(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2017 年 10 月 5 日 (05.10.2017)

WIPO | PCT

(10) 国际公布号
WO 2017/167062 A1

(51) 国际专利分类号:
G06F 9/52 (2006.01)

(21) 国际申请号: PCT/CN2017/077448

(22) 国际申请日: 2017 年 3 月 21 日 (21.03.2017)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201610195050.9 2016 年 3 月 30 日 (30.03.2016) CN

(71) 申请人: 阿里巴巴集团控股有限公司 (ALIBABA GROUP HOLDING LIMITED) [—/CN]; 开曼群岛大开曼资本大厦一座四层 847 号邮箱, Grand Cayman (KY)。

(72) 发明人; 及

(71) 申请人 (仅对美国): 邢勇 (XING, Yong) [CN/CN]; 中国浙江省杭州市余杭区文一西路 969 号 3 号楼 5 楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。

(74) 代理人: 北京国昊天诚知识产权代理有限公司 (CO-HORIZON INTELLECTUAL PROPERTY INC.); 中国北京市朝阳区小关北里甲 2 号渔阳置业大厦 B 座 605, Beijing 100029 (CN)。

(81) 指定国 (除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) 指定国 (除另有指明，要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:
— 包括国际检索报告(条约第 21 条(3))。

(54) Title: APPLICATION PROGRAM INTERFACE DEADLOCK MONITORING METHOD AND DEVICE

(54) 发明名称 : 一种应用程序接口死锁监控方法和装置

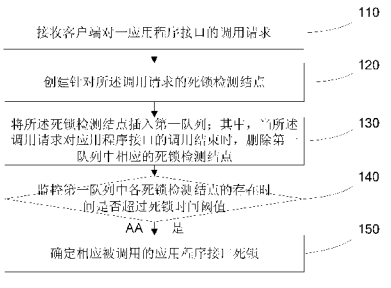


图 1

110 Receive a call request for an application program interface from a client end
120 Create a deadlock detection node targeting the call request
130 Insert the deadlock detection node into a first queue; and delete, after calling the application program interface corresponding to the call request, the corresponding deadlock detection node in the first queue
140 Monitor whether a duration of existence of the deadlock detection node in the first queue exceeds a deadlock time threshold
150 Determine the corresponding application program interface being called as a deadlock
AA Yes

(57) Abstract: The present invention relates to the technical field of computing. Disclosed are an application program interface deadlock monitoring method and device. The method comprises: receiving a call request for an application program interface from a client end (110); creating a deadlock detection node targeting the call request (120); inserting the deadlock detection node into a first queue, and deleting, after calling the application program interface corresponding to the call request, the corresponding deadlock detection node in the first queue (130); monitoring whether a duration of existence of the deadlock detection node in the first queue exceeds a deadlock time threshold (140); and if the duration of the existence of the deadlock detection node exceeds the deadlock time threshold, then determining the application program interface being called as a deadlock (150). The invention monitors, by using a deadlock monitoring node and by monitoring a duration of existence of the deadlock monitoring node, a deadlock of an API, thereby achieving improved applicability and a higher level of precision.

(57) 摘要: 一种应用程序接口死锁监控方法和装置，涉及计算机技术领域。所述方法包括：接收客户端对一应用程序接口的调用请求（110）；创建针对所述调用请求的死锁检测结点（120）；将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点（130）；监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值（140）；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁（150）。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，适用性更强，而且精确度较高。

WO 2017/167062 A1

一种应用程序接口死锁监控方法和装置

技术领域

本申请涉及计算机技术领域，特别是涉及一种应用程序接口死锁监控方法和一种应用程序接口死锁监控装置。

背景技术

在分步式系统中，系统架构和设计非常复杂，处理逻辑不可避免地存在错误。例如，客户端通过调用系统的应用程序接口（Application Programming Interface，API）请求服务，会有较低的概率很长时间没有得到响应，系统既不返回用户成功，也不返回用户错误。这就是API死锁，死锁的发生会导致程序无限等待或资源严重消耗，使整个系统没有可用资源而陷于瘫痪，严重影响系统安全性和可靠性。

现有的死锁检测的方法主要可以分为静态和动态两大类。静态方法最基本的是代码检查，而所有直接在源程序上做静态死锁检测的技术，都会使用静态程序分析技术。静态程序分析技术是对并发程序的源代码或者规格说明进行人工或者自动化分析，然后通过分析获得程序各个模块的数据依赖关系。静态方法能取得一定的效果，但如果出现并发缺陷如死锁和活锁，其产生来源于特定的程序状态，而且并发软件状态空间非常庞大，时间耗费较长，而且会使得以人工为主的代码检查难以胜任，适用性不高。

动态方法则是通过并发程序在真实环境或模拟环境中运行，通过检测软件对运行信息进行收集，然后利用收集到的信息进行死锁检测。动态分析方法可以分为测试和监控两大类。动态方法只能覆盖到有限的软件运行情况，对于死锁这类出现概率低的缺陷，并不能有效的检测，同样适用性不高。

发明内容

鉴于上述问题，提出了本申请实施例以便提供一种克服上述问题或者至少部分地解决上述问题的一种应用程序接口死锁监控方法和相应的一种应用程序接口死锁监控装置。

为了解决上述问题，本申请公开了一种应用程序接口死锁监控方法，其特征在于，包括:

接收客户端对一应用程序接口的调用请求;

创建针对所述调用请求的死锁检测结点;

将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点;

监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值;

如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。

可选地，所述将所述死锁检测结点插入第一队列的步骤之前，还包括:

根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位;

则所述将所述死锁检测结点插入第一队列的步骤包括：将所述死锁检测结点插入所述槽位中的第一队列。

可选地，所述监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值的步骤，包括:

定期将时间轮的指针移动到下一个槽位;

对于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结

点的死锁时间阈值是否超过一个时间轮周期;

如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值;

如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则确定相应被调用的应用程序接口死锁。

可选地，在根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位的步骤之前，还包括:

对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。

可选地，所述对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应的时间轮的步骤，包括:

获取死锁检测结点的内存地址的哈希值;

将所述哈希值对时间轮的数量取余数;

根据余数与时间轮的对应关系，将死锁检测结点分配到相应的时间轮中。

可选地，在确定相应被调用的应用程序接口死锁的步骤之后，还包括:

删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁;

或，删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求。

可选地，所述删除第一队列中相应的死锁检测结点，并重启客户端对

所述应用程序接口的调用请求的步骤之前，还包括:

接收客户端对重启全局标识的打开请求；所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求;

根据所述打开请求，打开所述重启全局标识。

可选地，所述通知调用请求所对应的客户端对所述应用程序接口的调用死锁的步骤，包括:

获取死锁的应用程序接口的信息;

将所述死锁的应用程序接口的信息序列化为字符串返回给客户端;所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

可选地，在创建针对所述调用请求的死锁检测结点之前，还包括:

接收客户端针对所述应用程序接口配置的时间长度阈值。

可选地，所述时间轮的每个槽位配置有自旋锁；所述自旋锁在第一队列被操作之前添加，以及在第一队列被操作完毕之后释放。

可选地，所述第一队列包括双向链表或单向链表。

可选地，当所述第一队列为双向链表时，所述死锁检测结点的数据结构包括:

双向链表结点参数、死锁时间阈值参数、时间轮哈希索引参数、时间轮指针槽位参数。

可选地，所述创建针对所述调用请求的死锁检测结点的步骤，包括:

在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

可选地，所述第一队列为优先级队列，则所述将所述死锁检测结点插

入第一队列的步骤包括:

计算所述死锁检测结点的优先级; 所述死锁检测结点的优先级为截止时间; 所述截止时间为检测结点创建时间与死锁时间阈值之和;

根据所述死锁检测结点的优先级, 将所述死锁检测结点插入优先级队列。

可选地, 所述监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值的步骤, 包括:

检测优先级队列首部的死锁检测结点的截止时间是否到达;

如果优先级队列首部的检测结点的截止时间到达, 则确定相应被调用的应用程序接口死锁。

本申请还公开了一种应用程序接口死锁监控装置, 其特征在于, 包括:

接收模块, 适于接收客户端对一应用程序接口的调用请求;

创建模块, 适于创建针对所述调用请求的死锁检测结点;

插入模块, 适于将所述死锁检测结点插入第一队列; 其中, 当所述调用请求对应用程序接口的调用结束时, 删除第一队列中相应的死锁检测结点;

监控模块, 适于监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值; 如果检测到任一死锁检测结点的存在时间超过死锁时间阈值, 则进入确认模块;

确认模块, 适于确定相应被调用的应用程序接口死锁。

可选地, 在所述插入模块之前, 还包括:

槽位计算模块, 适于根据当前的死锁时间阈值和时间轮中指针所在的槽位, 以及整个时间轮周期, 计算对应所述死锁检测结点的槽位;

则所述插入模块包括：插入子模块，适于将所述死锁检测结点插入所述槽位中的第一队列。

可选地，所述监控模块，包括：

指针移动子模块，适于定期将时间轮的指针移动到下一个槽位；

死锁时间判断子模块，适于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期；如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则进入死锁时间阈值更新子模块；如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则进入确认模块；

死锁时间阈值更新子模块，适于将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值。

可选地，在所述槽位计算模块之前，还包括：

死锁检测结点分配模块，适于对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。

可选地，所述死锁检测结点分配模块，包括：

哈希值获取子模块，适于获取死锁检测结点的内存地址的哈希值；

取余子模块，适于将所述哈希值对时间轮的数量取余数；

死锁检测结点分配子模块，适于根据余数与时间轮的对应关系，将死锁检测结点分配到相应的时间轮中。

可选地，在所述确认模块之后，还包括：

删除通知模块，适于删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁；

或，删除重启模块，适于删除第一队列中相应的死锁检测结点，并

重启客户端对所述应用程序接口的调用请求。

可选地，在所述删除重启模块之前，还包括:

打开请求接收模块，适于接收客户端对重启全局标识的打开请求；所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求；

重启全局标识开启模块，适于根据所述打开请求，打开所述重启全局标识。

可选地，所述删除通知模块，包括:

应用程序接口信息获取子模块，适于获取死锁的应用程序接口的信息；

序列化子模块，适于将所述死锁的应用程序接口的信息序列化为字符串返回给客户端；所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

可选地，在所述创建模块之前，还包括:

时间长度阈值接收模块，适于接收客户端针对所述应用程序接口配置的时间长度阈值。

可选地，所述时间轮的每个槽位配置有自旋锁；所述自旋锁在第一队列被操作之前添加，以及在第一队列被操作完毕之后释放。

可选地，所述第一队列包括双向链表或单向链表。

可选地，当所述第一队列为双向链表时，所述死锁检测结点的数据结构包括:

双向链表结点参数、死锁时间阈值参数、时间轮哈希索引参数、时间轮指针槽位参数。

可选地，所述创建模块，包括:

创建子模块，适于在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

可选地，所述第一队列为优先级队列，则所述插入模块，包括：

优先级计算子模块，适于计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之和；

插入子模块，适于根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列。

可选地，所述监控模块，包括：

截止时间检测模块，适于检测优先级队列首部的死锁检测结点的截止时间是否到达；如果优先级队列首部的检测结点的截止时间到达，则进入确认模块。

本申请实施例包括以下优点：

本申请实施例，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

附图说明

图 1 是本申请的一种应用程序接口死锁监控方法实施例的步骤流程

图;

图 2 是本申请的一种应用程序接口死锁监控方法实施例的步骤流程图;

图 2A 是本申请的一种 API 死锁监测结点在上下文环境中的位置示意图;

图 2B 是本申请的一种死锁检测结点的数据结构示意图;

图 2C 是本申请的一个时间轮的示意图;

图 2D 是本申请的一种双向链表的数据结构示意图;

图 2E 是本申请的一种用双向链表串联 API 死锁监测结点的示意图;

图 2F 是本申请的一种在双向链表中插入死锁检测结点的示意图;

图 2G 是本申请的一种内部状态的数据结构示意图;

图 3 是本申请的一种应用程序接口死锁监控方法实施例的步骤流程图;

图 4 是本申请的一种应用程序接口死锁监控装置实施例的结构框图;

图 5 是本申请的一种应用程序接口死锁监控装置实施例的结构框图;

图 6 是本申请的一种应用程序接口死锁监控装置实施例的结构框图。

具体实施方式

为使本申请的上述目的、特征和优点能够更加明显易懂，下面结合附图和具体实施方式对本申请作进一步详细的说明。

本申请实施例的核心构思之一在于，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死

锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

实施例一

参照图 1，示出了本申请的一种应用程序接口死锁监控方法实施例的步骤流程图，具体可以包括如下步骤:

步骤 110，接收客户端对一应用程序接口的调用请求。

在数据封装时，网络分层中的每个层相互之间会用接口进行交互并提供服务，其中应用层与客户端之间的接口称之为应用程序接口。在实际应用中，客户端可以通过调用一应用程序接口请求系统提供服务。

步骤 120，创建针对所述调用请求的死锁检测结点。

在本申请实施例中，死锁检测结点用于保存每个调用请求的死锁检测的信息。其中，API 死锁是指在分布式系统中，用户调用一 API 请求服务，而系统既不返回成功，也不返回失败，导致用户请求被阻塞的一种现象。

在本申请实施例中，可以先创建一个第一进程，利用该第一进程创建针对调用请求的死锁检测结点。第一进程可以在本步骤之前，或者是本步骤之前的任一步骤之前创建，对此本申请实施例不加以限定。

另外，在本申请实施例中，可以预先设定判定各 API 进入死锁状态的死锁时间阈值，若从开始调用 API 开始，等待的时间超过死锁时间阈值，系统仍未返回任何消息，则可以判定对应 API 死锁。需要说明的是，在实际应用中，各 API 对应的服务不同，所以不同 API 的死锁时间阈值也会不完全相同。各 API 的死锁时间阈值可以在各 API 在被调用之前根据情况设

定，对此本申请实施例不加以限定。

步骤 130，将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。

在本申请实施例中，会创建第一队列，用以存放死锁检测结点。其中，可以根据各 API 对应的死锁时间阈值，将各死锁检测节点按照死锁时间阈值从大的小的顺序插入第一队列。当然，也可以按照其他方式将死锁检测结点插入第一队列，对此本申请实施例不加以限定。

另外，在实际应用中，当调用请求对应用程序接口的调用结束时，此时不再需要对该应用程序接口进行死锁监控，在本申请实施例中，可以利用步骤 120 中的第一进程删除第一队列中相应的死锁检测节点。

在本申请另一优选的实施例中，所述第一队列包括双向链表或单向链表。

其中，双向链表也叫双链表，是链表的一种，它的每个数据结点中都有两个指针，分别指向前一个结点和后一个结点。在本申请实施例中，为了区分起见，可以将两个指针分别称为前向指针和后向指针，其中，前向指针是指向前一个结点的指针，后向指针是指向后一个结点的指针。所以，从双向链表中的任意一个结点开始，都可以很方便地访问它的前一个结点和后一个结点。而且在双向链表中，结点的插入、删除操作只涉及到指针的修改，时间复杂度是 O(1)。其中，O（f（n））是时间复杂度函数，可以定量的描述算法 f（n）的运行时间。

另外，在本申请实施例中，所述第一队列也可以为单向链表，与双向链表相比，单向链表的特点是链表的链接方向是单向的，对链表的访问要通过顺序读取从头部开始，所以，从单向链表中的任意一个结点开始，只可以固定地访问其前一个结点，或固定地访问其后一个结点。

综上分析可知，在本申请实施例中，利用双向链表作为第一队列，效果更好，但是也可以利用优先级队列或单向链表作为第一队列，对此本申请实施例不加以限定。

步骤 140，监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则进入步骤 150。

在本申请实施例中，可以创建一个监控进程，用以监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值。监控进程可以在本步骤之前，或者是本步骤之前的任一步骤之前设定，对此本申请实施例不加以限定。

由前述，在本申请实施例中，在接收到对一应用程序接口的调用请求后，即会创建一个针对该调用请求的死锁检测结点，在调用请求结束后会相应地删除针对该调用请求的死锁检测结点，而若到当前时刻针对该调用请求的死锁检测结点未被删除，则从其创建结束到当前时刻的时间即为该死锁检测结点的存在时间。

步骤 150，确定相应被调用的应用程序接口死锁。

在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

实施例二

参照图 2，示出了本申请的一种应用程序接口死锁监控方法实施例的步骤流程图，具体可以包括如下步骤:

步骤 210，接收客户端对一应用程序接口的调用请求。

步骤 220，在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

在本申请实施例中，会将死锁检测结点创建于对应 API 的上下文环境数据结构中，从而死锁检测结点的生命周期与对应的 API 上下文环境数据结构相同，而且死锁检测结点的内存空间可以分配在上下文环境中，从而避免频繁调用函数为死锁检测结点单独分配、释放内存空间，另外死锁检测结点与上下文环境的内存空间相连续，还可以避免产生内存碎片。另外，不同的 API 上下文环境是不同的，但其中的死锁检测结点的结构是相同的。因此，在 API 上下文环境数据结构中创建死锁检测结点，既可以解决用同一个容器管理所有 API 调用的问题，又解决了方便地获取 API 相关信息的问题。 API 死锁监测结点在上下文环境中的位置如图 2A 所示。

因此，在本申请实施例中，API 上下文环境数据结构包括死锁检测结点以及其他的 API 相关字段。其中，其他的 API 相关字段可以包括与 API 相关的其他的可存在与其上下文环境数据结构中的任何字段，对此本申请实施例不加以限定。

在本申请实施例中，所述死锁检测结点的数据结构如图 2B 所示，包括:

双向链表结点参数、死锁时间阈值参数、时间轮哈希索引参数、时间轮指针槽位参数。

其中，双向链表结点参数是指用以连接当前死锁检测结点的各个双

向链结点的参数，例如各个双向链表结点中的指针等。

死锁时间阈值参数，是指可以接受的等待系统响应的时间阈值参。

时间轮哈希索引参数是指当前死锁检测结点所属时间轮的索引参数，例如时间轮标号等。在本申请实施例中，为了降低检监控进程之间的竞争强度，可以将各死锁检测结点分配到不同的时间轮。具体的分配方法包括哈希（Hash）算法，所谓哈希算法，是一种单向密码体制，即它是一个从明文到密文的不可逆的映射，只有加密过程，没有解密过程。同时，哈希函数可以将任意长度的输入经过变化以后得到固定长度的输出，哈希函数的这种单向特征和输出数据长度固定的特征使得它可以生成消息或者数据。其中，时间轮（Timing Wheel）是一种数据结构，其主体是一个循环列表（circular buffer），每个列表中包含一个称之为槽位（slot）的结构，如图 2C 所示为一个时间轮的示意图。时间轮的工作原理可以类比于时钟，如图 2C 中指针(箭头)按某一个方向按固定频率轮动，每一次跳动称为一个槽位（tick），指针当前所指向的槽位为当前选定的槽位。这样可以看出时间轮有三个重要的属性参数，分别为一整个时间轮的槽位个数、时间轮周期以及每个槽位的持续时间，其中时间轮周期为槽位个数与每个槽位持续时间的乘积。例如，当一整个时间轮的槽位个数为 60，一个槽位的持续时间为 1 秒，这就和现实中时钟的秒针走动完全类似了。

时间轮指针槽位参数是指当前死锁检测结点所在时间轮中的具体槽位参数，例如槽位标号等。

步骤 230，对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。

如前述，在本申请实施例中，需要将各死锁检测结点分配到相应地时间轮。可以通过对各死锁检测结点按照时间轮的个数进行哈希计算，

并根据计算结果将死锁检测结点分配到对应哈希计算结果的时间轮。例如，利用哈希算法计算各死锁检测结点针对某个参数的哈希值，然后基于时间轮的个数，根据计算出来的哈希值，按照一定规则将死锁检测结点分配到对应其哈希值的时间轮。其中，参数可以为可利用的且互相唯一的死锁检测结点的相关参数，例如死锁检测结点的内存地址。具体的规则可以包括：直接寻址法、数字分析法、平方取中法、折叠法、随机数法、除留余数法等。

在本申请另一优选的实施例中，步骤 230 包括:

子步骤 231，获取死锁检测结点的内存地址的哈希值的哈希值。

在本申请实施例中，以死锁检测结点的内存地址为基础，获取死锁检测结点的内存地址的哈希值。具体的计算死锁检测结点的内存地址的哈希值的方法，可以利用可以采用的上述任何一种哈希算法，对此本申请不加以限定。

子步骤 232，将所述哈希值对时间轮的数量取余数。

因为时间轮的数量是一定的，可以根据需求设定。为了实现根据哈希值将死锁检测结点分配到不同的时间轮中，所以哈希值与时间轮之间必然存在一定的对应关注，例如哈希值为某固定值对应于某一固定时间轮，当然也可以为其他的对应关系，对此本申请不加以限定。优选地，可以将哈希值对时间轮的数量取余，获得的余数可以为零到时间轮的数量减一之间的整数，也就是说余数可能的取值的数量等于时间轮的数量，从而可以预置余数与时间轮的对应关系，此时为尽量平均各时间轮中死锁检测结点的数量，避免数据混乱，余数与时间轮是一一对应的，当然也可以存在多个余数对应一个时间轮，或者一个余数对应多个时间轮的情况，对此本申请不加以限定。

子步骤 233，根据余数与时间轮的对应关系，将死锁检测结点分配

到相应的时间轮中。

在确定了余数与时间轮的对应关系并且获取对应死锁检测结点的余数后，则可以相应地将死锁检测结点分配到对应的时间轮中。

步骤 240，根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位。

在本申请实施例中，同一个时间轮中每个槽位的持续时间是一致的。在实际应用中，可以设定时间轮中某一个槽位为起始槽位，例如，本申请实施例可以以当前时间轮指针所在的槽位为初始槽位，则指针从初始槽位移动到初始槽位的下一个槽位需要经过一个持续时间，而若指针从初始槽位移动到初始槽位之后的第 N（N 大于 1，且小于时间轮的槽位个数）个槽位，则需要经过 N 个持续时间。

另外，若死锁时间阈值大于整个时间轮周期，则可以将死锁时间阈值对时间轮周期取余，将取得的余数作为对应该死锁时间阈值的新的死锁时间阈值，然后按照上述方法，计算原始死锁时间阈值对应的死锁检测结点的槽位。

所以在本申请实施例中，可以根据各死锁监测结点对应的死锁时间阈值，计算经过各死锁时间阈值后，对应时间轮的指针从初始槽位可以移动到指向的槽位，则为该死锁时间阈值对应的死锁检测结点所属的槽位。

例如，针对某一时间轮 A，其槽位个数为 10，分别标号为 1-10，每个槽位的持续时间为 2 秒。经步骤 230 后分配到时间轮 A 的死锁检测结点包括死锁检测结点 a、死锁检测结点 b。其中，死锁检测结点 a 的死锁时间阈值为 3 秒，死锁检测结点 b 的思索时间阈值为 6 秒，当前时间轮中指针所在的槽位的标号为 5。

则可以判断出，经过 3 秒后，时间轮的指针从当前槽位移动到其之后

的第二个槽位，因此，死锁检测结点 a 所在的槽位标号为 7；经过 4 秒后，时间轮的指针从当前时间所在的槽位移动到其之后的第三个槽位，此时死锁检测结点 b 所在的槽位标号为 8。

在本申请另一优选的实施例中，所述时间轮的每个槽位配置有自旋（Spin）锁；所述自旋锁在第一队列被操作之前添加，以及在第一队列被操作完毕之后释放。当第一队列为双向链表时，自旋锁在双向链表被操作之前添加，以及在双向链表被操作完毕之后释放。自旋锁的目的是为了保证对链双向链表操作的安全性，在对双向链表进行插入、删除和遍历等操作时，自在该双向链表中添加自旋锁，在执行完对应操作后再释放自旋锁。

步骤 250，将所述死锁检测结点插入所述槽位中的第一队列；所述第一队列为双向链表或单向链表。

在本申请实施例中，第一队列为双向链表或单向链表。其中，双向链表也叫双链表，是链表的一种，它的每个数据结点中都有两个指针，分别指向前一个结点和后一个结点。如图 2D 所示为双向链表的数据结构，图 2E 为用双向链表串联 API 死锁监测结点的示意图。在本申请实施例中，为了区分起见，可以将两个指针分别称为前向指针和后向指针，其中，前向指针是指向前一个结点的指针，后向指针是指向后一个结点的指针。所以，从双向链表中的任意一个结点开始，都可以很方便地访问它的前一个结点和后一个结点。而且在双向链表中，结点的插入、删除操作只涉及到指针的修改，时间复杂度是 O(1)。其中，O（f（n））是时间复杂度函数，可以定量的描述算法 f（n）的运行时间。

图 2F 为在双向链表中插入死锁检测结点的示意图。其中，假设双向链表有五个结点，其中虚线部分为地址值，地址值是为了描述方便随机设定的值，在实际应用中，可以根据情况设定其他值，对此本申请实施

例不加以限定。这些地址值都会被放到某个变量当中，只要对变量进行赋值传递就能实现链表的构建。从图 2F 中很容易看出蓝色的箭头组成了一个单链表，红色的箭头又组成了一个单链表（逆向的）。设 p 指向双向链表中某结点，s 指向待插入的新结点，将*s 插入到*p 的前面，插入时需要更改两个指针变量。如图 2F 所示，将 q 结点中原本指向 p 的后向指针指向 s 结点，同时将 s 结点的后向指针指向 p 结点的，在反向上，将 p 结点中原本指向 q 结点的前向指针指向 s 结点的，将 s 结点的前向指针指向 q 结点，从而实现了将 s 结点插入到双向链表的 p 结点之前。

在本申请实施例中，死锁检测结点相当于图 2F 中所述的结点的一种。因此，将死锁检测结点插入所述槽位中的双向链表的具体过程与前述一致。

另外，在本申请另一个优选的实施例中，所述第一队列也可以为单向链表，与双向链表相比，单向链表的特点是链表的链接方向是单向的，对链表的访问要通过顺序读取从头部开始，所以，从单向链表中的任意一个结点开始，只可以固定地访问其前一个结点，或固定地访问其后一个结点。相较于单向链表，利用双向链表作为第一队列，效果更好，但是也可以利用单向链表作为第一队列，对此本申请实施例不加以限定。

步骤 260，定期将时间轮的指针移动到下一个槽位。

如前述，在实际应用中，时间轮的每个槽位具有一定的持续时间，从指针开始在某一槽位时刻开始，经过一个固定时间，指针则会移动到该槽位的后一个槽位。在本申请实施例中，可以创建一个监控进程，按照每个槽位的持续时间为单位，定期地将时间轮的指针移动到下一个槽位。

步骤 270，对于指针指向的槽位，判断所述槽位中的第一队列的各死

锁检测结点的死锁时间阈值是否超过一个时间轮周期；如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则进入步骤 280；如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则进入步骤 290。

如前述，在本申请实施例中，时间轮中每个槽位中所包含的死锁检测结点对应的死锁时间阈值可能大于一个时间轮周期，对于此类死锁检测检点，可能需要经过至少一个时间轮周期后，才可以判断出其存在时间是否超过死锁时间阈值，相对而言，会耗费比较多的时间。所以在本申请实施例中，对于指针指向的槽位，会判断槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期，若一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值，然后等待进入下一个时间轮周期，再执行步骤 270 进行判断。

在本申请实施例中，本步骤仍可以由上述的监控进程执行。

步骤 280，将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值。

在本申请实施例中，本步骤可以由上述的监控进程执行。

步骤 290，确定相应被调用的应用程序接口死锁。

在本申请另一优选的实施例中，在步骤 290 之后，还包括：

步骤 2110，删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁。

对于确认为死锁的应用程序接口，不需要继续的执行上述步骤再次判断，所以会直接删除第一队列中相应的死锁监测结点，在本申请实施例中，可以创建一个执行删除操作的进程，相应地删除第一队列中对应确认为死锁的应用程序接口的死锁检测结点。另外，在本申请实施例

中，为了避免给调用该应用程序接口的客户端造成不必要的损失，例如客户端持续等待应用程序接口响应而浪费时间等。在本申请实施例中，还可以通知调用请求所对应的客户端对所述应用程序接口的调用死锁，具体地，可以利用任何一种现有技术实现通知调用请求所对应的客户端对所述应用程序接口的调用死锁，对此本申请实施例不加以限定。

或，步骤 2120，删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求。

在本申请实施例中，对于已经确认为死锁的应用程序接口，在删除第一队列中相应的死锁检测结点的同时，还可以重新启动客户端对该应用程序接口的调用请求，然后再次执行步骤 210 以及后续的步骤。

在实际应用中，可以调用 abort 函数强制终止对死锁应用程序接口调用的进程，然后系统会重新启动被强制终止的该进程，进而会重新启动客户端对死锁的应用程序接口的调用请求，避免死锁对上层业务造成严重影响。

需要说明的是，上述的利用 abort 函数的调用请求重启方式是一种比较激进的故障恢复方式，在本申请实施例中，可以默认该重启方式不开启，当然也可以根据需求默认开启，对此本申请实施例不加以限定。

在本申请另一优选的实施例中，在步骤 2120 之前，还包括:

步骤 2130，接收客户端对重启全局标识的打开请求；所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求。

在本申请实施例中，可以提供一个重启全局标志，用于指示当发生 API 死锁时，直接重新启动该用户进程。其中，用户进程中包括了对死锁 API 的调用，因此，此时重启全局标识可以重启客户端对发生死锁的应用程序接口的调用请求。

此时在本申请实施例中，在执行步骤 2120 之前，首先会接收客户端对重启全局标识的打开请求。

步骤 2140，根据所述打开请求，打开所述重启全局标识。

如步骤 2130 所述，在根据所述打开请求，打开重启全局标识，则会重启客户端对发生死锁的应用程序接口的调用请求。

具体地，可以通过调用 abort 函数，对调用了死锁 API 的用户进程进行关闭。而在系统中，当一用户进程被异常关闭时，在其他进程的作用下，也即可以设置一个监控异常关闭的第一进程，该第一进程监控到对应客户端的用户进程关闭后，将该异常关闭的用户进程重新启动。进而可以实现重启客户端对发生死锁的应用程序接口的调用请求。

在本申请另一优选的实施例中，步骤 2110，包括:

子步骤 2111，获取死锁的应用程序接口的信息。

在本申请实施例中，对于步骤 2110 中所述的通知调用请求所对应的客户端对所述应用程序接口的调用死锁，首先，需要获取死锁的应用程序的信息，例如应用程序接口的名称，功能等等。

子步骤 2112，将所述死锁的应用程序接口的信息序列化为字符串返回给客户端；所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

在本申请实施例中，对于获取到的死锁的应用程序接口的信息，可以将其序列化为字符串后再返回值客户端。其中，序列化（Serialization）是将对象的状态信息转换为可以存储或传输的形式的过程。在本申请实施例中，对象的状态信息包括死锁的应用程序接口的信息，转换的形式为字符串。

相应的，在客户端接收到上述字符串后，会对字符串进行反序列化，即执行序列化的逆过程，将字符串恢复为未序列化之前的死锁的应用程

序接口信息。

在本申请实施例中，客户端还可以调用死锁API信息反馈接口，通过死锁 API 信息反馈接口将死锁的 API 的信息反馈给客户端。该死锁 API 信息反馈接口反馈的信息是：每种类型的API在当前时刻死锁的次数。该死锁API信息反馈接口按如下形式定义：

返回码类型　　函数名(序列化字符串)

其中，返回码类型是一个整数，标示本次查询成功或失败的原因；序列化字符串是输出参数，其将死锁API的信息的序列化字符串输出，如果没有API死锁，则为空串。死锁API信息反馈接口返回后，客户端检查到返回码成功，并且字符串不为空时，说明存在API死锁，则可以先将字符串反序列化，得到一个查询系统内部状态的数据结构，该内部状态的数据结构如图 2G，包含一个死锁监测结果 map<API 名，死锁次数>,该 map 的key是API函数名，value是当前时刻死锁次数。另外，考虑到死锁API信息反馈接口的可扩展性，如果有新增的功能时，可以在图 2G 的数据结构中，增加字段“用户传入参数”或“传出结果”用于扩展功能，而对它序列化成字符串可以做到不改变函数接口，保证兼容性。

在本申请另一优选的实施例中，在步骤220之前，还包括：

步骤 2150，接收客户端针对所述应用程序接口配置的时间长度阈值。

在实际应用中，根据各个应用程序接口不同，以及调用各应用程序接口的客户端不同等原因，各应用程序接口的时间长度阈值可能会有所不同。所以在本申请实施例中，在步骤220之前，可以接收客户端针对应用程序接口配置的时间长度阈值。

在本申请实施例中，接收到的时间长度阈值即为后续的死锁检测结

点所包括的死锁时间阈值。

在本申请实施例中，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

另外，在本申请中，基于死锁时间阈值，时间轮周期以及槽位个数等参数，将死锁检测结点按照一定规则插入到位于不同时间轮的槽位中的双向链表或单向链表中，鉴于双向链表和单向链表插入和删除的便捷特性，同时可以预置多个时间轮同时运行，从而可以进一步地提高死锁监控的效率以及有效性。进一步地，双向链表中的结点可以访问与之相邻的前后两个方向上的结点，而单向链表中的结点只可以访问与之相邻的前向结点或者是与之相邻的后向结点，因此相对而言，利用双向链表作为第一队列比单向链表的效率会更高。

实施例三

参照图 3，示出了本申请的一种应用程序接口死锁监控方法实施例的步骤流程图，具体可以包括如下步骤:

步骤 310，接收客户端对一应用程序接口的调用请求。

步骤 320，创建针对所述调用请求的死锁检测结点。

步骤 330，计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之

和。

在本申请实施例中，用以存储死锁检测结点的第一队列为优先级队列。其中，优先级队列(priority queue) 是 0 个或多个元素的集合，每个元素都有一个优先级，对于优先级相同的元素，可按先进先出次序处理或按任意优先级进行。对于优先级队列而言，只可以按照优先级顺序依次访问其中的各个元素，每次从优先级队列中取出的是当前优先级队列中具有最高优先级的元素。优先级队列是将元素按照优先级顺序进行存储的队列，所以需要计算死锁检测结点的优先级。在本申请中，以截止时间作为死锁检测结点的优先级，截止时间越短，优先级越高。其中，截止时间为检测结点创建时间与死锁时间阈值之和。当然，考虑到创建时间区别不明显，截止时间也可以只包括死锁时间阈值，对此本申请不加以限定。

所以在本发明实施例中，若要将死锁检测结点插入到优先级队列中，首先需要计算死锁检测结点的优先级，然后根据优先级，将该死锁检测结点插入到优先级队列中。例如，若当前死锁监测结点的优先级高于优先级队列中全部的死锁监测结点的优先级，则可以将当前死锁监测结点置于优先级队列的首部；若当前死锁监测结点的优先级未高于优先级队列中全部的死锁监测结点的优先级，则将该死锁检测结点插入到优先级队列中，优先级大于当前死锁检测结点的各死锁检测结点中优先级最小的死锁检测结点之后。

步骤 340，根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列。

对于优先级队列，用户请求调用一 API 时，会在优先级队列中插入对应的死锁检测结点，退出 API 时，在优先级队列中删除对应的死锁检测结点。对优先级队列的插入、删除操作，时间复杂度是 O(logN)，N 是系统

正在处理的请求数目。

因为优先级队列是按照优先级进行排序存储的，在获取了死锁检测结点的优先级后，根据其优先级，将死锁检测结点插入到其所属槽位中的优先级队列中高于该死锁检测结点的优先级且优先级最小的死锁检测结点之后。需要注意的是，若在优先级队列中插入一个死锁检测结点，此时在优先级小于该死锁检测结点在优先级队列中的存储位置都需要相应地下移一个死锁检测结点的位置。因此，在将多个死锁检测结点插入优先级队列的过程中，已存入优先级队列中的死锁检测结点的位置可能会被多次调整。所以相对而言，利用优先级队列存放死锁检测结点的方法相对于实施例二中利用双向链表存放死锁检测结点的方法的效率会低一些。

另外，在本申请的另一优选的实施例中，也可以按照利用多个优先级队列存储死锁检测结点。那么，首先需要利用前述的哈希算法，将死锁检测结点分配到不同的优先级队列，然后进一步执行步骤 330，然后按照本步骤根据死锁检测结点的优先级，将死锁检测结点插入其所属优先级队列中的对应位置。

需要说明的是，若死锁检测结点对应的应用程序接口在截止时间内响应，则在优先级列表中会自动删除该死锁检测结点。

步骤 350，检测优先级队列首部的死锁检测结点的截止时间是否到达；如果优先级队列首部的检测结点的截止时间到达，则进入步骤 360。

如前述，优先级队列中的优先级越高的死锁检测结点的截止时间越短，所以，检测优先级队列首部的死锁检测结点的截止时间是否到达，如果优先级队列首部的死锁检测结点的截止时间到达，则可以确定相应被调用的应用程序接口死锁，然后可以将该死锁检测结点从优先级队列中删除。而若优先级队列首部的死锁检测结点的截止时间未到达，则继

续等待直到该死锁检测结点被删除或该死锁检测结点的截止时间到达。

在判断完当前优先级队列首部的死锁检测结点的截止时间是否到达后，该死锁检测结点可以从优先级队列中删除，然后其后一位死锁检测结点则成为优先级队列首部的死锁检测结点，然后继续执行本步骤。

步骤360，确定相应被调用的应用程序接口死锁。

需要说明的是，对于方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本申请实施例并不受所描述的动作顺序的限制，因为依据本申请实施例，某些步骤可以采用其他顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于优选实施例，所涉及的动作并不一定是本申请实施例所必须的。

在本申请实施例中，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后计算死锁检测结点的优先级，并根据死锁检测结点的优先级将死锁检测结点插入优先级队列中，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对API的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

另外，在本申请实施例中，按照死锁检测结点的截止时间，获取各死锁检测结点的优先级，然后将各死锁检测结点按照优先级从高到低的顺序插入优先级队列中，同样可以进一步提高死锁检测的精确性以及效率，但是相对而言，实施例二所述的利用双向链表或者单向链表存放死

锁检测结点的死锁检测方案比本实施例的效率更高。

实施例四

参照图 4，示出了本申请的一种应用程序接口死锁监控装置实施例的结构框图，具体可以包括如下模块：

接收模块 410，适于接收客户端对一应用程序接口的调用请求。

创建模块 420，适于创建针对所述调用请求的死锁检测结点。

插入模块 430，适于将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。

监控模块 440，适于监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则进入确认模块 450。

确认模块 450，适于确定相应被调用的应用程序接口死锁。

在本申请实施例中，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

实施例五

参照图 5，示出了本申请的一种应用程序接口死锁监控装置实施例的结构框图，具体可以包括如下模块：

接收模块 510，适于接收客户端对一应用程序接口的调用请求。

创建模块 520，适于创建针对所述调用请求的死锁检测结点。具体包括:

创建子模块 521，适于在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

死锁检测结点分配模块 530，适于对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。具体包括:

哈希值获取子模块 531，适于获取死锁检测结点的内存地址的哈希值。

取余子模块 532，适于将所述哈希值对时间轮的数量取余数。

死锁检测结点分配子模块 533，适于根据余数与时间轮的对应关系，将死锁检测结点分配到相应的时间轮中。

槽位计算模块 540，适于根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位。

插入模块 550，适于将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。具体包括:

第一插入子模块 551，适于将所述死锁检测结点插入所述槽位中的第一队列。

监控模块 560，适于监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则进入确认模块。具体包括:

指针移动子模块 561，适于定期将时间轮的指针移动到下一个槽位。

死锁时间判断子模块 562，适于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期；如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则进入死锁时间阈值更新子模块；如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则进入确认模块。

死锁时间阈值更新子模块 563，适于将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值。

确认模块 570，适于确定相应被调用的应用程序接口死锁。

在本申请另一优选的实施例中，在确认模块 570 之后，还包括：

删除通知模块 580，适于删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁。

或，删除重启模块 590，适于删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求。

在本申请另一优选的实施例中，在删除通知模块 580 之前，还包括：

打开请求接收模块 590，适于接收客户端对重启全局标识的打开请求；所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求。

重启全局标识开启模块 5110，适于根据所述打开请求，打开所述重启全局标识。

在本申请另一优选的实施例中，所述删除通知模块 580，包括：

应用程序接口信息获取子模块 581，适于获取死锁的应用程序接口的信息。

序列化子模块 582，适于将所述死锁的应用程序接口的信息序列化为

字符串返回给客户端；所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

在本申请另一优选的实施例中，在创建模块之前，还包括：

时间长度阈值接收模块 5120，适于接收客户端针对所述应用程序接口配置的时间长度阈值。

在本申请实施例中，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后将死锁检测结点插入第一队列，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

另外，在本申请中，基于死锁时间阈值，时间轮周期以及槽位个数等参数，将死锁检测结点按照一定规则插入到位于不同时间轮的槽位中的双向链表或单向链表中，鉴于双向链表和单向链表插入和删除的便捷特性，同时可以预置多个时间轮同时运行，从而可以进一步地提高死锁监控的效率以及有效性。进一步地，双向链表中的结点可以访问与之相邻的前后两个方向上的结点，而单向链表中的结点只可以访问与之相邻的前向结点或者是与之相邻的后向结点，因此相对而言，利用双向链表作为第一队列比单向链表的效率会更高。

实施例六

参照图 6，示出了本申请的一种应用程序接口死锁监控装置实施例的结构框图，具体可以包括如下模块：

接收模块 610，适于接收客户端对一应用程序接口的调用请求。

创建模块 620，适于创建针对所述调用请求的死锁检测结点。

插入模块 630，适于将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。在本申请实施例中，所述第一队列为优先级队列，则插入模块 630，具体包括：

优先级计算子模块 631，适于计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之和。

第二插入子模块 632，适于根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列。

监控模块 640，适于监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则进入确认模块 650。具体包括：

截止时间检测模块 641，适于检测优先级队列首部的死锁检测结点的截止时间是否到达；如果优先级队列首部的检测结点的截止时间到达，则进入确认模块 650。

确认模块 650，适于确定相应被调用的应用程序接口死锁。

在本申请实施例中，在接收客户端对一应用程序接口的调用请求后会创建针对调用请求的死锁检测结点，然后计算死锁检测结点的优先级，并根据死锁检测结点的优先级将死锁检测结点插入优先级队列中，其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点。进一步监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值，如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。从而可以基于

死锁监控结点，通过监控利用死锁监控结点的存在时间，实现对 API 的死锁监控，相对于背景技术中的死锁检测方法，本申请所述的应用程序接口死锁监控方法适用性更强，而且精确度较高。

另外，在本申请实施例中，按照死锁检测结点的截止时间，获取各死锁检测结点的优先级，然后将各死锁检测结点按照优先级从高到低的顺序插入优先级队列中，同样可以进一步提高死锁检测的精确性以及效率，但是相对而言，实施例二所述的利用双向链表或者单向链表存放死锁检测结点的死锁检测方案比本实施例的效率更高。

对于装置实施例而言，由于其与方法实施例基本相似，所以描述的比较简单，相关之处参见方法实施例的部分说明即可。

本说明书中的各个实施例均采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似的部分互相参见即可。

本领域内的技术人员应明白，本申请实施例的实施例可提供为方法、装置、或计算机程序产品。因此，本申请实施例可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请实施例可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

在一个典型的配置中，所述计算机设备包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。内存可能包括计算机可读介

质中的非永久性存储器，随机存取存储器(RAM) 和/或非易失性内存等形式，如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存 (PRAM)、静态随机存取存储器 (SRAM)、动态随机存取存储器 (DRAM)、其他类型的随机存取存储器 (RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带，磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括非持续性的电脑可读媒体(transitory media)，如调制的数据信号和载波。

本申请实施例是参照根据本申请实施例的方法、终端设备(系统)、和计算机程序产品的流程图和／或方框图来描述的。应理解可由计算机程序指令实现流程图和／或方框图中的每一流程和／或方框、以及流程图和／或方框图中的流程和／或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理终端设备的处

理器以产生一个机器，使得通过计算机或其他可编程数据处理终端设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和／或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理终端设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和／或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理终端设备上，使得在计算机或其他可编程终端设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程终端设备上执行的指令提供用于实现在流程图一个流程或多个流程和／或方框图一个方框或多个方框中指定的功能的步骤。

尽管已描述了本申请实施例的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例做出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本申请实施例范围的所有变更和修改。

最后，还需要说明的是，在本文中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺

序。而且，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、物品或者终端设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、物品或者终端设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、物品或者终端设备中还存在另外的相同要素。

以上对本申请所提供的一种应用程序接口死锁监控方法和一种应用程序接口死锁监控装置，进行了详细介绍，本文中应用了具体个例对本申请的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本申请的方法及其核心思想；同时，对于本领域的一般技术人员，依据本申请的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本申请的限制。

权 利 要 求 书

1、一种应用程序接口死锁监控方法，其特征在于，包括:

接收客户端对一应用程序接口的调用请求;

创建针对所述调用请求的死锁检测结点;

将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点;

监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值;

如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则确定相应被调用的应用程序接口死锁。

2、根据权利要求 1 所述的方法，其特征在于，所述将所述死锁检测结点插入第一队列的步骤之前，还包括:

根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位;

则所述将所述死锁检测结点插入第一队列的步骤包括：将所述死锁检测结点插入所述槽位中的第一队列。

3、根据权利要求 2 所述的方法，其特征在于，所述监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值的步骤，包括:

定期将时间轮的指针移动到下一个槽位;

对于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期;

如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值;

如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则确定相应被调用的应用程序接口死锁。

4、根据权利要求 2 或 3 所述的方法，其特征在于，在根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位的步骤之前，还包括:

对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。

5、根据权利要求 4 所述的方法，其特征在于，所述对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应的时间轮的步骤，包括:

获取死锁检测结点的内存地址的哈希值;

将所述哈希值对时间轮的数量取余数;

根据余数与时间轮的对应关系，将死锁检测结点分配到相应的时间轮中。

6、根据权利要求 1 所述的方法，其特征在于，在确定相应被调用的应用程序接口死锁的步骤之后，还包括:

删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁;

或，删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求。

7、根据权利要求 6 所述的方法，其特征在于，所述删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求的步骤之前，还包括:

接收客户端对重启全局标识的打开请求；所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求;

根据所述打开请求，打开所述重启全局标识。

8、根据权利要求 6 所述的方法，其特征在于，所述通知调用请求所对应的客户端对所述应用程序接口的调用死锁的步骤，包括:

获取死锁的应用程序接口的信息;

将所述死锁的应用程序接口的信息序列化为字符串返回给客户端；所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

9、根据权利要求 1 所述的方法，其特征在于，在创建针对所述调用请求的死锁检测结点之前，还包括:

接收客户端针对所述应用程序接口配置的时间长度阈值。

10、根据权利要求 2 所述的方法，其特征在于，所述时间轮的每个槽

位配置有自旋锁；所述自旋锁在第一队列被操作之前添加，以及在第一队列被操作完毕之后释放。

11、根据权利要求 1-10 其中之一所述的方法，其特征在于，

所述第一队列包括双向链表或单向链表。

12、根据权利要求 11 所述的方法，其特征在于，

当所述第一队列为双向链表时，所述死锁检测结点的数据结构包括:

双向链表结点参数、死锁时间阈值参数、时间轮哈希索引参数、时间轮指针槽位参数。

13、根据权利要求 12 所述的方法，其特征在于，所述创建针对所述调用请求的死锁检测结点的步骤，包括:

在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

14、根据权利要求 1 所述的方法，其特征在于，

所述第一队列为优先级队列，则所述将所述死锁检测结点插入第一队列的步骤包括:

计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之和;

根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列。

15、根据权利要求 14 所述的方法，其特征在于，所述监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值的步骤，包括:

检测优先级队列首部的死锁检测结点的截止时间是否到达;

如果优先级队列首部的检测结点的截止时间到达，则确定相应被调用的应用程序接口死锁。

16、一种应用程序接口死锁监控装置，其特征在于，包括:

接收模块，适于接收客户端对一应用程序接口的调用请求;

创建模块，适于创建针对所述调用请求的死锁检测结点;

插入模块，适于将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点;

监控模块，适于监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值；如果检测到任一死锁检测结点的存在时间超过死锁时间阈值，则进入确认模块;

确认模块，适于确定相应被调用的应用程序接口死锁。

17、根据权利要求 16 所述的装置，其特征在于，在所述插入模块之前，还包括:

槽位计算模块，适于根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位;

则所述插入模块包括：插入子模块，适于将所述死锁检测结点插入

所述槽位中的第一队列。

18、根据权利要求 17 所述的装置，其特征在于，所述监控模块，包括:

指针移动子模块，适于定期将时间轮的指针移动到下一个槽位;

死锁时间判断子模块，适于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期；如果一死锁检测结点的当前的死锁时间阈值超过一个时间轮周期，则进入死锁时间阈值更新子模块；如果一死锁检测结点的当前的死锁时间阈值不超过一个时间轮周期，则进入确认模块;

死锁时间阈值更新子模块，适于将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值。

19、根据权利要求 17 或 18 所述的装置，其特征在于，在所述槽位计算模块之前，还包括:

死锁检测结点分配模块，适于对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮。

20、根据权利要求 19 所述的装置，其特征在于，所述死锁检测结点分配模块，包括:

哈希值获取子模块，适于获取死锁检测结点的内存地址的哈希值;

取余子模块，适于将所述哈希值对时间轮的数量取余数;

死锁检测结点分配子模块，适于根据余数与时间轮的对应关系，将

死锁检测结点分配到相应的时间轮中。

21、根据权利要求 16 所述的装置，其特征在于，在所述确认模块之后，还包括:

删除通知模块，适于删除第一队列中相应的死锁检测结点，并通知调用请求所对应的客户端对所述应用程序接口的调用死锁;

或，删除重启模块，适于删除第一队列中相应的死锁检测结点，并重启客户端对所述应用程序接口的调用请求。

22、根据权利要求 21 所述的装置，其特征在于，在所述删除重启模块之前，还包括:

打开请求接收模块，适于接收客户端对重启全局标识的打开请求;所述重启全局标识允许在确定相应被调用的应用程序接口死锁后，重启客户端对所述应用程序接口的调用请求;

重启全局标识开启模块，适于根据所述打开请求，打开所述重启全局标识。

23、根据权利要求 21 所述的装置，其特征在于，所述删除通知模块，包括:

应用程序接口信息获取子模块，适于获取死锁的应用程序接口的信息;

序列化子模块，适于将所述死锁的应用程序接口的信息序列化为字符串返回给客户端；所述客户端接收到所述字符串后，对所述字符串进行反序列化，得到死锁的应用程序接口的信息。

24、根据权利要求 16 所述的装置，其特征在于，在所述创建模块之前，还包括:

时间长度阈值接收模块，适于接收客户端针对所述应用程序接口配置的时间长度阈值。

25、根据权利要求 17 所述的装置，其特征在于，所述时间轮的每个槽位配置有自旋锁；所述自旋锁在第一队列被操作之前添加，以及在第一队列被操作完毕之后释放。

26、根据权利要求 16-25 其中之一所述的装置，其特征在于，

所述第一队列包括双向链表或单向链表。

27、根据权利要求 26 所述的装置，其特征在于，

当所述第一队列为双向链表时，所述死锁检测结点的数据结构包括:

双向链表结点参数、死锁时间阈值参数、时间轮哈希索引参数、时间轮指针槽位参数。

28、根据权利要求 27 所述的装置，其特征在于，所述创建模块，包括:

创建子模块，适于在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点。

29、根据权利要求 26 所述的装置，其特征在于，

所述第一队列为优先级队列，则所述插入模块，包括:

优先级计算子模块，适于计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之和；

插入子模块，适于根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列。

30、根据权利要求 29 所述的装置，其特征在于，所述监控模块，包括：

截止时间检测模块，适于检测优先级队列首部的死锁检测结点的截止时间是否到达；如果优先级队列首部的检测结点的截止时间到达，则进入确认模块。

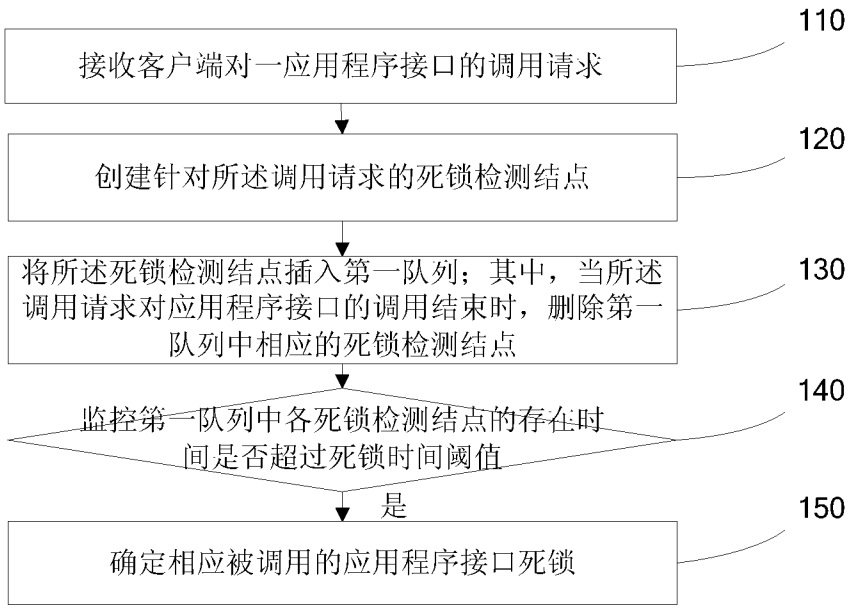

接收客户端对一应用程序接口的调用请求
110
创建针对所述调用请求的死锁检测结点
120
将所述死锁检测结点插入第一队列；其中，当所述调用请求对应用程序接口的调用结束时，删除第一队列中相应的死锁检测结点
130
监控第一队列中各死锁检测结点的存在时间是否超过死锁时间阈值
140
是
确定相应被调用的应用程序接口死锁
150

图 1

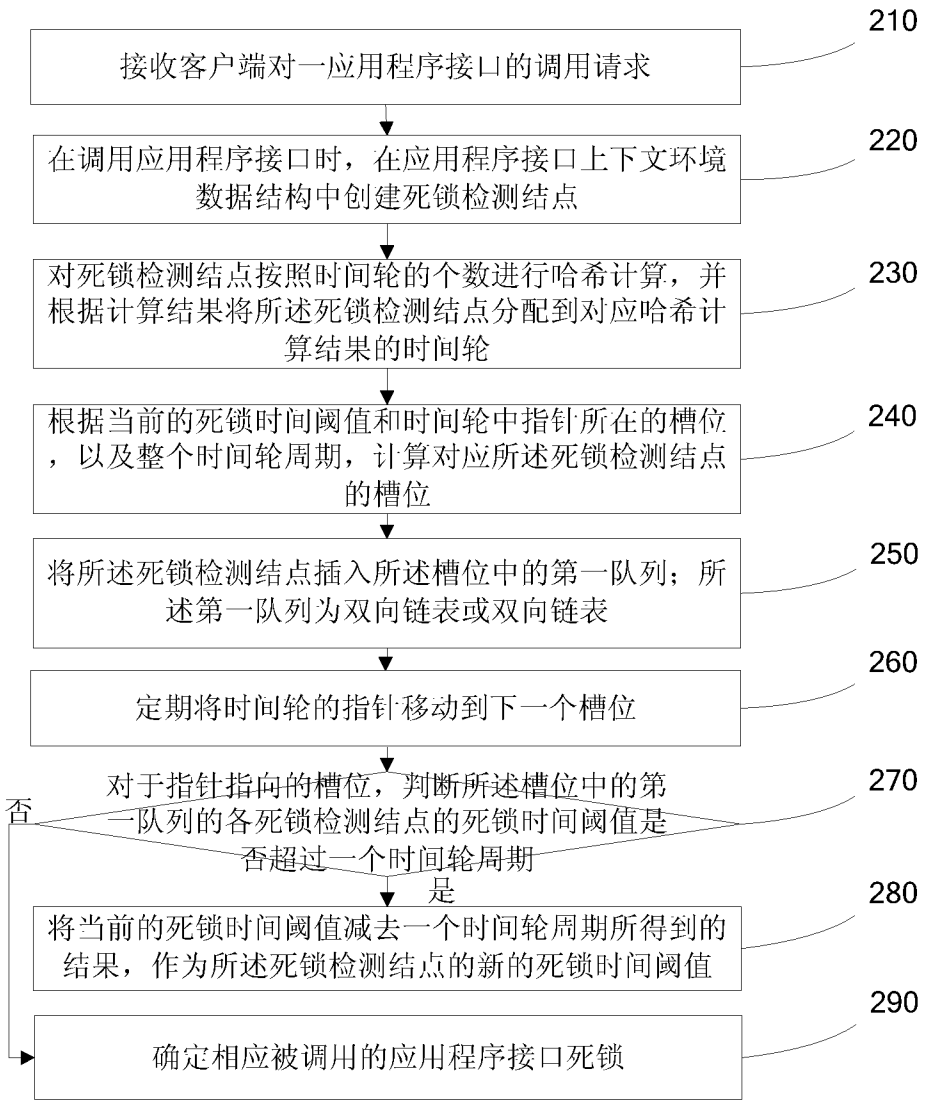

接收客户端对一应用程序接口的调用请求
210
在调用应用程序接口时，在应用程序接口上下文环境数据结构中创建死锁检测结点
220
对死锁检测结点按照时间轮的个数进行哈希计算，并根据计算结果将所述死锁检测结点分配到对应哈希计算结果的时间轮
230
根据当前的死锁时间阈值和时间轮中指针所在的槽位，以及整个时间轮周期，计算对应所述死锁检测结点的槽位
240
将所述死锁检测结点插入所述槽位中的第一队列；所述第一队列为双向链表或双向链表
250
定期将时间轮的指针移动到下一个槽位
260
对于指针指向的槽位，判断所述槽位中的第一队列的各死锁检测结点的死锁时间阈值是否超过一个时间轮周期
270
否
是
将当前的死锁时间阈值减去一个时间轮周期所得到的结果，作为所述死锁检测结点的新的死锁时间阈值
280
确定相应被调用的应用程序接口死锁
290

图 2

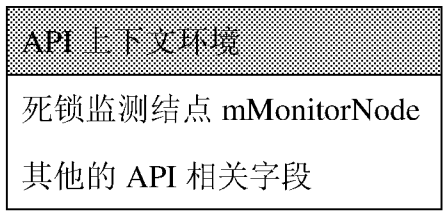

API 上下文环境
死锁监测结点 mMonitorNode
其他的 API 相关字段

图 2A

API 死锁检测结点 MonitorNode
双向链表结点参数 mListNode
死锁时间阈值参数 mThreshold
时间轮哈希索引参数 mWheelHash
时间轮指针槽位参数 mSecondSlot

图 2B

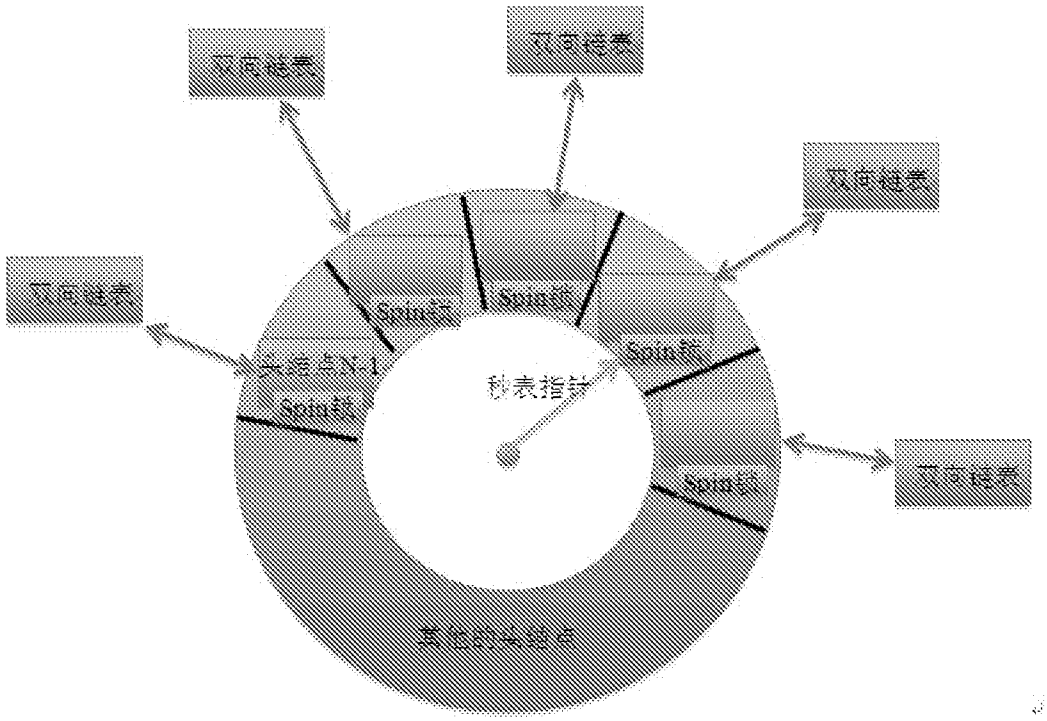


图 2C

双向链表结点
前向指针 mPrev
后向指针 mNext

图 2D

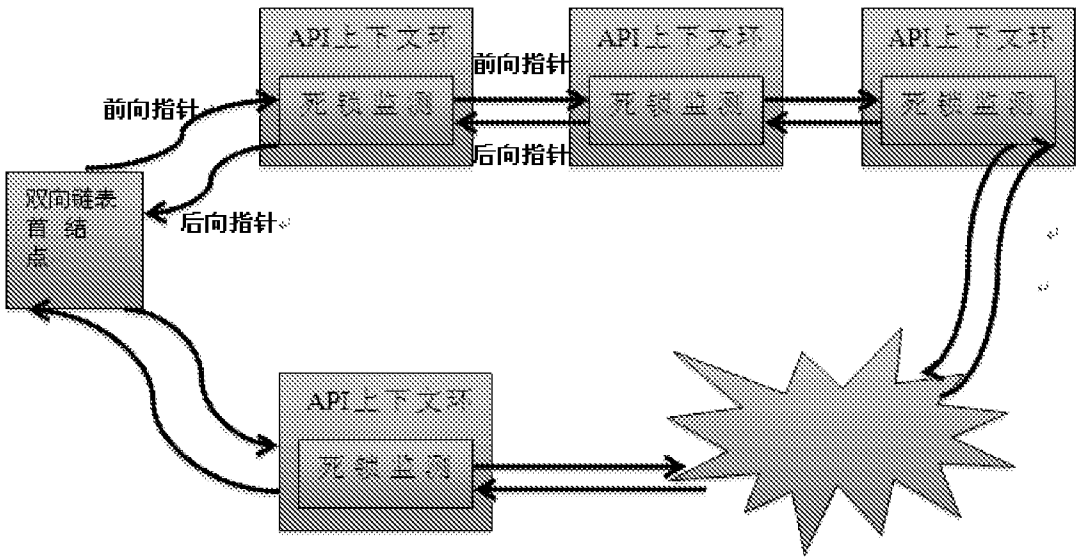


图 2E

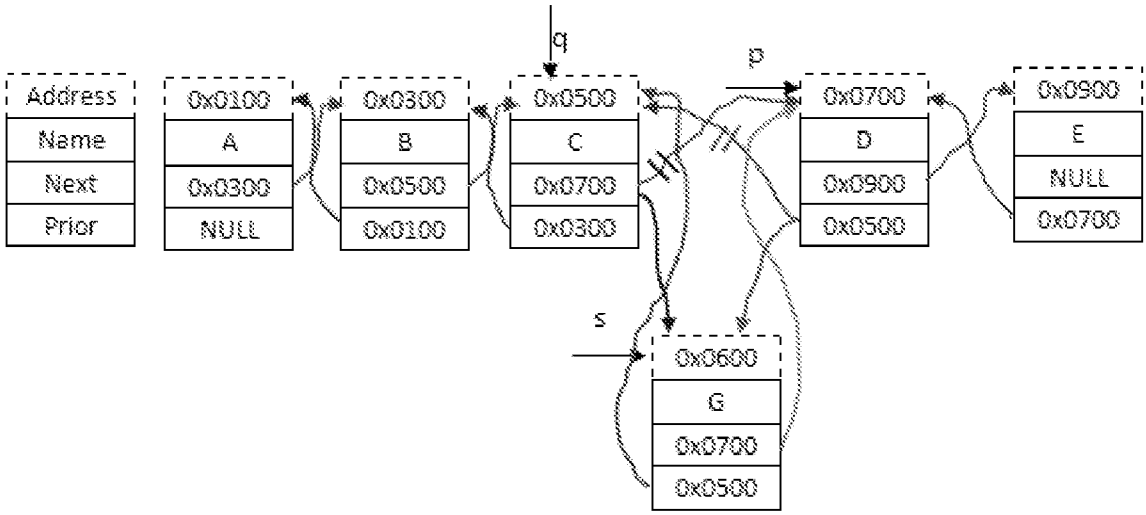


图 2F

系统内部状态
死锁监测结果 map<API 名，死锁次数>
用户传入参数（用于功能扩展）
其他传出结果（用于功能扩展）

图 2G

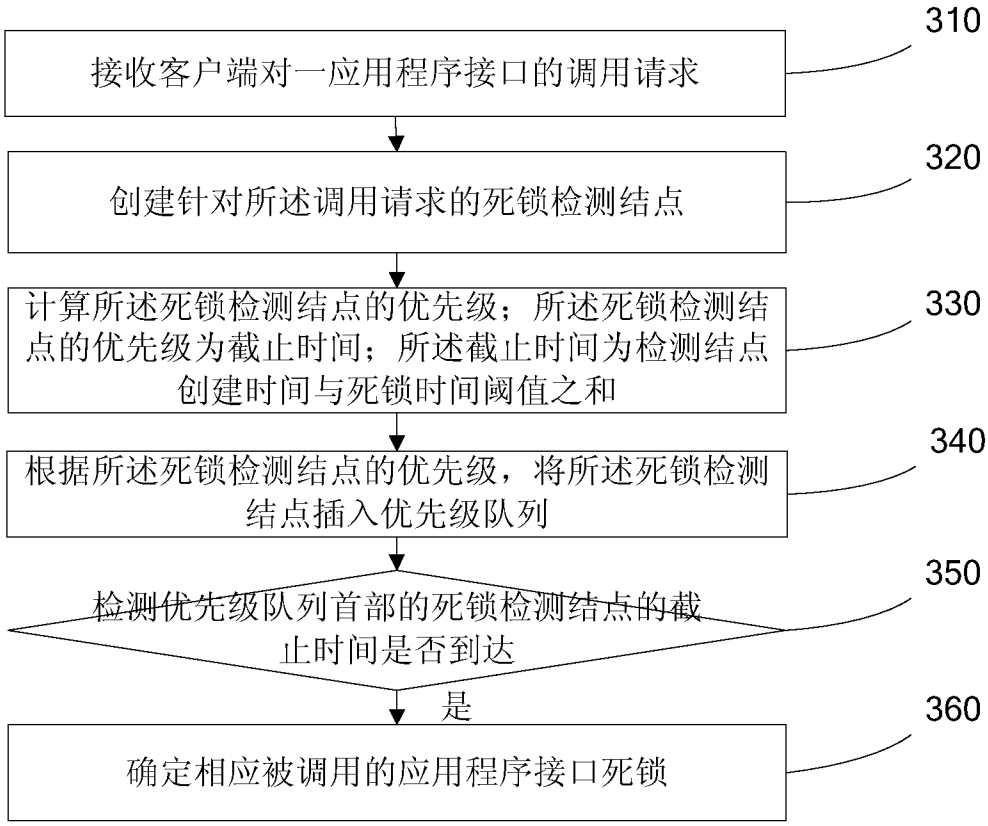

接收客户端对一应用程序接口的调用请求
310
创建针对所述调用请求的死锁检测结点
320
计算所述死锁检测结点的优先级；所述死锁检测结点的优先级为截止时间；所述截止时间为检测结点创建时间与死锁时间阈值之和
330
根据所述死锁检测结点的优先级，将所述死锁检测结点插入优先级队列
340
检测优先级队列首部的死锁检测结点的截止时间是否到达
350
是
确定相应被调用的应用程序接口死锁
360

图 3

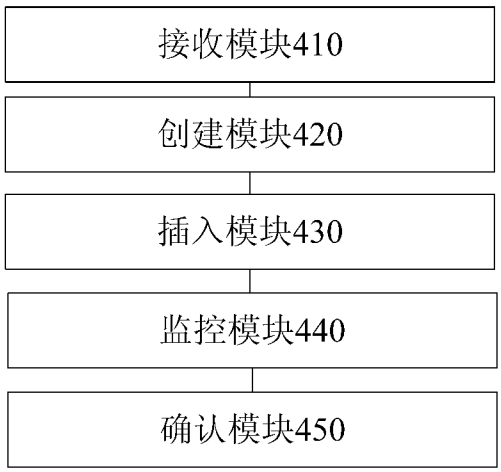

接收模块410
创建模块420
插入模块430
监控模块440
确认模块450

图 4


接收模块510
创建模块520
创建子模块521
死锁检测结点分配模块530
哈希值获取子模块531
取余子模块532
死锁检测结点分配子模块533
槽位计算模块540
插入模块550
第一插入子模块551
监控模块560
指针移动子模块561
死锁时间判断子模块562
死锁时间阈值更新子模块563
确认模块570

图 5


接收模块610
创建模块620
插入模块630
优先级计算子模块631
第二插入子模块632
监控模块640
截止时间检测模块641
确认模块650

图 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2017/077448

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/52 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNKI, CNPAT, WPI, EPODOC: function, process, thread, insertion, duration, API, application programming interface, deadlock, node?, threshold, time

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 102222015 A (SAMSUNG ELECTRONICS (CHINA) R&D CENTER et al.), 19 October 2011 (19.10.2011), description, paragraphs [0025]-[0028], and figure 1	1-30
A	CN 104636259 A (XIAMEN YAXON NETWORKS CO., LTD.), 20 May 2015 (20.05.2015), the whole document	1-30
A	CN 103246552 A (TENCENT TECHNOLOGY (SHENZHEN) CO., LTD.), 14 August 2013 (14.08.2013), the whole document	1-30
A	US 2007101324 A1 (MICROSOFT CORPORATION), 03 May 2007 (03.05.2007), the whole document	1-30

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 May 2017 (08.05.2017)	**31 May 2017 (31.05.2017)**
Name and mailing address of the ISA/CN: State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No.: (86-10) 62019451	Authorized officer **WU, Yao** Telephone No.: (86-10) **62413705**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2017/077448

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 102222015 A	19 October 2011	US 8713568 B2	29 April 2014
		KR 20110114432 A	19 October 2011
CN 104636259 A	20 May 2015	None	
CN 103246552 A	14 August 2013	None	
US 2007101324 A1	03 May 2007	US 2011214017 A1	01 September 2011
		US 2007101338 A1	03 May 2007

国际检索报告

国际申请号
PCT/CN2017/077448

A. 主题的分类

G06F 9/52(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用）)

CNKI, CNPAT, WPI, EPODOC: 应用程序接口，死锁，函数，进程，线程，插入，节点，结点，时间，时长，阈值，阀值，API, application programming interface, deadlock, node?, threshold, time

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	CN 102222015 A (三星电子中国研发中心 等) 2011年 10月 19日 (2011-10-19) 说明书第[0025]-[0028]段，图1	1-30
A	CN 104636259 A (厦门雅迅网络股份有限公司) 2015年 5月 20日 (2015-05-20) 全文	1-30
A	CN 103246552 A (腾讯科技深圳有限公司) 2013年 8月 14日 (2013-08-14) 全文	1-30
A	US 2007101324 A1 (MICROSOFT CORPORATION) 2007年 5月 3日 (2007-05-03) 全文	1-30

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2017年 5月 8日	2017年 5月 31日

ISA/CN的名称和邮寄地址	受权官员
中华人民共和国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088	吴瑶
传真号 (86-10)62019451	电话号码 (86-10)62413705

表 PCT/ISA/210 (第2页) (2009年7月)

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2017/077448

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	102222015	A	2011年 10月 19日	US	8713568	B2	2014年 4月 29日
				KR	20110114432	A	2011年 10月 19日
CN	104636259	A	2015年 5月 20日		无		
CN	103246552	A	2013年 8月 14日		无		
US	2007101324	A1	2007年 5月 3日	US	2011214017	A1	2011年 9月 1日
				US	2007101338	A1	2007年 5月 3日