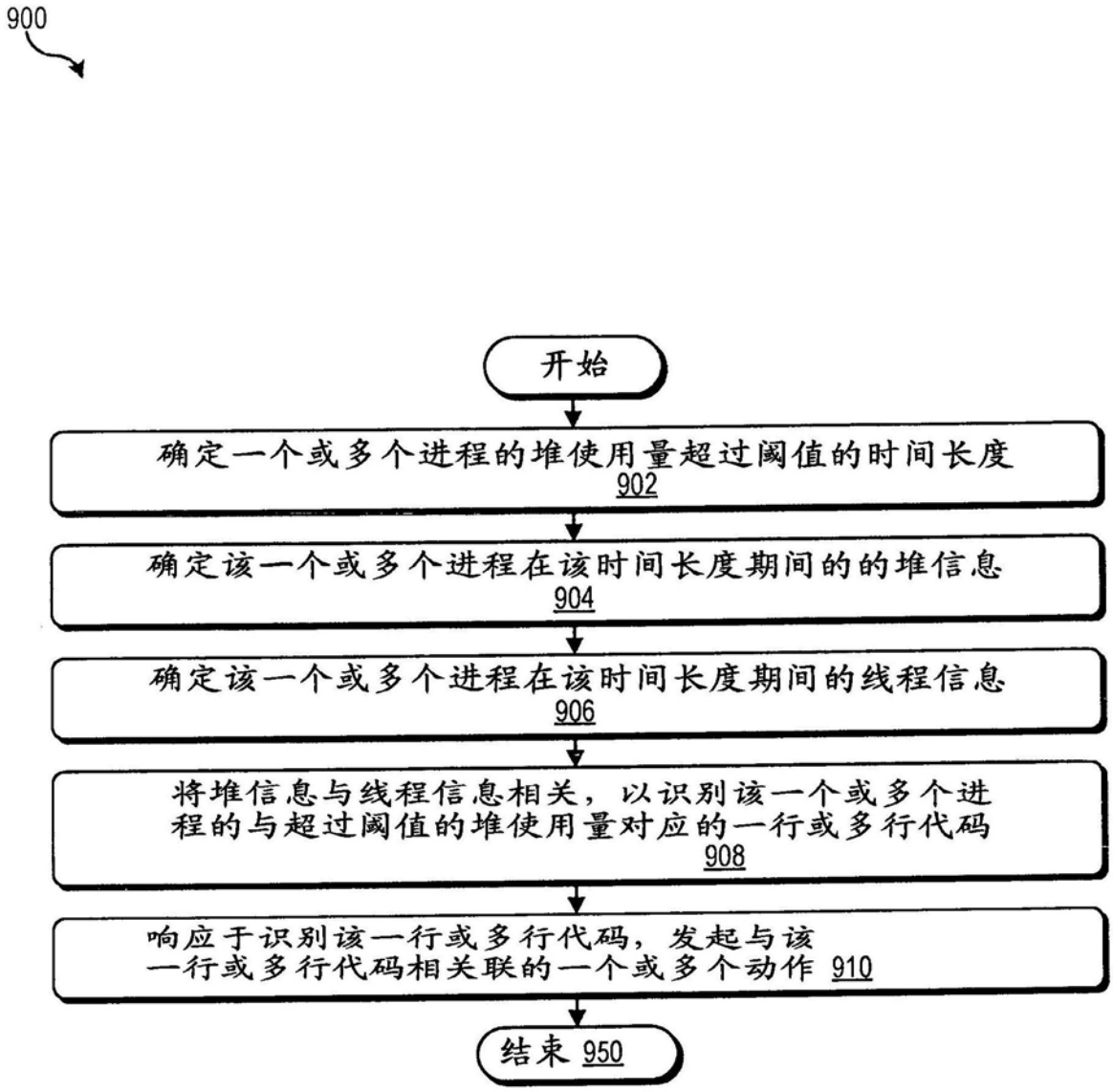

900
开始
确定一个或多个进程的堆使用量超过阈值的时间长度
902
确定该一个或多个进程在该时间长度期间的的堆信息
904
确定该一个或多个进程在该时间长度期间的线程信息
906
将堆信息与线程信息相关，以识别该一个或多个进程的与超过阈值的堆使用量对应的一行或多行代码
908
响应于识别该一行或多行代码，发起与该一行或多行代码相关联的一个或多个动作 910
结束 950

图9

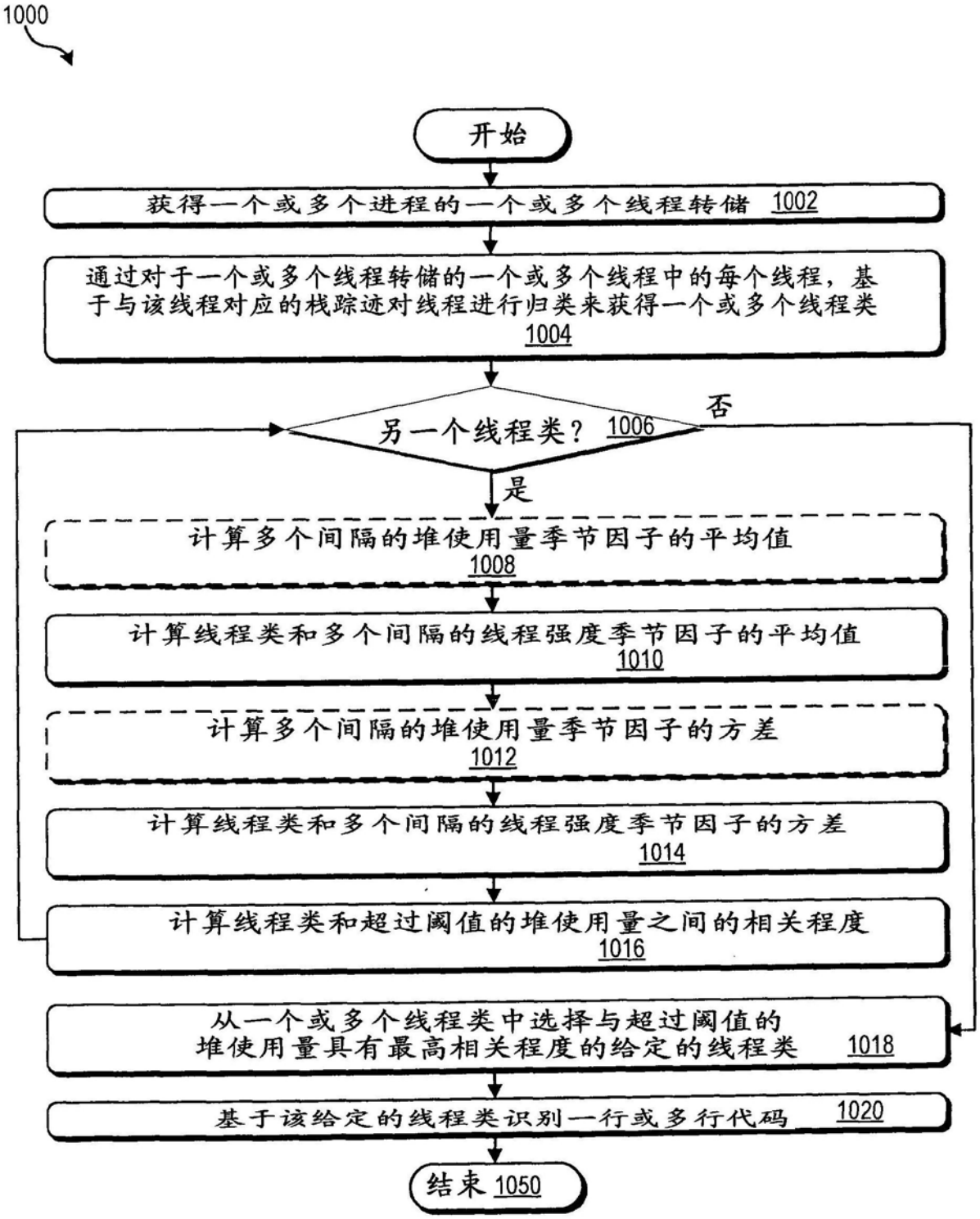

1000
开始
获得一个或多个进程的一个或多个线程转储 1002
通过对于一个或多个线程转储的一个或多个线程中的每个线程，基于与该线程对应的栈踪迹对线程进行归类来获得一个或多个线程类 1004
另一个线程类？ 1006
否
是
计算多个间隔的堆使用量季节因子的平均值 1008
计算线程类和多个间隔的线程强度季节因子的平均值 1010
计算多个间隔的堆使用量季节因子的方差 1012
计算线程类和多个间隔的线程强度季节因子的方差 1014
计算线程类和超过阈值的堆使用量之间的相关程度 1016
从一个或多个线程类中选择与超过阈值的堆使用量具有最高相关程度的给定的线程类 1018
基于该给定的线程类识别一行或多行代码 1020
结束 1050

图10

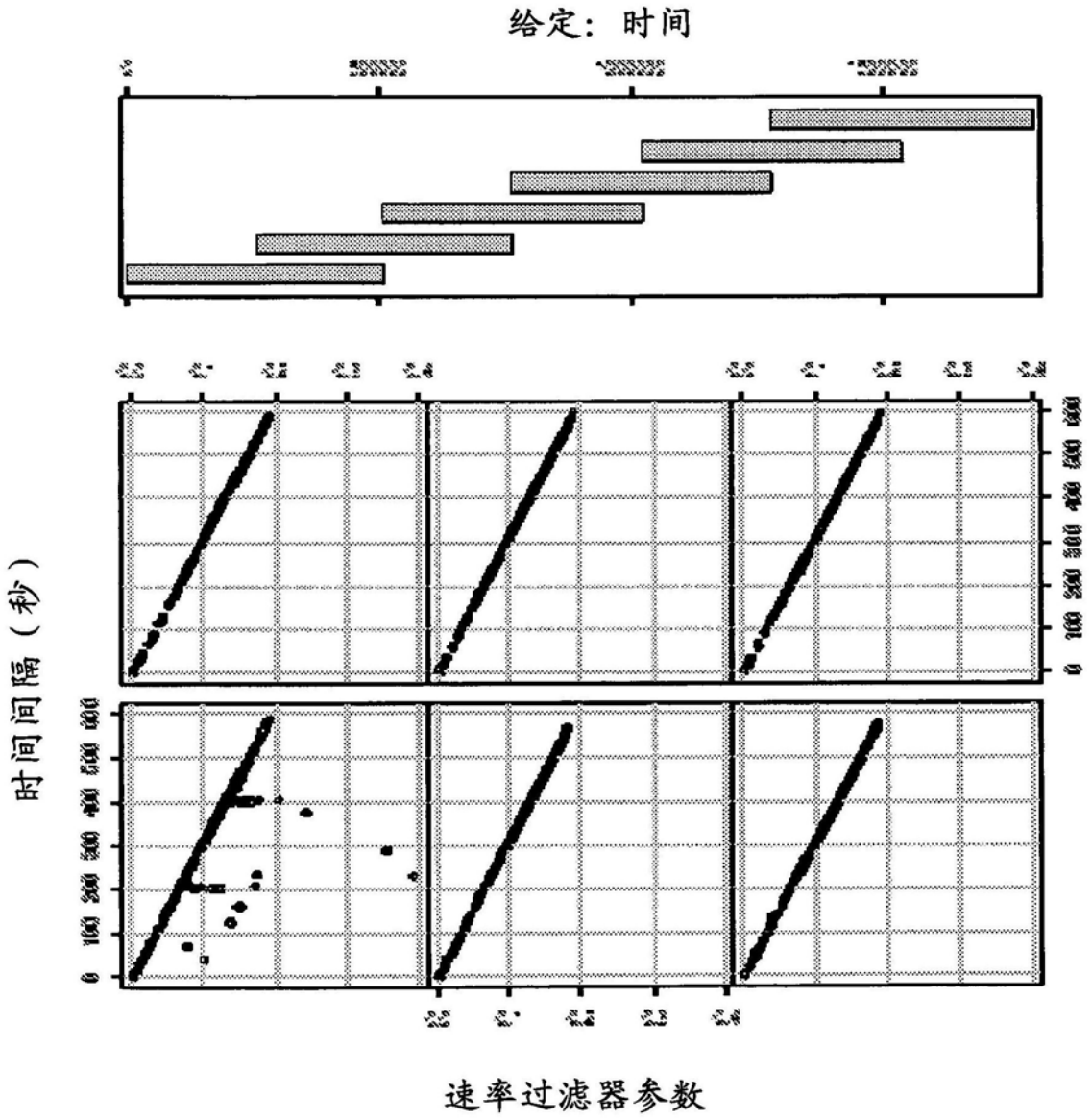



图11

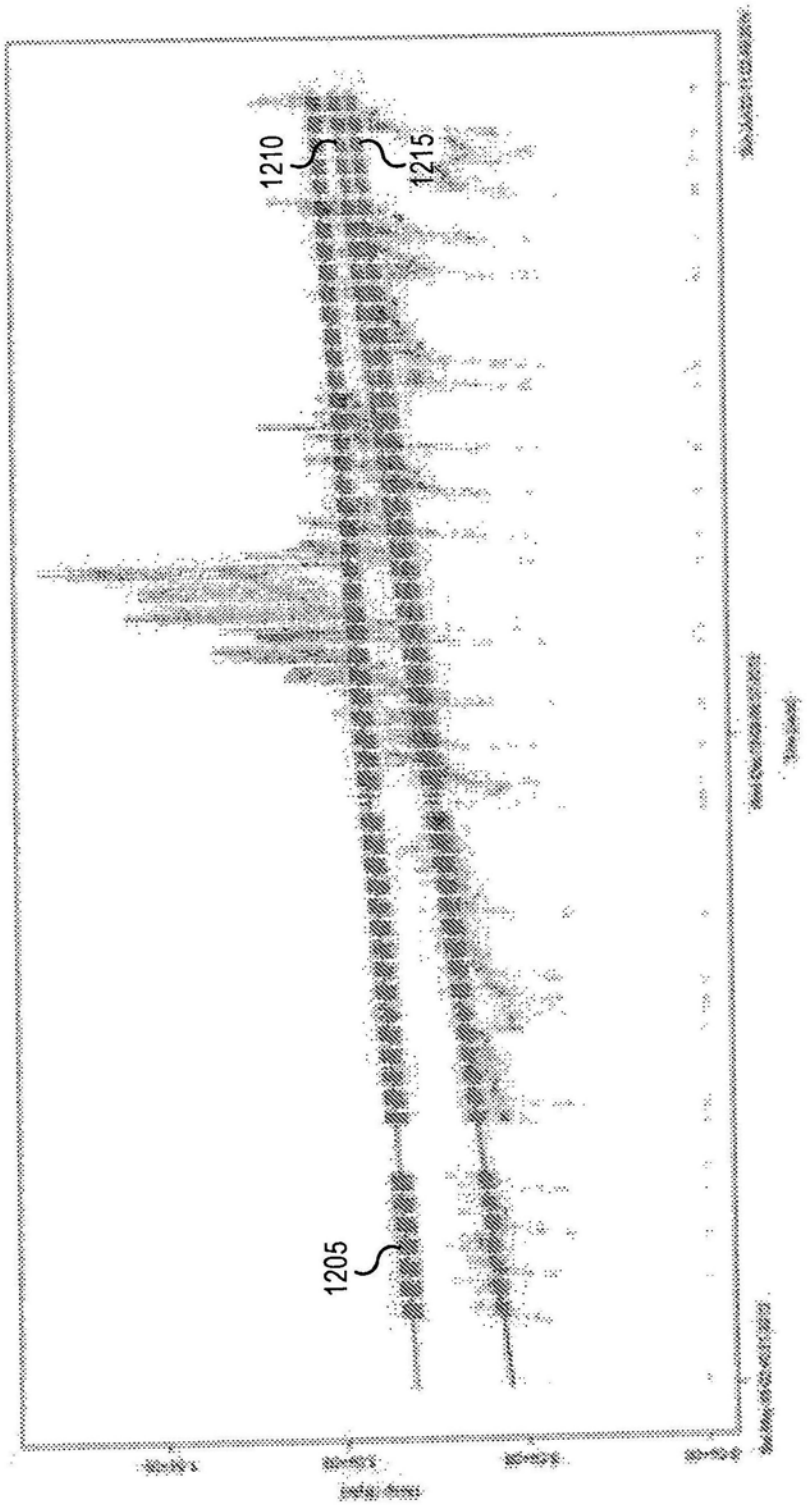



图12

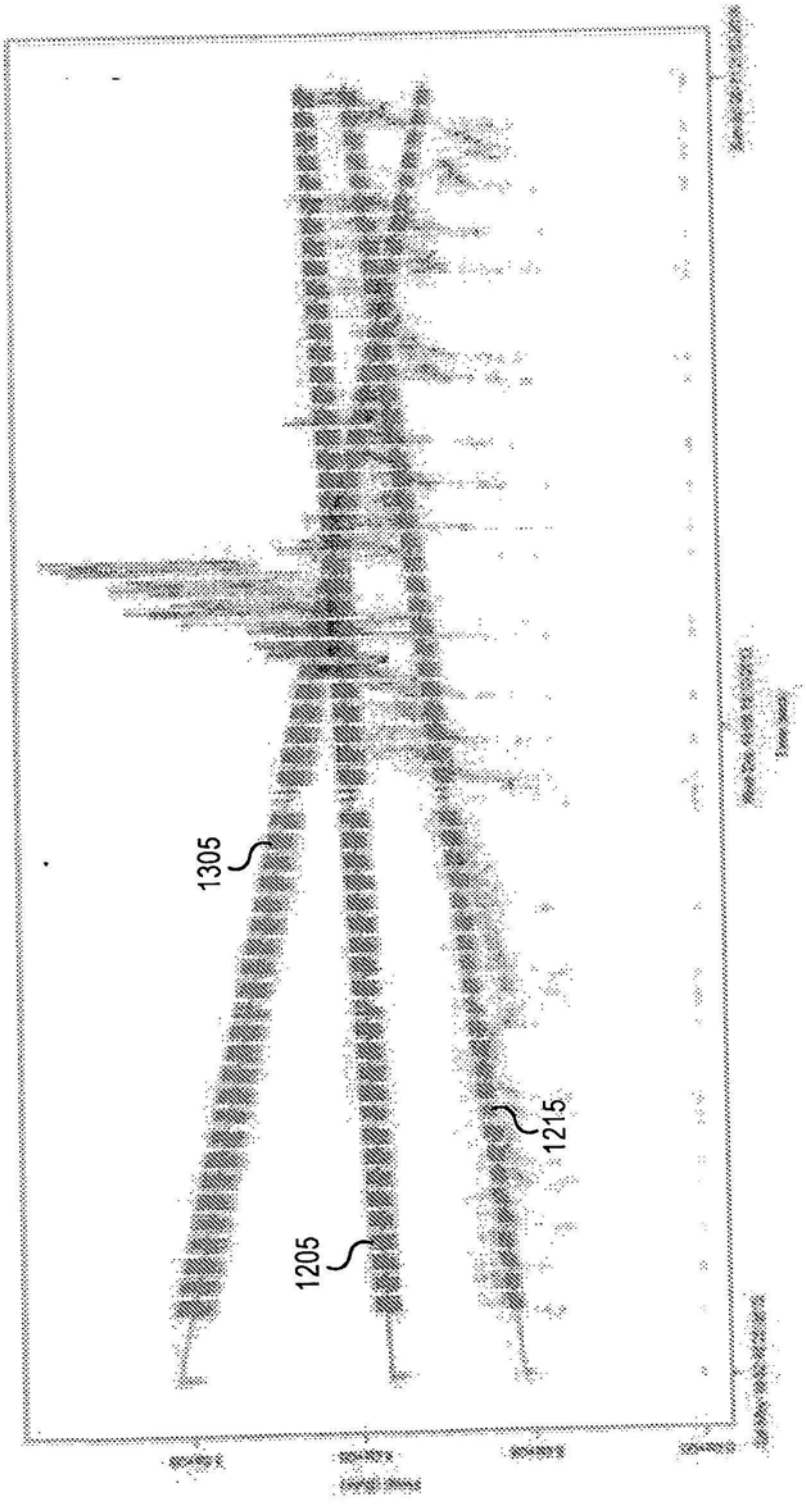



图13

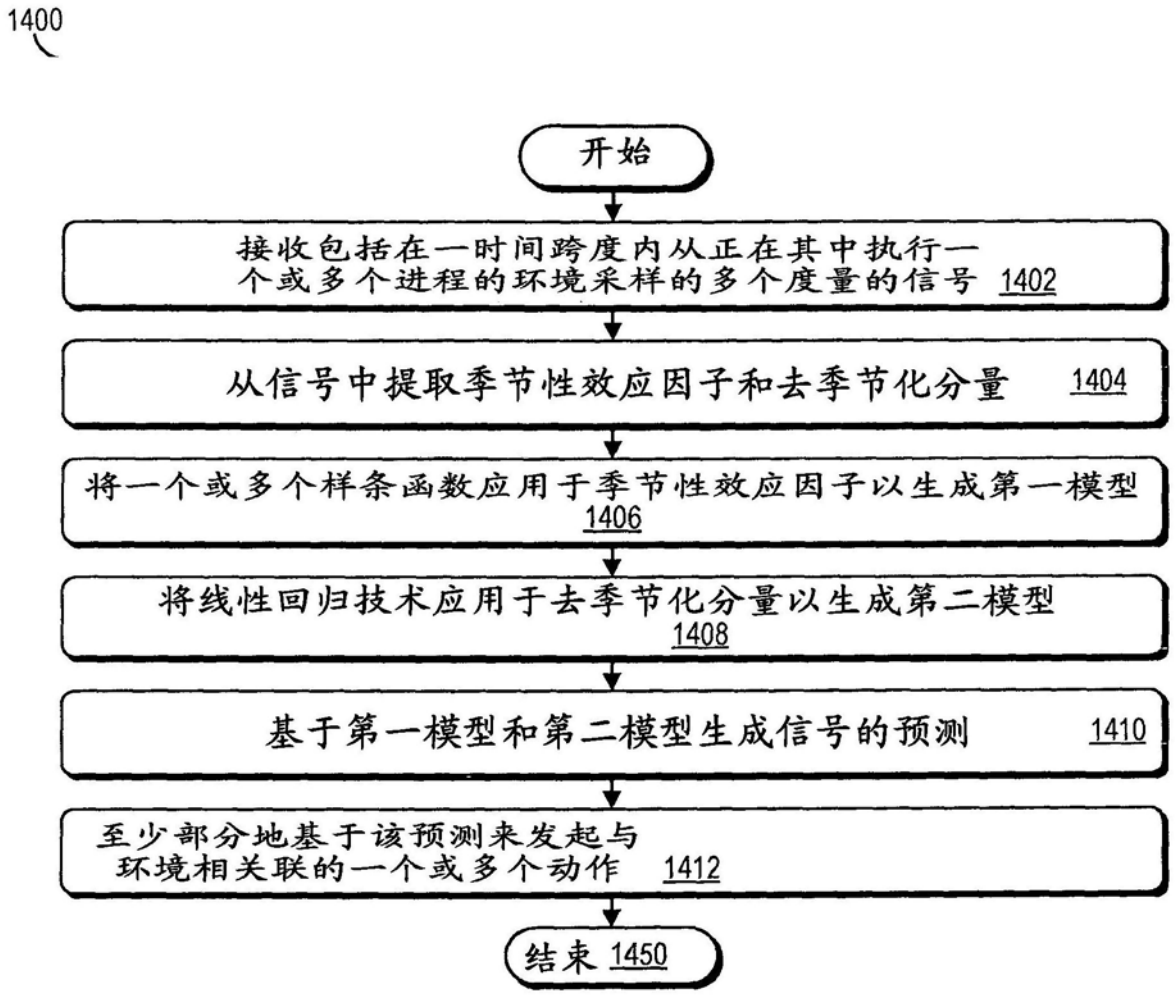

1400
开始
接收包括在一时间跨度内从正在其中执行一个或多个进程的环境采样的多个度量的信号 1402
从信号中提取季节性效应因子和去季节化分量 1404
将一个或多个样条函数应用于季节性效应因子以生成第一模型 1406
将线性回归技术应用于去季节化分量以生成第二模型 1408
基于第一模型和第二模型生成信号的预测 1410
至少部分地基于该预测来发起与环境相关联的一个或多个动作 1412
结束 1450

图14

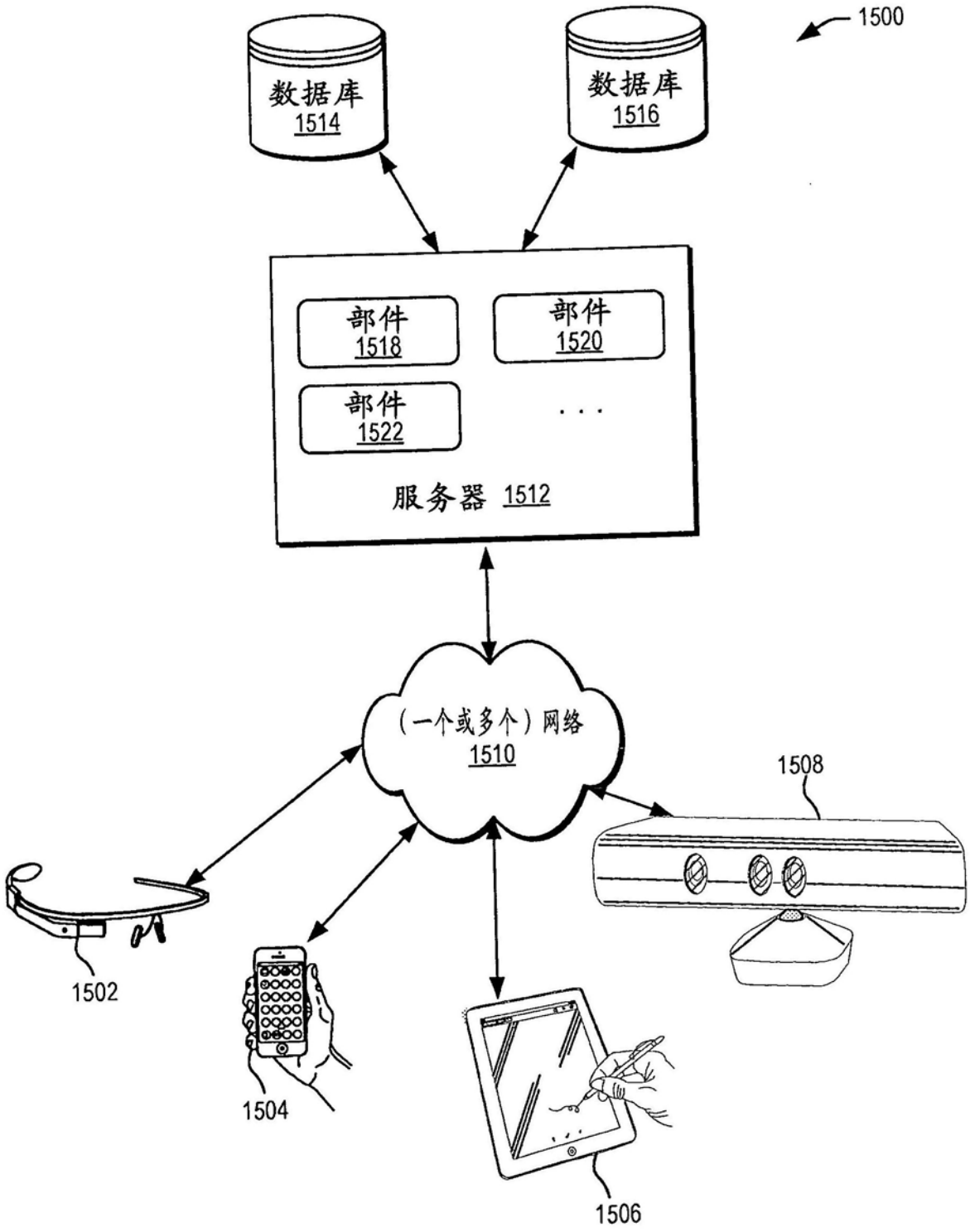

1500
数据库
1514
数据库
1516
部件
1518
部件
1520
部件
1522
. . .
服务器 1512
(一个或多个)网络
1510
1508
1502
1504
1506

图15

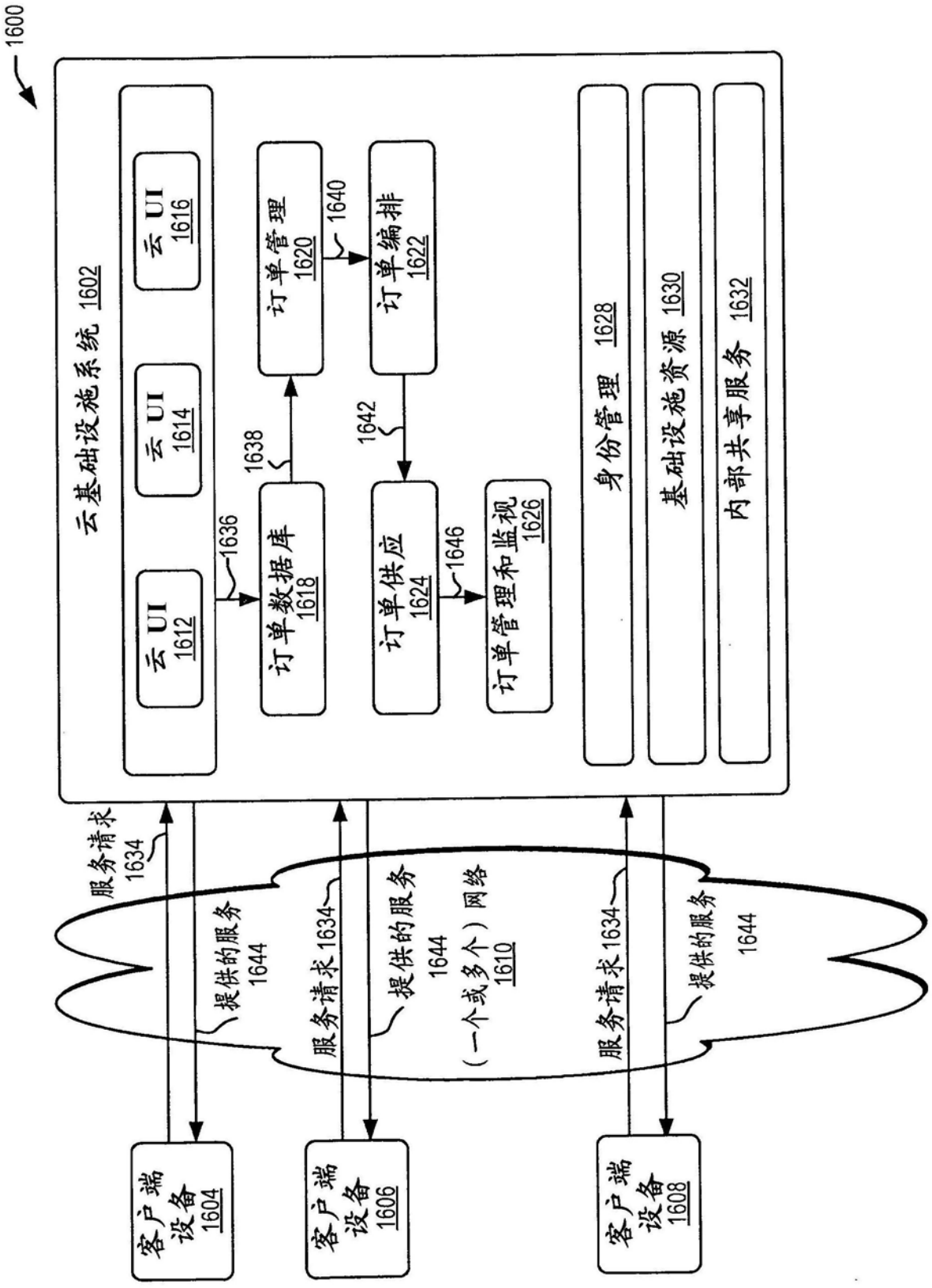

1600
云基础设施系统 1602
云 UI 1612
云 UI 1614
云 UI 1616
订单数据库 1618
订单管理 1620
订单编排 1622
订单供应 1624
订单管理和监视 1626
身份管理 1628
基础设施资源 1630
内部共享服务 1632
1636
1638
1640
1642
1646
服务请求 1634
提供的服务 1644
（一个或多个）网络 1610
客户端设备 1604
客户端设备 1606
客户端设备 1608

图16

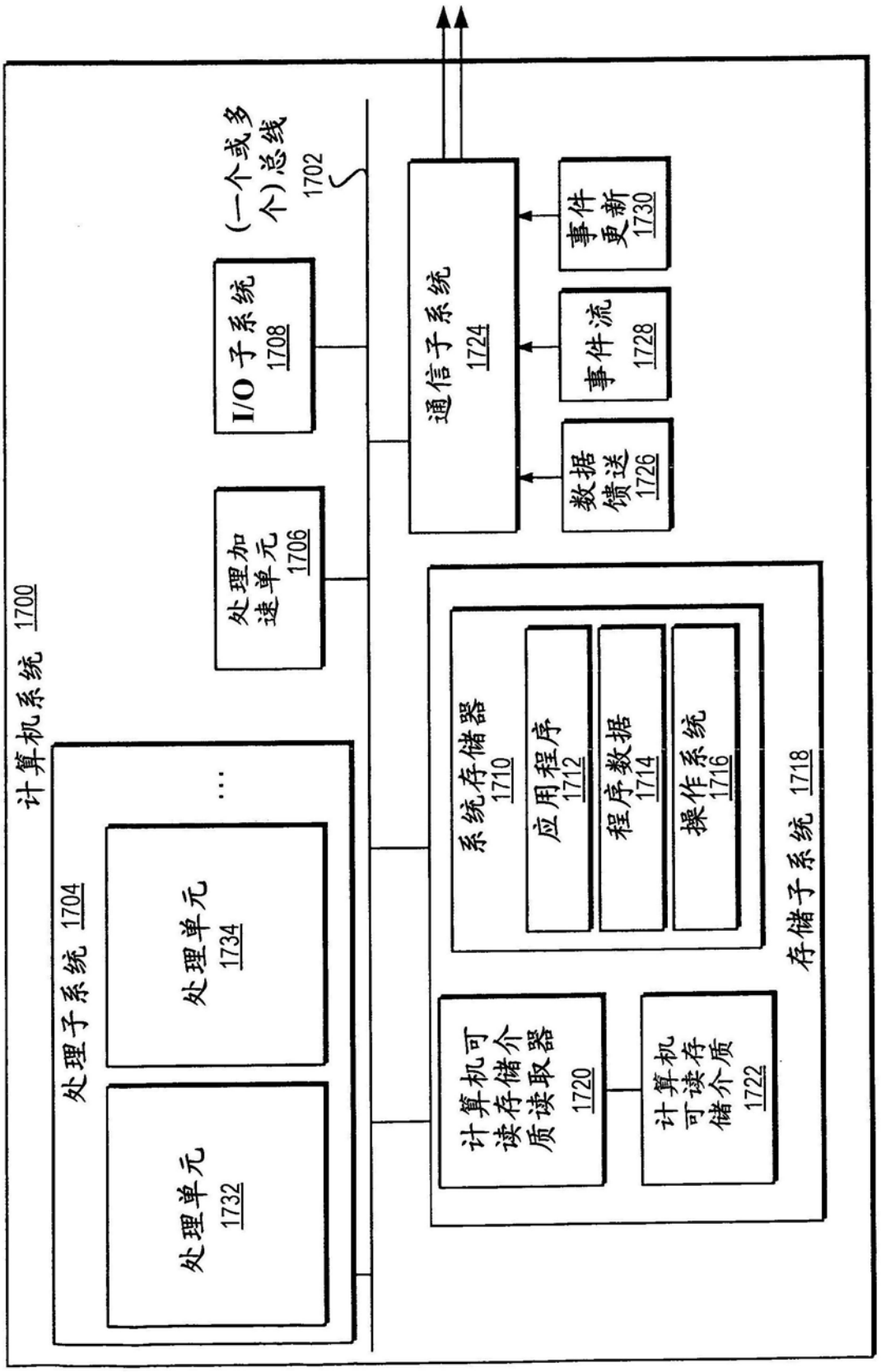

计算机系统 1700
处理子系统 1704
处理单元 1732
处理单元 1734
...
(一个或多个)总线 1702
I/O 子系统 1708
处理加速单元 1706
通信子系统 1724
事件更新 1730
事件流 1728
数据馈送 1726
存储子系统 1718
系统存储器 1710
应用程序 1712
程序数据 1714
操作系统 1716
计算机可读存储介质读取器 1720
计算机可读存储介质 1722

图17