

(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)



(10) 国际公布号
WO 2017/167061 A1

(51) 国际专利分类号:

G06F 12/02 (2006.01)

(21) 国际申请号:

PCT/CN2017/077446

(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:

201610187840.2 2016 年 3 月 29 日 (29.03.2016) 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): 吴哲锋 (WU, Zhefeng) [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)。 杨江华 (YANG, Jianghua) [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,

[见续页]

(54) Title: PARALLEL MARKING PROCESSING METHOD AND DEVICE FOR GARBAGE COLLECTION

(54) 发明名称: 用于垃圾回收的并行标记处理方法及装置

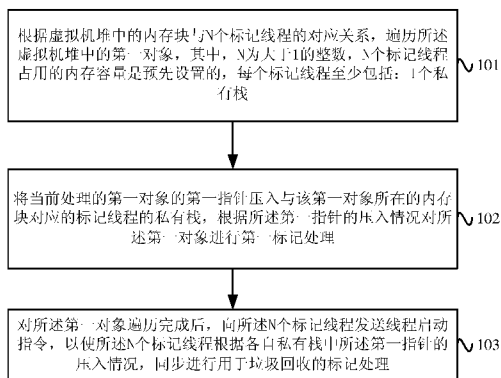


图 1

- 101 TRAVERSE, ACCORDING TO A CORRESPONDING RELATIONSHIP BETWEEN A MEMORY BLOCK IN A VIRTUAL MACHINE HEAP AND N MARKED THREADS, A FIRST OBJECT IN THE VIRTUAL MACHINE HEAP, WHEREIN N IS AN INTEGER GREATER THAN 1, A MEMORY VOLUME OCCUPIED BY THE N MARKED THREADS IS PRE-CONFIGURED, AND EACH OF THE MARKED THREADS AT LEAST COMPRISES: ONE PRIVATE STACK
- 102 PUSH A FIRST POINTER OF A FIRST OBJECT BEING CURRENTLY PROCESSED ONTO A PRIVATE STACK OF A MARKED THREAD CORRESPONDING TO A MEMORY BLOCK WHERE THE FIRST OBJECT IS LOCATED, AND PERFORM, ACCORDING TO A PUSH STATE OF THE FIRST POINTER, FIRST MARKING PROCESSING ON THE FIRST OBJECT
- 103 AFTER TRAVERSING OF THE FIRST OBJECT IS COMPLETED, SEND A THREAD INITIATION INSTRUCTION TO THE N MARKED THREADS, SUCH THAT THE N MARKED THREADS ARE ABLE TO SYNCHRONOUSLY PERFORM MARKING PROCESSING FOR GARBAGE COLLECTION ACCORDING TO A PUSH STATE OF THE FIRST POINTER IN THE RESPECTIVE PRIVATE STACKS OF THE N MARKED THREADS

(57) Abstract: A marking method and device for garbage collection, wherein the method comprises: traversing, according to a corresponding relationship between a memory block in a virtual machine heap and N marked threads, a first object in the virtual machine heap, wherein N is an integer greater than 1, a memory volume occupied by the N marked threads is pre-configured, and each of the marked threads at least comprises: one private stack (101); pushing a first pointer of a first object being currently processed onto a private stack of a marked thread corresponding to a memory block where the first object is located, and performing, according to a push state of the first pointer, first marking processing on the first object (102); after traversing of the first object is completed, sending a thread initiation instruction to the N marked threads, such that the N marked threads are able to synchronously perform marking processing for garbage collection according to the push state of the first pointer in the respective private stacks of the N marked threads (103). In the present invention, parallel marking processing of multiple threads can be realized using limited internal memory, improving the overall garbage collection performance.

(57) 摘要: 一种用于垃圾回收的标记方法及装置, 通过根据虚拟机堆中的内存块与 N 个标记线程的对应关系, 遍历所述虚拟机堆中的第一对象, 其中, N 为大于 1 的整数, N 个标记线程占用的内存容量是预先设置的, 每个标记线程至少包括: 1 个私有栈 (101); 将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈, 根据所述第一指针的压入情况对所述第一对象进行第一标记处理 (102); 对所述第一对象遍历完成后, 向所述 N 个标记线程发送线程启动指令, 以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况, 同步进行用于垃圾回收的标记处理 (103)。由此, 利用有限的内存实现多线程的并行标记处理, 提高整个垃圾回收的性能。



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))。

用于垃圾回收的并行标记处理方法及装置

技术领域

本申请涉及计算机技术领域，尤其涉及一种用于垃圾回收的并行标记
5 处理方法及装置。

背景技术

垃圾回收（即 Garbage Collection 简称 GC）技术被广泛使用在当今流
行的许多高级语言虚拟机中。GC 技术根据垃圾收集器（简称 Collector）
10 和宿主（简称 Mutator）的关系可以分为两种：Collector 工作时 Mutator 暂
停（简称 Stop-The-World-GC 即 STW-GC），Collector 工作时 Mutator 不暂
停（简称并发 GC 即 Concurrent-GC）。目前完全并发 GC 还没有哪一款虚
拟机真正实现，流行的高级语言虚拟机比如 JVM，V8 等都是 STW-GC 或
者部分 Concurrent-GC。非引用计数的 STW-GC 方案可以分为 3 种，即标
15 记-拷贝（简称 marking-copy），标记-清理（简称 marking-sweep），和标记-
压缩（简称 marking-compact）。

目前流行的高级语言虚拟机产品比如 JVM，V8 等的内存都是通过堆
（heap）来进行统一管理，堆以指定大小的内存块（简称为一个 page，一
般为操作系统的内存页大小的整数倍）为基本单位进行组织。比如 V8 虚
20 拟机的 page 大小为 1MByte。而且在每一个内存块（即 page）的起始部分
划分出一块位图区域（称为 bitmap）用来标记所在的 page 中每一个对象是
否是活跃对象。比如某个 page 中的某个对象对应在该 page 头部的 bitmap
位被置为 1，则说明该对象是活跃对象，进行 GC 时候不应该进行回收。

当前用于垃圾回收的标记方法为单线程的标记方式，根据标记对象的
25 不断增加，所占用的内存空间会不断增大，由此可见，目前的单线程标记
方式浪费了大量的内存空间，降低了处理性能和效率。

发明内容

本申请旨在至少在一定程度上解决相关技术中的技术问题之一。

为此，本申请的第一个目的在于提出一种用于垃圾回收的并行标记处理
5 理方法，该方法利用有限的内存实现多线程的并行标记处理，提高整个垃圾回收的性能。

本申请的第二个目的在于提出一种用于垃圾回收的并行标记处理方法。

本申请的第三个目的在于提出一种用于垃圾回收的并行标记处理装
10 置。

本申请的第四个目的在于提出一种用于垃圾回收的并行标记处理装置。

为达上述目的，本申请第一方面实施例提出了一种用于垃圾回收的并行标记处理方法，包括：根据虚拟机堆中的内存块与 N 个标记线程的对应
15 关系，遍历所述虚拟机堆中的第一对象，其中，N 为大于 1 的整数，N 个标记线程占用的内存容量是预先设置的，每个标记线程至少包括：1 个私有栈；将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理；对所述第一对象遍历完成后，向所述 N 个标记线程发
20 送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。

本申请实施例的用于垃圾回收的并行标记处理方法，通过根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 个标记线程占用的内存容量是预先设置的，将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私
25

有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理，
对所述第一对象遍历完成后，向所述N个标记线程发送线程启动指令，以
使所述N个标记线程根据各自私有栈中所述第一指针的压入情况，同步进
行用于垃圾回收的标记处理。由此，利用有限的内存实现多线程的并行标
5 记处理，提高整个垃圾回收的性能。

为达上述目的，本申请第二方面实施例提出了一种用于垃圾回收的并
行标记处理方法，包括：预设的N个标记线程中的每个标记线程至少包括：
1 个私有栈，所述方法应用在每个标记线程中，其中，应用在所述第一标记线
程中的所述方法包括以下步骤：第一标记线程根据线程启动指令，从第一
10 私有栈中取出预先压入的第一对象的第一指针；所述第一标记线程遍历所
述第一指针指示的所述第一对象引用的第二对象，并根据预设的N个标记
线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。

本申请实施例的用于垃圾回收的并行标记处理方法，通过第一标记线
程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指
15 针，遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设
的N个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二
标记处理。由此，参照第一标记线程的标记处理过程，通过N个标记线程
的并行标记处理，实现了利用有限的内存进行并行标记处理，提高整个垃
圾回收的性能。

为达上述目的，本申请第三方面实施例提出了一种用于垃圾回收的并
行标记处理装置，包括：遍历模块，用于根据虚拟机堆中的内存块与N个
标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N为大于
1 的整数，N个标记线程占用的内存容量是预先设置的，每个标记线程至
少包括：1 个私有栈；第一标记模块，用于将当前处理的第一对象的第一
25 指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述

第一指针的压入情况对所述第一对象进行第一标记处理；启动模块，用于对所述第一对象遍历完成后，向所述N个标记线程发送线程启动指令，以使所述N个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。

5 本申请实施例的用于垃圾回收的并行标记处理装置，通过根据虚拟机堆中的内存块与N个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N个标记线程占用的内存容量是预先设置的，将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理，
10 对所述第一对象遍历完成后，向所述N个标记线程发送线程启动指令，以使所述N个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。由此，利用有限的内存实现多线程的并行标记处理，提高整个垃圾回收的性能。

 为达上述目的，本申请第四方面实施例提出了一种用于垃圾回收的并行
15 标记处理装置，包括：预设的N个标记线程中的每个标记线程至少包括：
1 个私有栈，所述装置应用在每个标记线程中，其中，应用在第一标记线程中的所述装置包括：获取模块，用于根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针；第二标记模块，用于遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的N个标记线程
20 与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。

 本申请实施例的用于垃圾回收的并行标记处理装置，通过第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针，遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的N个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二
25 标记处理。由此，参照第一标记线程的标记处理过程，通过N个标记线程

的并行标记处理，实现了利用有限的内存进行并行标记处理，提高整个垃圾回收的性能。

附图说明

5 本发明上述的和/或附加的方面和优点从下面结合附图对实施例的描述中将变得明显和容易理解，其中：

图 1 是本申请一个实施例的用于垃圾回收的并行标记处理方法的流程图；

10 图 2 是本申请另一个实施例的用于垃圾回收的并行标记处理方法的流程图；

图 3 为用于垃圾回收的第一标记处理流程示意图；

图 4 为预先申请的 N 个标记线程的示意图；

图 5 是应用图 4 所示的标记线程并行标记处理的流程图一；

图 6 是应用图 4 所示的标记线程并行标记处理的流程图二；

15 图 7 是应用图 4 所示的标记线程并行标记处理的流程图三；

图 8 是应用图 4 所示的标记线程并行标记处理的流程图四；

图 9 是应用图 4 所示的标记线程并行标记处理的流程图五；

图 10 是应用图 4 所示的标记线程并行标记处理的流程图六；

20 图 11 是本申请一个实施例的用于垃圾回收的并行标记处理装置的结构示意图；

图 12 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图；

图 13 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图；

25 图 14 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的

结构示意图；

图 15 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图；

图 16 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图；

图 17 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

具体实施方式

下面详细描述本申请的实施例，所述实施例的示例在附图中示出，其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施例是示例性的，旨在用于解释本申请，而不能理解为对本申请的限制。

下面参考附图描述本申请实施例的用于垃圾回收的并行标记处理方法及装置。

图 1 是本申请一个实施例的用于垃圾回收的并行标记处理方法的流程图。

如图 1 所示，该用于垃圾回收的并行标记处理方法包括：

步骤 101，根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 为大于 1 的整数，N 个标记线程占用的内存容量是预先设置的，每个标记线程至少包括：1 个私有栈。

具体地，为了提高用于垃圾回收的标记处理效率，本发明各实施例提供的处理方法利用了多核 CPU 的并行处理能力，预先申请 N 个标记线程，其中，每个标记线程至少包括：1 个私有栈，私有栈用于存储本线程负责标记的对象的指针。

需要说明的是，N 为大于 1 的整数，根据实际应用的 CPU 的并行处理能力而定。例如：

如果 CPU 的并行处理能力为双核，可以申请 2 个容量限定的标记线程；
或，

5 如果 CPU 的并行处理能力为四核，可以申请 4 个容量限定的标记线程；
或，

如果 CPU 的并行处理能力为八核，可以申请 8 个容量限定的标记线程。

需要注意的是，以上仅为举例说明，可以根据实际应用需要进行设置和调整。

10 需要强调的是，随着标记对象的增加，现有的标记处理技术需要不断的增加标记线程的内存占用量，降低了标记处理的性能和效率。因此，本发明提供的并行标记处理方法，预先对 N 个标记线程占用的内存容量进行设置，也就是会设置，N 个标记线程占用的内存容量不会随着标记对象的溢出而增加，以便保证标记处理的性能和效率。

15 需要说明的是，可以根据应用需要采用多种方式对 N 个标记线程占用的内存容量进行设置，例如：

示例一，对每个标记线程的容量进行设置；

示例二，对 N 个标记线程占用的内存总容量进行设置，每个标记线程可以不限定，或者，比如限定最大的一个等。

20 为了提高标记效率，实现多线程并行标记的负载均衡，预先建立虚拟机堆中的内存块与 N 标记线程的对应关系。也就是说，预先为每个标记线程配置好对应负责标记的虚拟机堆中的内存块。

需要注意的是，可以根据实际应用需要建立虚拟机堆中内存块与 N 个标记线程的对应关系，例如包括：

25 示例一：为每个内存块指定对应的标记线程；或者，

示例二：为每个标记线程指定负责标记的内存块。

进而，从虚拟机系统中获取预先存储的虚拟机堆中所有的第一对象，根据上述设置的虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历虚拟机堆中的第一对象。

5 步骤 102，将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理。

步骤 103，对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压
10 入情况，同步进行用于垃圾回收的标记处理。

具体地，首先确定当前处理的第一对象所在的内存块，然后根据上述的对应关系获取与该第一对象所在的内存块对应的标记线程。将该第一对象的第一指针压入到与该第一对象所在的内存块对应的标记线程的私有栈中。

15 进而，根据该第一对象的第一指针的压入情况，对当前处理的第一对象进行第一标记处理。也就是说，如果将当前第一对象的第一指针成功压入到对应标记线程的私有栈中，则确定该私有栈没有溢出，将当前处理的第一对象标记为压入状态；如果没有将当前第一对象的第一指针成功压入到对应标记线程的私有栈中，则确定该私有栈溢出，将当前处理的第一对
20 象标记为溢出状态。

需要说明的是，对第一对象是否溢出的标记方式有很多，可以根据需要进行选择，例如：

示例一，可以通过列表的方式记录与每个标记线程的私有栈对应的第一对象的压入情况；或者，

25 示例二，可以通过第一对象所在内存块位图中的相应位置标记第一对

象的压入情况。

需要说明的是，以上仅为举例说明，可以根据实际应用需要进行选择标记方式。

对虚拟机堆中的第一对象遍历完成后，各个标记线程的私有栈中都存
5 放了本线程负责标记的内存块中第一对象的第一指针。如果标记线程的私有栈中溢出，说明私有栈的空间已满，无法继续存放本线程负责标记的内存块中第一对象的第一指针，则将这些第一对象标记为溢出状态。当标记线程的私有栈中有新的处理空间时，继续存放本线程负责标记的内存块中第一对象的第一指针。

10 进而，向 N 个标记线程发送线程启动指令，从而 N 个标记线程根据各自私有栈中第一指针的压入情况，同步进行用于垃圾回收的标记处理。

本申请实施例的用于垃圾回收的并行标记处理方法，通过根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 个标记线程占用的内存容量是预先设置的，将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理，
15 对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。由此，利用有限的内存实现多线程的并行标记处理，提高整个垃圾回收的性能。
20

为了更加清楚的说明上述实施例中虚拟机堆中所有内存块与 N 个标记线程的对应关系的建立过程，以及第一标记过程，通过图 2 所示实施例进行说明。

图 2 是本申请另一个实施例的用于垃圾回收的并行标记处理方法的流程图。
25

如图 2 所示,该用于垃圾回收的并行标记处理方法可以包括以下步骤:

步骤 201, 申请 N 个标记线程, 其中, N 为大于 1 的整数, N 个标记线程占用的内存容量是预先设置的, 每个标记线程至少包括: 1 个私有栈。

具体地, 根据实际应用需要和 CPU 的多核处理能力, 设置标记线程的数量 N, 以及设置 N 个标记线程所占的内存容量。其中, 每个标记线程包
5 括一个私有栈, 私有栈用于存储本线程负责标记的对象的指针。

步骤 202, 遍历虚拟机堆的所有内存块, 为每个内存块分配编号;

步骤 203, 根据内存块编号和标记线程总数 N, 确定与每个内存块编号对应的标记线程编号。

10 具体地, 为了向虚拟机堆的每个内存块分配负责标记内存块对象的标记线程, 遍历虚拟机堆的内存块, 为每个内存块分配编号。

进而, 根据内存块编号和标记线程总数 N, 确定与每个内存块编号对应的标记线程编号。

需要说明的是, 根据内存块编号和标记线程总数 N, 确定与每个内存块编号对应的标记线程编号的方式有很多, 例如: 可以通过获取标记线程
15 总数 N 被内存块编号整除后的余数, 确定与每个内存块编号对应的标记线程编号, 公式表达如下所示:

$ID = id \% N$, 其中,

N 为标记线程总数; id 为内存块编号; ID 为标记线程编号; “%”表示
20 N 被 id 整除后的取余处理。

例如: 假设当前申请的标记线程总数 $N=4$, 编号分别为 0-3, 内存块为十个, 编号分别为 1-10, 通过上述公式处理后获知:

内存块编号 1 对应的标记线程编号为 0;

内存块编号 2 对应的标记线程编号为 1;

25 内存块编号 3 对应的标记线程编号为 2;

内存块编号 4 对应的标记线程编号为 3;

内存块编号 5 对应的标记线程编号为 0;

内存块编号 6 对应的标记线程编号为 1;

内存块编号 7 对应的标记线程编号为 2;

5 内存块编号 8 对应的标记线程编号为 3;

内存块编号 9 对应的标记线程编号为 1;

内存块编号 10 对应的标记线程编号为 2。

步骤 204, 根据虚拟机堆中的内存块与 N 个标记线程的对应关系, 遍历所述虚拟机堆中的第一对象, 将当前处理的第一对象的第一指针压入与
10 该第一对象所在的内存块对应的标记线程的私有栈;

步骤 205, 判断是否将当前第一对象的第一指针成功压入对应标记线程的私有栈中;

步骤 206, 如果将所述第一指针成功压入对应标记线程的私有栈, 则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为压
15 入状态;

步骤 207, 如果未能将所述第一指针成功压入对应标记线程的私有栈, 则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为溢出状态。

步骤 208, 对所述第一对象遍历完成后, 向所述 N 个标记线程发送线程启动指令, 以使所述 N 个标记线程根据各自私有栈中所述第一指针的压
20 入情况, 同步进行用于垃圾回收的标记处理。

具体地, 确定当前处理的第一对象所在的内存块, 然后根据上述的对应关系获取与该第一对象所在的内存块对应的标记线程。将该第一对象的第一指针压入到与该第一对象所在的内存块对应的标记线程的私有栈中。

25 进而, 根据该第一对象的第一指针的压入情况, 对当前处理的第一对

象进行第一标记处理。也就是说，如果将当前第一对象的第一指针成功压入到对应标记线程的私有栈中，则确定该私有栈没有溢出，将当前处理的第一对象标记为压入状态；如果没有将当前第一对象的第一指针成功压入到对应标记线程的私有栈中，则确定该私有栈溢出，将当前处理的第一对象标记为溢出状态。

对所述第一对象遍历完成后，各个标记线程的私有栈中都存放了本线程负责标记的内存块中第一对象的第一指针。如果标记线程的私有栈中溢出，说明私有栈的空间已满，无法继续存放本线程负责标记的内存块中第一对象的第一指针，则将这些第一对象标记为溢出状态。当标记线程的私有栈中有新的处理空间时，继续存放本线程负责标记的内存块中第一对象的第一指针。

进而，向 N 个标记线程发送线程启动指令，从而 N 个标记线程根据各自私有栈中第一指针的压入情况，同步进行用于垃圾回收的标记处理。

为了更加清楚的说明上述实施例中涉及的对第一对象进行第一标记处理的处理过程，通过图 3 所示实施例说明如下。

图 3 为用于垃圾回收的第一标记处理流程示意图。

参见图 3，本实施例中采用在内存块位图中进行着色标记的方式对第一对象的压入情况进行第一标记处理，具体标记如下：

如果将当前处理的第一对象的第一指针成功压入对应标记线程的私有栈中，则将与该第一对象所在内存块位图中的对应位置进行标黑处理；

如果没有将当前处理的第一对象的第一指针成功压入对应标记线程的私有栈中，则将与该第一对象所在内存块位图中的对应位置进行标灰处理。

需要说明的是，以上根据第一对象的第一指针向对应标记线程的私有栈的不同压入情况，在内存块位图中与第一对象的对应位置进行的着色处

理仅仅是示例性说明，可以根据具体的应用需要进行调整。

基于上述标记方式，具体以处理虚拟堆中的根对象 A（本示例中虚拟堆中的根对象相当于上述实施例中涉及的第一对象）为例说明如下：

步骤 10：申请 N 个标记线程，其中，N 为大于 1 的整数，N 个标记线程占用的内存容量是预先设置的，每个标记线程至少包括：1 个私有栈。

步骤 20：建立虚拟机堆中的内存块与 N 个标记线程的对应关系。

步骤 30：遍历虚拟机堆中的根对象，如果遍历完毕，则退出，否则执行步骤 40。

步骤 40：将当前处理的根对象 A 的第一指针压入与 A 所在的内存块 10 对应的标记线程 1 的私有栈 1 中。

步骤 50：判断是否将 A 的第一指针成功压入标记线程 1 的私有栈 1 中；

步骤 60：如果将 A 的第一指针成功压入标记线程 1 的私有栈 1 中，则将内存块 10 的位图中、与 A 对应的位置标黑，即 A 为压入状态。

步骤 70：如果没有将 A 的第一指针成功压入标记线程 1 的私有栈 1 中，则将内存块 10 的位图中、与 A 对应的位置标灰，即 A 为溢出状态。

本申请实施例的用于垃圾回收的并行标记处理方法，通过申请 N 个标记线程，遍历虚拟机堆的所有内存块，为每个内存块分配编号；根据内存块编号和标记线程总数 N，确定与每个内存块编号对应的标记线程编号，进而将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理，对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。由此，利用有限的内存实现高效的并行标记，同时实现标记线程负载均衡，提高整个垃圾回收的

性能。

基于上述实施例，对所述第一对象遍历完成后，各个标记线程的私有栈中都存放了本线程负责标记的内存块中第一对象的第一指针。如果标记线程的私有栈中溢出，说明私有栈的空间已满，无法继续存放本线程负责
5 标记的内存块中第一对象的第一指针，则将这些第一对象标记为溢出状态。当标记线程的私有栈中有新的处理空间时，继续存放本线程负责标记的内存块中第一对象的第一指针。

进而，向 N 个标记线程发送线程启动指令，从而 N 个标记线程根据各自私有栈中第一指针的压入情况，同步进行用于垃圾回收的标记处理。

10 需要说明的是，预先申请的 N 个标记线程的处理过程是同步并行的，为了更加清楚的说明 N 个标记线程的标记处理过程，以第一标记线程的标记处理过程为例，通过下述实施例说明如下，其他 (N-1) 个标记线程的标记处理过程参见第一标记线程，不再一一赘述。

图 4 为预先申请的 N 个标记线程的示意图。

15 参见图 4，预先申请的 N 个标记线程，以及 N 个标记线程所占用的内存容量。N 个标记线程包括：Thread-1、Thread-2，...Thread-N。其中，每个标记线程至少包括：1 个私有栈 (stack)，经过上述处理，各个私有栈中存储有本线程负责标记的第一对象 (Root Objects) 的第一指针。

图 5 是应用图 4 所示的标记线程并行标记处理的流程图一。

20 参见图 5，本实施例中以第一标记线程 (Thread-1) 的标记处理过程为例，该用于垃圾回收的并行标记处理方法包括：

步骤 301，第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针。

步骤 302，所述第一标记线程遍历所述第一指针指示的所述第一对象
25 引用的第二对象，并根据预设的 N 个标记线程与虚拟机堆中内存块的对应

关系对所述第二对象进行第二标记处理。

具体地，第一标记线程接收到线程启动指令之后，根据该线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针。

进而，第一标记线程查询预存的对象关系表，获取当前处理的第一指针指示的第二对象，需要解释的是，第二对象为第一对象的引用对象。

第一标记线程遍历当前处理的第一指针指示的第二对象，根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系，判断负责处理当前第二对象所在的内存块的标记线程是否为本线程，从而对第二对象进行第二标记处理。

本申请实施例，通过第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针，遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的并行标记处理，实现了利用有限的内存进行并行标记处理，提高整个垃圾回收的性能。

图 6 是应用图 4 所示的标记线程并行标记处理的流程图二。

继续参见图 4，基于图 5 所示实施例，每个标记线程还包括：(N-1) 个公用输出队列 (queue-1、queue-2...queue-N-1)，以及 1 个缓存队列 (buf-queue)。其中，(N-1) 个公用输出队列用于存储其他 (N-1) 个标记线程负责标记的对象的指针；当与其他 (N-1) 个标记线程对应的公用输出队列溢出时，1 个缓存队列用于缓存其他 (N-1) 个标记线程负责标记的对象的指针。

参见图 6，继续以第一标记线程 (Thread-1) 的标记处理过程为例，图 5 所示实施例中的步骤 302 具体包括：

步骤 401，第一标记线程根据所述对应关系，判断是否负责对当前处

理的第二对象所在的内存块 P1 进行标记处理;

具体地, 第一标记线程根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系, 判断本线程是否负责对当前处理的第二对象所在的内存块 P1 进行标记处理。

- 5 如果第一标记线程获知负责对该内存块 P1 进行标记处理的标记线程为第一标记线程, 即本线程负责对该内存块 P1 进行标记处理, 进而执行步骤 402-步骤 406;

- 如果第一标记线程获知负责对该内存块 P1 进行标记处理的标记线程为标记线程 M, 标记线程 M 不是第一标记线程, 即本线程不负责对该内存块 P1 进行标记处理, 进而执行步骤 407-步骤 415。

步骤 402, 将当前处理的第二对象的第二指针压入到所述第一私有栈中。

- 具体地, 针对步骤 401 中第一标记线程负责对所述内存块 P1 进行标记处理的判断分支, 第一标记线程将负责处理对象的指针压入到本线程的私有栈中进行处理。即将当前处理的第二对象的第二指针压入到所述第一私有栈中进行第二标记处理。

步骤 403, 所述第一标记线程判断是否成功将所述第二指针压入到所述第一私有栈中。

- 具体地, 第一标记线程判断是否成功将当前处理的第二对象的第二指针压入到本线程的第一私有栈中。

如果第一标记线程未能成功将当前处理的第二对象的第二指针压入到本线程的第一私有栈中, 则执行步骤 404 和步骤 405 对第二对象进行溢出状态的标记处理;

- 如果第一标记线程成功将当前处理的第二对象的第二指针压入到本线程的第一私有栈中, 则执行步骤 406 继续对下一个第二对象进行第二标

记处理。

步骤 404, 将所述内存块 P1 的位图中、与该第二对象对应的状态标记为溢出状态。

步骤 405, 设置与所述内存块 P1 对应的溢出标志。

5 具体地, 针对步骤 403 中第一标记线程未能成功将当前处理的第二对象的第二指针压入到本线程的第一私有栈中的判断分支, 将第二对象设置为溢出状态。例如: 将上述内存块 P1 的位图中、与该第二对象对应的状态标记为溢出状态。并且进一步地设置与该内存块 P1 对应的溢出标志。

10 步骤 406, 如果所述第一标记线程成功将所述第二指针压入到所述第一私有栈中, 则继续对所述第一指针对应的下一个第二对象进行处理。

具体地, 针对步骤 403 中第一标记线程成功将所述第二指针压入到所述第一私有栈中的判断分支, 继续对从第一私有栈中取出的第一指针对应的下一个第二对象进行处理。

步骤 407, 确定标记线程 M 负责对所述内存块 P1 进行标记处理;

15 步骤 408, 将所述第二指针压入到所述第一标记线程中与所述标记线程 M 对应的公用输出队列 M 中;

具体地, 针对步骤 401 中第一标记线程不负责与所述内存块 P1 进行标记处理的判断分支, 根据 N 个标记线程与虚拟机堆中内存块的对应关系, 确定标记线程 M 负责对所述内存块 P1 进行标记处理。

20 进而, 第一标记线程将当前处理的第二对象的第二指针压入到第一标记线程中与标记线程 M 对应的公用输出队列 M 中, 以使标记线程 M 后续从第一标记线程中的公用输出队列 M 中获取待处理的第二对象的第二指针。

25 步骤 409, 所述第一标记线程判断是否成功将所述第二指针压入到所述公用输出队列 M 中;

第一标记线程判断是否成功将当前处理的第二对象的第二指针压入到公用输出队列 M 中。

如果第一标记线程成功将当前处理的第二对象的第二指针压入到第一标记线程中的公用输出队列 M 中，则执行步骤 410，继续对当前从第一私有栈中中获取的第一指针对应的下一个第二对象进行处理。

如果第一标记线程未能成功将当前处理的第二对象的第二指针压入到第一标记线程中的公用输出队列 M 中，则执行步骤 411，通过第一标记线程的第一缓存队列缓存该第二指针。

步骤 410，继续对所述第一指针对应的下一个第二对象进行处理。

具体地，针对步骤 409 中第一标记线程成功将当前第二指针压入到第一标记线程中的公用输出队列 M 中的判断分支，继续对上述第一指针对应的下一个第二对象进行处理。

步骤 411，将所述第二指针压入到所述第一标记线程中的第一缓存队列中；

具体地，针对步骤 409 中第一标记线程未能成功将当前第二指针压入到第一标记线程中的公用输出队列 M 中的判断分支，将该第二指针压入到第一标记线程中的第一缓存队列中。

步骤 412，所述第一标记线程判断是否成功将所述第二指针压入到所述第一缓存队列中；

具体地，第一标记线程判断是否成功将该第二指针压入到第一标记线程的第一缓存队列中；

如果第一标记线程未能成功将所述第二指针压入到所述第一缓存队列中，则执行步骤 413 和步骤 414，对当前处理的第二对象标记溢出状态；

如果第一标记线程成功将所述第二指针压入到所述第一缓存队列中，则执行步骤 415，对下一个第二对象继续进行标记处理。

步骤 413, 将所述内存块 P1 的位图中、与当前处理的第二对象对应的状态标记为溢出状态。

步骤 414, 设置与所述内存块 P1 对应的溢出标志。

具体地, 针对步骤 412 中第一标记线程未能成功将所述第二指针压入到所述第一缓存队列中的判断分支, 将商户内存块 P1 的位图中、与当前处理的第二对象对应的状态标记为溢出状态, 并进一步地设置与该内存块 P1 对应的溢出标志。

步骤 415, 继续对所述第一指针对应的下一个第二对象进行处理。

具体地, 针对步骤 412 中第一标记线程成功将所述第二指针压入到所述第一缓存队列中的判断分支, 继续对当前第一指针对应的下一个第二对象进行处理。

基于图 6 所示实施例, 进一步地, 在步骤 401 之前, 还包括:

查询当前处理的第二对象所在内存块 P1 的位图中与该第二对象对应的状态标记。如果与该第二对象对应的状态标记为未标记状态, 则将与该第二对象对应的状态修改为压入状态。

需要说明的是, 对第二对象是否溢出的标记方式有很多, 可以根据需要进行选择, 例如:

示例一, 可以通过列表的方式记录与每个第二对象的压入情况; 或者,

示例二, 可以通过第二对象所在内存块位图中的相应位置标记第二对象的压入情况。

需要说明的是, 以上仅为举例说明, 可以根据实际应用需要进行选择标记方式。

本申请实施例, 通过第一标记线程根据线程启动指令, 从第一私有栈中取出预先压入的第一对象的第一指针, 遍历所述第一指针指示的所述第一对象引用的第二对象, 并根据预设的 N 个标记线程与虚拟机堆中内存块

的对应关系对所述第二对象进行第二标记处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的并行标记处理，利用有限的内存实现高效的并行标记，同时实现标记线程负载均衡，从而减少 CPU 缓存一致性冲突，提高整个垃圾回收的性能。

5 图 7 是应用图 4 所示的标记线程并行标记处理的流程图三。

继续参见图 4，每个标记线程包括：1 个私有栈 (stack)、(N-1) 个公用输出队列(queue-1、queue-2...queue-N-1)，以及 1 个缓存队列(buf-queue)。其中，1 个私有栈 (stack)用于存储本线程负责标记的对象的指针；(N-1) 个公用输出队列用于存储其他 (N-1) 个标记线程负责标记的对象的指针；
10 当与其他 (N-1) 个标记线程对应的公用输出队列溢出时，1 个缓存队列用于缓存其他 (N-1) 个标记线程负责标记的对象的指针。

参见图 7，基于图 5 所示实施例，所述方法还包括：

步骤 501，第一标记线程接到线程启动指令后，判断本线程的第一私有栈是否为空；

15 具体地，第一标记线程判断本线程的第一私有栈是否为空；

如果第一标记线程的第一私有栈不为空，则按照图 5 中所示实施例中的步骤 302 从第一私有栈中取出预先压入的第一对象的第一指针，并按照图 5 或图 6 所示的实施过程进行处理，此处不再赘述。

如果第一标记线程的第一私有栈为空，则执行步骤 502 继续判断第一
20 标记线程中的第一缓存队列是否为空。

步骤 502，所述第一标记线程判断本线程的第一缓存队列是否为空；

具体地，第一标记线程判断本线程的第一缓存队列是否为空；

如果第一标记线程中的第一缓存队列不为空，执行步骤 503，从而对
25 第一标记线程不负责处理的第三对象向本线程中与该第三对象对应的公用输出队列进行分配处理；

如果第一标记线程中的第一缓存队列为空，则通过图 8 所示实施例描述如何进一步地对其他 (N-1) 个标记线程中与第一标记线程对应的公用输出队列中的第四对象进行标记处理，后续实施例详细说明。

步骤 503，所述第一标记线程从所述第一缓存队列中取出预先压入的
5 第三对象的第三指针，其中，所述第一标记线程不负责当前处理的第三对象所在的内存块 P2 进行标记处理；

步骤 504，所述第一标记线程根据所述对应关系，确定标记线程 W 负责当前处理的第三对象所在的内存块 P2 进行标记处理；

步骤 505，所述第一标记线程将所述第三指针压入到所述第一标记线
10 程中与所述标记线程 W 对应的公用输出队列 W 中；

具体地，第一标记线程从本线程的第一缓存队列中取出预先压入的第三对象的第三指针，由于第一缓存队列中存放的对象是需要放入第一标记线程中的公用输出队列中，以供对应的其他标记线程进行处理的。当第一标记线程中的公用输出队列中出现溢出时，将溢出的第三对象放入第一标
15 记线程中的第一缓存队列中缓存。

由此可见，第一标记线程不负责当前处理的第三对象所在的内存块 P2 进行标记处理。第一标记线程根据虚拟机堆的内存块与标记线程的对应关系，确定标记线程 W 负责当前处理的第三对象所在的内存块 P2 进行标记处理。

20 进而，第一标记线程将该第三指针压入到第一标记线程中与标记线程 W 对应的公用输出队列 W 中，以供标记线程 W 从第一标记线程的公用输出队列 W 中获取该第三指针进行标记处理。

步骤 506，所述第一标记线程判断是否成功将当前处理的第三指针压入到所述公用输出队列 W 中；

25 步骤 507，继续对所述第一缓存队列中的下一个第三对象向对应的公

用输出队列进行压入处理。

步骤 508, 将所述第三指针重新压入所述第一缓存队列中。

具体地, 第一标记线程判断是否成功将当前处理的第三指针压入到本线程的公用输出队列 W 中。

5 如果第一标记线程成功将该第三指针压入到本线程的公用输出队列 W 中, 则执行步骤 507, 继续对所第一缓存队列中的下一个第三对象向对应的公用输出队列进行压入处理。

10 如果第一标记线程未能成功将该第三指针压入到本线程的公用输出队列 W 中, 则将该第三指针重新压入本线程的第一缓存队列中。此时, 不存在溢出的问题, 因为该第三指针就是从本线程的第一缓存队列中取出来的。

15 基于上述实施例, 本申请在第一标记线程的第一私有栈为空时, 且第一缓存队列不为空, 从第一缓存队列中取出预先压入的第三对象的第三指针, 向本线程中与对应标记线程的公用输出队列中进行压入处理, 以供与第三对象对应的标记线程对其进行标记处理。由此, 参照第一标记线程的标记处理过程, 通过 N 个标记线程的进行协同并行的标记处理, 利用有限的内存实现高效的并行标记, 同时实现标记线程负载均衡, 从而减少 CPU 缓存一致性冲突, 提高整个垃圾回收的性能。

图 8 是应用图 4 所示的标记线程并行标记处理的流程图四。

20 参见图 8, 基于图 7 所示实施例, 本实施例是基于图 7 中的步骤 502, 针对如果第一标记线程中的第一缓存队列为空的判断分支, 通过图 8 所示实施例描述如何进一步地对其他 (N-1) 个标记线程中与第一标记线程对应的公用输出队列中的第四对象进行标记处理, 具体包括:

25 步骤 601, 第一标记线程判断其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列是否为空;

具体地，第一标记线程判断其他（N-1）个标记线程中与所述第一标记线程对应的公用输出队列是否为空；

如果其他（N-1）个标记线程中与第一标记线程对应的公用输出队列不为空，则执行步骤 602，从而第一标记线程从其他标记线程中获取本线程负责处理的第四对象，向本线程的第一私有栈进行压入处理。

如果其他（N-1）个标记线程中与第一标记线程对应的公用输出队列为空，则通过图 9 所示实施例描述如何进一步地对第一标记线程负责标记的、且具有溢出状态的第五对象进行标记处理，后续实施例详细说明。

步骤 602，第一标记线程从当前处理的公用输出队列中取出预先压入的第四对象的第四指针，其中，所述第四对象为所述标记线程 T 预先压入的、由所述第一标记线程负责对所述第四对象所在的内存块 P3 进行标记处理；

步骤 603，所述第一标记线程将所述第四指针压入到所述第一私有栈中；

具体地，针对步骤 601 中如果其他（N-1）个标记线程中与第一标记线程对应的公用输出队列不为空的判断分支，说明其他的标记线程中还具有需要由第一标记线程标记处理的第四对象。

进而，第一标记线程从当前处理的公用输出队列中取出预先压入的第四对象的第四指针，其中，该第四对象为标记线程 T 预先压入的、并且由第一标记线程负责对该第四对象所在的内存块 P3 进行标记处理。从而第一标记线程将该第四指针压入到所述第一私有栈中。

步骤 604，所述第一标记线程判断是否成功将所述第四指针压入到所述第一私有栈中；

步骤 605，将所述内存块 P3 的位图中、与该第四对象对应的状态标记为溢出状态。

步骤 606, 设置与所述内存块 P3 对应的溢出标志。

步骤 607, 继续对下一个第四对象向所述第一私有栈中进行压入处理。

具体地, 第一标记线程判断是否成功将该第四指针压入到本线程的第一私有栈中。

5 如果第一标记线程未能成功将该第四指针压入到本线程的第一私有栈中, 则执行步骤 605 和步骤 606, 将该第四对象所在的内存块 P3 的位图中、与该第四对象对应的状态标记为溢出状态, 并且设置与所述内存块 P3 对应的溢出标志。

 如果第一标记线程成功将该第四指针压入到本线程的第一私有栈中,
10 则则执行步骤 607, 继续对下一个第四对象向本线程的第一私有栈中进行压入处理。

 基于上述实施例, 本申请的第一标记线程在第一私有栈和第一缓存队列为空, 且其他标记线程中与第一标记线程对应的公用输出队列不为空时, 取出预先压入的第四对象的第四指针, 向本线程中的第一私有栈压入
15 处理。由此, 参照第一标记线程的标记处理过程, 通过 N 个标记线程的进行协同并行的标记处理, 利用有限的内存实现高效的并行标记, 同时实现标记线程负载均衡, 从而减少 CPU 缓存一致性冲突, 提高整个垃圾回收的性能。

图 9 是应用图 4 所示的标记线程并行标记处理的流程图五。

20 参见图 9, 基于图 8 所示实施例, 本实施例是基于图 9 中的步骤 601, 针对如果其他 (N-1) 个标记线程中与第一标记线程对应的公用输出队列为空的判断分支, 则通过图 9 所示实施例描述如何进一步地对第一标记线程负责标记的、且具有溢出状态的第五对象进行标记处理, 具体包括:

 步骤 701, 第一标记线程判断本线程负责标记的内存块中是否有溢出
25 状态的第五对象;

步骤 702, 如果所述第一标记线程负责标记的内存块中不具有溢出状态的第五对象, 则标记结束, 退出线程。

步骤 703, 如果所述第一标记线程负责标记的内存块中具有溢出状态的第五对象, 所述第一标记线程将当前处理的内存块 P4 中标记溢出状态的第五对象修改为压入状态, 将当前第五对象的第五指针压入到所述第一私有栈中;

具体地, 第一标记线程判断本线程负责标记的内存块中是否有溢出状态的第五对象。

如果第一标记线程负责标记的内存块中不具有溢出状态的第五对象, 则说明本线程负责标记的对象遍历完成, 标记结束, 执行步骤 702 退出线程。

如果第一标记线程负责标记的内存块中具有溢出状态的第五对象, 则说明本线程负责标记的对象还没有遍历完成。进而, 第一标记线程执行步骤 703, 将当前处理的内存块 P4 中标记溢出状态的第五对象修改为压入状态, 将当前第五对象的第五指针压入到所述第一私有栈中。

步骤 704, 所述第一标记线程判断是否成功将所述第五指针压入到所述第一私有栈中;

步骤 705, 如果所述第一标记线程未能成功将所述第五指针压入到所述第一私有栈中, 则将所述内存块 P4 的位图中、与该第五对象对应的状态标记为溢出状态。

步骤 706, 设置与所述内存块 P5 对应的溢出标志。

步骤 707, 如果所述第一标记线程成功将所述第五指针压入到所述第一私有栈中, 则继续对下一个第五对象进行处理。

具体地, 第一标记线程判断是否成功将当前的第五指针压入到本线程的第一私有栈中。

如果第一标记线程未能成功将该第五指针压入到本线程的第一私有栈中，则说明第一私有栈空间不足有溢出，则执行步骤 705 和步骤 706，将该第五对象所在的内存块 P4 的位图中、与该第五对象对应的状态标记为溢出状态，并且进一步地设置与所述内存块 P5 对应的溢出标志。

- 5 如果第一标记线程成功将该第五指针压入到本线程的第一私有栈中，则执行步骤 707，继续对下一个第五对象进行处理。

基于上述实施例，本申请的第一标记线程在第一私有栈、第一缓存队列、以及其他标记线程中与第一标记线程对应的公用输出队列都为空时，根据本线程负责标记的具有溢出状态的第五对象进行压入处理。由此，参
10 照第一标记线程的标记处理过程，通过 N 个标记线程的并行的标记处理，利用有限的内存实现高效的并行标记，提高整个垃圾回收的性能。

为了更加清楚的说明图 4-图 9 所示实施例中涉及的对第二对象至第五对象进行处理的处理过程，通过图 10 所示实施例说明如下。

图 10 是应用图 4 所示的标记线程并行标记处理的流程图六。

- 15 参见图 10，本实施例中采用在内存块位图中进行着色标记的方式对第二对象至第五对象的压入情况进行标记处理，具体标记如下：

如果将当前处理的对象的指针成功压入对应的目标空间中，则将与当前对象所在内存块位图中的对应位置进行标黑处理；

- 20 如果没有将当前处理的对象的指针成功压入对应的目标空间中，则将与当前对象所在内存块位图中的对应位置进行标灰处理；

如果当前处理对象所在内存块位图中的对应位置为标白状态，说明该对象为未处理状态。

- 需要说明的是，以上根据处理对象的指针向对应目标空间的不同压入情况，在内存块位图中与当前处理对象的对应位置进行的着色处理仅仅是
25 示例性说明，可以根据具体的应用需要进行调整。

基于上述标记方式，具体以对象 A 等同于上述实施例中涉及的第一对象，对象 A1 等同于上述实施例中涉及的第二对象，对象 B 等同于上述实施例中涉及的第三对象，对象 D 等同于上述实施例中涉及的第四对象，对象 C 等同于上述实施例中涉及的第五对象，举例说明如下：

5 如图 10 所示，该并行标记处理方法包括：

步骤 1：线程启动。

步骤 2：如果私有 stack 已经空了，则跳转到步骤 3。否则弹出一个对象比如 A，跳转到步骤 4。

步骤 3：如果 buf-queue 为空，跳转到步骤 7，否则遍历 buf-queue，弹出对象比如 B，跳转到步骤 5。如果已经遍历完成，则跳转到步骤 7。

步骤 4：依次遍历对象 A 所引用的所有对象，遍历完成则跳转到步骤 2。否则，比如遍历到对象 A1，并跳转到步骤 9。

步骤 5：尝试把步骤 3 中弹出来的对象 B 压入本线中 T(B)号 queue 中，压入成功则跳转到步骤 3，反之标记对象 B 所在的 page（内存块）溢出，
15 然后跳转到步骤 6。

步骤 6：对象 B 重新压回到 buf-queue 中，这时候不会失败，因为对象 B 是从 buf-queue 中弹出来的。

步骤 7：遍历另外 N-1 个线程中的本线程对应的 queue，取出标记对象。只要有一个 queue 非空，则 pop 出（弹出）标记对象，比如对象 D，然后
20 跳转到步骤 16。如果所有 N-1 个线程中和本线程对应的 queue 都为空，则跳转到步骤 8。

步骤 8：遍历本线程负责的 page 时如发现有溢出，则遍历有溢出的 page 的 bitmap（位图），找到灰色的对象比如 C，如果找到则跳转到步骤 14。遍历完毕，清理 page 的溢出标志，然后跳转到步骤 15。

25 步骤 9：判断步骤 4 中遍历到的对象 A1 所在的 page 的对应的 bitmap

是否为白色，如果已经是黑色则跳转到步骤 4。如果是白色，则跳转到步骤 10。

步骤 10: 标记对象 A1 所在的 page 的对应的 bitmap 值为黑色。如果 T(A1)为本线程，则跳转到步骤 17。否则跳转到步骤 11。

5 步骤 11: 尝试把对象 A1 压入本线程的 T(A1)号 queue 中，如果压入失败，则设置 A1 所在的 Page 的溢出。并跳转到步骤 12，如果成功，则跳转到步骤 4。

步骤 12: 尝试把对象 A1 压入 buf-queue，如果成功则跳转到步骤 4。反之跳转到步骤 13。

10 步骤 13: 设置对象 A1 所在的 page 对应的 bitmap 的值为灰色，并设置 page 的溢出标志。

步骤 14: 把步骤 8 遍历到的灰色对象 C 标记对应的 bitmap 为黑色并压入私有栈 stack 中，如果成功则跳转到步骤 8。如果失败，则设置 C 所在的 Page 的溢出标志，并跳转到步骤 19。

15 步骤 15: 如果标记结束，则线程退出，否则跳转到步骤 2，继续并行标记。

步骤 16: 标记对象 D 对应的 bitmap 为黑色，尝试把对象 D 压入本线程的 stack 中，如压入成功，跳转到步骤 7。溢出跳转到步骤 19。

20 步骤 17: 尝试把对象 A1 压入本线程的私有 stack 中，如果压入成功即没有溢出，则跳转到步骤 4，否则跳转到步骤 18。

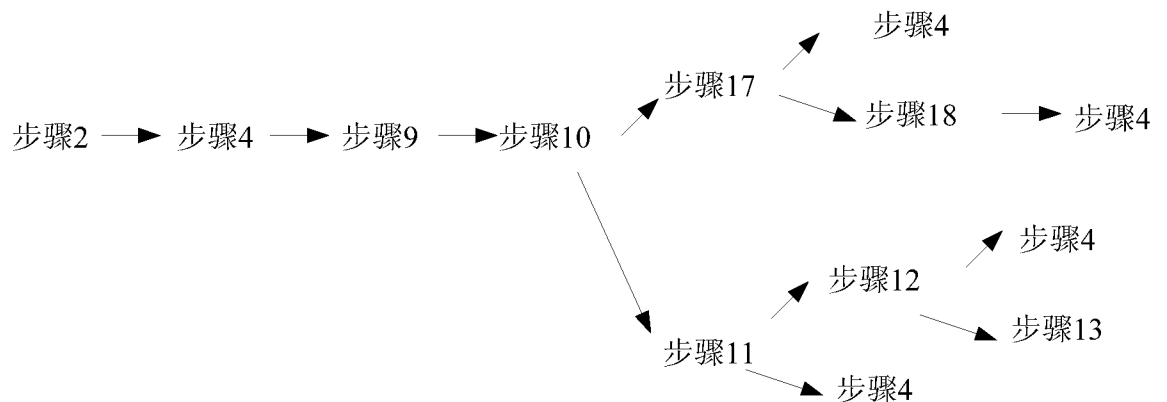
步骤 18: 标记对象 A1 对应的 bitmap 为灰色，并设置 A1 所在的 Page 的溢出标志。然后跳转到步骤 4。

25 步骤 19: 标记对象 (D 或者 C，即可能来自步骤 14 或者 16) 对应的 bitmap 为灰色，并设置标记对象所在的 Page 的溢出标志。然后跳转到步骤 2。

具体地，结合本示例中的 19 个步骤和附图，具体通过下述的处理步骤示意图 6 至图 9 的处理过程，其详细的描述过程参见图 6 至图 9 所示实施例，此处不再赘述。

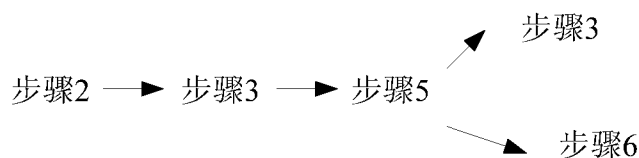
针对图 6 所示实施例的处理过程如下示意的步骤流程 (1)，其具体实施过程参见上述实施例，此处不再赘述。

步骤流程 (1):



针对图 7 所示实施例的处理过程如下示意的步骤流程 (2)，其具体实施过程参见上述实施例，此处不再赘述。

步骤流程 (2):



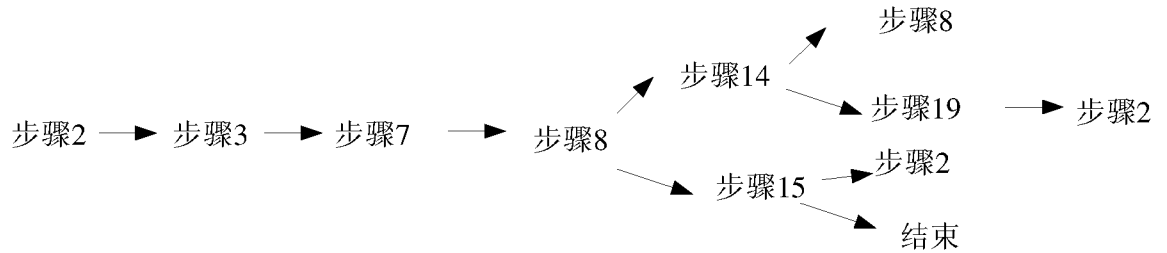
针对图 8 所示实施例的处理过程如下示意的步骤流程 (3)，其具体实施过程参见上述实施例，此处不再赘述。

步骤流程 (3):



针对图 9 所示实施例的处理过程如下示意的步骤流程 (4)，其具体实施过程参见上述实施例，此处不再赘述。

步骤流程 (4):



5 为了实现上述实施例，本申请还提出一种用于垃圾回收的并行标记处理装置。

图 11 是本申请一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

参见图 11，该用于垃圾回收的并行标记处理装置包括：

10 遍历模块 11，用于根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 为大于 1 的整数，N 个标记线程占用的内存容量是预先设置的，每个标记线程至少包括：1 个私有栈；

15 第一标记模块 12，用于将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理；

启动模块 13，用于对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。

20 需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，通过根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 个标记线程占用的内存容量是预先设置的，将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象进行第一标记处理，
5 对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。由此，利用有限的内存实现多线程的并行标记处理，提高整个垃圾回收的性能。

10 图 12 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

参见图 12，基于图 11，该用于垃圾回收的并行标记处理装置还包括：
申请模块 14，用于申请所述 N 个标记线程；

15 建立模块 15，用于遍历所述虚拟机堆中的内存块，建立内存块与 N 个标记线程的对应关系。

具体地，所述建立模块 15 用于：

为每个内存块指定对应的标记线程；或者，

为每个标记线程指定负责标记的内存块。

其中，在一个具体的示例中，所述建立模块 15 包括：

20 分配单元 151，用于遍历虚拟机堆中的内存块，为每个内存块分配编号；

确定单元 152，用于根据内存块的编号和标记线程总数 N，确定与每个内存块编号对应的标记线程编号。

具体地，所述确定单元 152 用于：

25 获取标记线程总数 N 被内存块编号整除后的余数，确定与每个内存块

编号对应的标记线程编号。

具体地，所述第一标记模块 12 用于：

如果将所述第一指针成功压入与当前第一对象所在的内存块对应的
标记线程的私有栈，则将当前第一对象所在内存块的位图中、与该第一对
5 象对应的状态标记为压入状态；

如果未能将所述第一指针成功压入与当前第一对象所在的内存块对
应的标记线程的私有栈，则将当前第一对象所在内存块的位图中、与该第
一对象对应的状态标记为溢出状态。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解
10 释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再
赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，通过申请 N 个标
记线程，遍历虚拟机堆的所有内存块，为每个内存块分配编号；根据内存
块编号和标记线程总数 N，确定与每个内存块编号对应的标记线程编号，
15 进而将当前处理的第一对象的第一指针压入与该第一对象所在的内存块
对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象
进行第一标记处理，对所述第一对象遍历完成后，向所述 N 个标记线程发
送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针
的压入情况，同步进行用于垃圾回收的标记处理。由此，利用有限的内存
20 实现高效的并行标记，同时实现标记线程负载均衡，提高整个垃圾回收的
性能。

图 13 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的
结构示意图。

参见图 13，本实施例中预先申请的 N 个标记线程，以及 N 个标记线
25 程所占用的内存容量。N 个标记线程包括：Thread-1、Thread-2，...Thread-N。

其中，每个标记线程至少包括：1 个私有栈 (stack)，经过上述处理，各个私有栈中存储有本线程负责标记的第一对象 (Root Objects) 的第一指针。本实施例中用于垃圾回收的并行标记处理装置包括：

获取模块 21，用于根据线程启动指令，从第一私有栈中取出预先压入
5 的第一对象的第一指针；

第二标记模块 22，用于遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再赘述。
10

本实施例提供的用于垃圾回收的并行标记处理装置，通过第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针，遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设
15 的 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的并行标记处理，实现了利用有限的内存进行并行标记处理，提高整个垃圾回收的性能。

图 14 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。
20

参见图 14，基于图 13 所示实施例，所述第二标记模块 22 包括：

第一判断单元 221，用于根据所述对应关系，判断是否负责对当前处理的第二对象所在的内存块 P1 进行标记处理；

第一处理单元 222，用于如果所述第一标记线程负责对所述内存块 P1
25 进行标记处理，则将当前处理的第二对象的第二指针压入到所述第一私有

栈中；

第二判断单元 223，用于判断是否成功将所述第二指针压入到所述第一私有栈中；

5 第二处理单元 224，用于如果所述第一标记线程未能成功将所述第二指针压入到所述第一私有栈中，则将所述内存块 P1 的位图中、与该第二对象对应的状态标记为溢出状态。

进一步地，所述第二处理单元 224 还用于：

设置与所述内存块 P1 对应的溢出标志。

进一步地，在另一个实施例中，

10 所述第二处理单元 224 还用于：如果所述第一标记线程成功将所述第二指针压入到所述第一私有栈中，则继续对所述第一指针对应的下一个第二对象进行处理。

进一步地，在另一个实施例中，所述第二标记模块 22 还包括：

15 所述第一处理单元 222，还用于如果所述第一标记线程不负责将所述内存块 P1 进行标记处理，确定标记线程 M 负责对所述内存块 P1 进行标记处理，将所述第二指针压入到所述第一标记线程中与所述标记线程 M 对应的公用输出队列 M 中；

第三判断单元 225，用于判断是否成功将所述第二指针压入到所述公用输出队列 M 中；

20 第三处理单元 226，用于如果所述第一标记线程成功将所述第二指针压入到所述公用输出队列 M 中，则继续对所述第一指针对应的下一个第二对象进行处理。

进一步地，在另一个实施例中，所述第二标记模块 22 还包括：

25 所述第三处理单元 226，还用于如果所述第一标记线程未能成功将所述第二指针压入到所述公用输出队列 M 中，则把所述第二指针压入到所述

第一标记线程中的第一缓存队列中；

第四判断单元 227，用于判断是否成功将所述第二指针压入到所述第一缓存队列中；

5 第四处理单元 228，用于如果所述第一标记线程未能成功将所述第二指针压入到所述第一缓存队列中，则将所述内存块 P1 的位图中、与当前处理的第二对象对应的状态标记为溢出状态。

进一步地，在另一个实施例中，

所述第四处理单元 228 还用于：

设置与所述内存块 P1 对应的溢出标志。

10 进一步地，在另一个实施例中，

所述第四处理单元 228，还用于如果所述第一标记线程成功将所述第二指针压入到所述第一缓存队列中，则继续对所述第一指针对应的下一个第二对象进行处理。

进一步地，在另一个实施例中，

15 所述第二处理单元 224，还用于查询所述内存块 P1 的位图中与该第二对象对应的状态标记；如果与该第二对象对应的状态标记为未标记状态，则将与该第二对象对应的状态修改为压入状态。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再
20 赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，通过第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针，遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二
25 标记处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程

的并行标记处理，利用有限的内存实现高效的并行标记，同时实现标记线程负载均衡，从而减少 CPU 缓存一致性冲突，提高整个垃圾回收的性能。

图 15 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

5 参见图 15，基于图 13 所示实施例，本实施例中预设的 N 个标记线程中每个标记线程还包括：1 个缓存队列，以及与其他 (N-1) 个标记线程对应的 (N-1) 个公用输出队列；所述第二标记模块 22 还包括：

第五判断单元 229，用于判断所述第一私有栈是否为空；

10 第六判断单元 210，用于如果所述第一私有栈为空，判断第一标记线程的第一缓存队列是否为空；

第五处理单元 211，用于如果所述第一缓存队列不为空，所述第一标记线程从所述第一缓存队列中取出预先压入的第三对象的第三指针，其中，所述第一标记线程不负责当前处理的第三对象所在的内存块 P2 进行标记处理；根据所述对应关系，确定标记线程 W 负责当前处理的第三对象所在的内存块 P2 进行标记处理；将所述第三指针压入到所述第一标记线程中与所述标记线程 W 对应的公用输出队列 W 中；

第七判断单元 212，用于判断是否成功将当前处理的第三指针压入到所述公用输出队列 W 中；

20 第六处理单元 213，用于如果所述第一标记线程成功将所述第三指针压入到所述公用输出队列 W 中，则继续对所述第一缓存队列中的下一个第三对象向对应的公用输出队列进行压入处理。

进一步地，在另一个实施例中，

25 所述第六处理单元 213，还用于如果所述第一标记线程未能成功将所述第三指针压入到所述公用输出队列 W 中，则将所述第三指针重新压入所述第一缓存队列中。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，在第一标记线程
5 的第一私有栈为空时，且第一缓存队列不为空，从第一缓存队列中取出预先压入的第三对象的第三指针，向本线程中与对应标记线程的公用输出队列中进行压入处理，以供与第三对象对应的标记线程对其进行标记处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的进行协同并行的标记处理，利用有限的内存实现高效的并行标记，同时实现标记线
10 程负载均衡，从而减少 CPU 缓存一致性冲突，提高整个垃圾回收的性能。

图 16 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

参见图 16，基于图 15 所示实施例，所述第二标记模块 22 还包括：

第八判断单元 214，用于如果所述第一缓存队列为空，判断其他 (N-1)
15 个标记线程中与所述第一标记线程对应的公用输出队列是否为空；

第七处理单元 215，用于如果当前处理的标记线程 T 中与所述第一标记线程对应的公用输出队列不为空，所述第一标记线程从当前处理的公用输出队列中取出预先压入的第四对象的第四指针，其中，所述第四对象为所述标记线程 T 预先压入的、由所述第一标记线程负责对所述第四对象所
20 在的内存块 P3 进行标记处理；将所述第四指针压入到所述第一私有栈中；

第九判断单元 216，用于判断是否成功将所述第四指针压入到所述第一私有栈中；

第八处理单元 217，用于如果所述第一标记线程未能成功将所述第四指针压入到所述第一私有栈中，则将所述内存块 P3 的位图中、与该第四
25 对象对应的状态标记为溢出状态。

进一步地，在另一个实施例中，

所述第八处理单元 217，还用于设置与所述内存块 P3 对应的溢出标志。

进一步地，在另一个实施例中，

所述第八处理单元 217，还用于如果所述第一标记线程成功将所述第四指针压入到所述第一私有栈中，则继续对下一个第四对象向所述第一私有栈中进行压入处理。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，第一标记线程在第一私有栈和第一缓存队列为空，且其他标记线程中与第一标记线程对应的公用输出队列不为空时，取出预先压入的第四对象的第四指针，向本线程中的第一私有栈压入处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的进行协同并行的标记处理，利用有限的内存实现高效的并行标记，同时实现标记线程负载均衡，从而减少 CPU 缓存一致性冲突，提高整个垃圾回收的性能。

图 17 是本申请另一个实施例的用于垃圾回收的并行标记处理装置的结构示意图。

参见图 17，基于图 16 所示实施例，所述第二标记模块 22 还包括：

第十判断单元 218，用于如果其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列都为空，所述第一标记线程判断本线程负责标记的内存块中是否有溢出状态的第五对象；

第九处理单元 219，用于如果所述第一标记线程负责标记的内存块中具有溢出状态的第五对象，所述第一标记线程将当前处理的内存块 P4 中标记溢出状态的第五对象修改为压入状态，将当前第五对象的第五指针压

入到所述第一私有栈中；

第十一判断单元 220，用于所述第一标记线程判断是否成功将所述第五指针压入到所述第一私有栈中；

第十处理单元 221，用于如果所述第一标记线程未能成功将所述第五指针压入到所述第一私有栈中，则将所述内存块 P4 的位图中、与该第五对象对应的状态标记为溢出状态。

进一步地，在另一个实施例中，

所述第十处理单元 221，还用于设置与所述内存块 P5 对应的溢出标志。

进一步地，在另一个实施例中，

所述第十处理单元 221，还用于如果所述第一标记线程成功将所述第五指针压入到所述第一私有栈中，则继续对下一个第五对象进行处理。

进一步地，在另一个实施例中，

所述第九处理单元 219，还用于如果所述第一标记线程负责标记的内存块中不具有溢出状态的第五对象，则标记结束，退出线程。

需要说明的是，前述对用于垃圾回收的并行标记处理方法实施例的解释说明也适用于该实施例的用于垃圾回收的并行标记处理装置，此处不再赘述。

本实施例提供的用于垃圾回收的并行标记处理装置，第一标记线程在第一私有栈、第一缓存队列、以及其他标记线程中与第一标记线程对应的公用输出队列都为空时，根据本线程负责标记的具有溢出状态的第五对象进行压入处理。由此，参照第一标记线程的标记处理过程，通过 N 个标记线程的并行的标记处理，利用有限的内存实现高效的并行标记，提高整个垃圾回收的性能。

在本说明书的描述中，参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的

具体特征、结构、材料或者特点包含于本申请的至少一个实施例或示例中。在本说明书中，对上述术语的示意性表述不必须针对的是相同的实施例或示例。而且，描述的具体特征、结构、材料或者特点可以在任一个或多个实施例或示例中以合适的方式结合。此外，在不相互矛盾的情况下，本领域的技术人员可以将本说明书中描述的不同实施例或示例以及不同实施例或示例的特征进行结合和组合。

此外，术语“第一”、“第二”仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或者隐含地包括至少一个该特征。在本申请的描述中，“多个”的含义是至少两个，例如两个，三个等，除非另有明确具体的限定。

流程图中或在此以其他方式描述的任何过程或方法描述可以被理解为，表示包括一个或更多个用于实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分，并且本申请的优选实施方式的范围包括另外的实现，其中可以不按所示出或讨论的顺序，包括根据所涉及的功能按基本同时的方式或按相反的顺序，来执行功能，这应被本申请的实施例所属技术领域的技术人员所理解。

权 利 要 求 书

1、一种用于垃圾回收的并行标记处理方法，其特征在于，包括以下步骤：

5 根据虚拟机堆中的内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象，其中，N 为大于 1 的整数，N 个标记线程占用的内存容量是预先设置的，每个标记线程至少包括：1 个私有栈；

将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈，根据所述第一指针的压入情况对所述第一对象
10 进行第一标记处理；

对所述第一对象遍历完成后，向所述 N 个标记线程发送线程启动指令，以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况，同步进行用于垃圾回收的标记处理。

2、如权利要求 1 所述的方法，其特征在于，所述根据虚拟机堆中的
15 内存块与 N 个标记线程的对应关系，遍历所述虚拟机堆中的第一对象之前，还包括：

申请所述 N 个标记线程；

遍历所述虚拟机堆中的内存块，建立内存块与 N 个标记线程的对应关系。

20 3、如权利要求 2 所述的方法，其特征在于，所述建立内存块与 N 个标记线程的对应关系，包括：

为每个内存块指定对应的标记线程；或者，

为每个标记线程指定负责标记的内存块。

4、如权利要求 2 所述的方法，其特征在于，所述遍历所述虚拟机堆
25 中的内存块，建立内存块与 N 个标记线程的对应关系，包括：

遍历虚拟机堆中的内存块，为每个内存块分配编号；

根据内存块的编号和标记线程总数 N，确定与每个内存块编号对应的标记线程编号。

5、如权利要求 4 所述的方法，其特征在于，所述根据内存块编号和
30 标记线程总数 N，确定与每个内存块编号对应的标记线程编号，包括：

获取标记线程总数 N 被内存块编号整除后的余数，确定与每个内存块编号对应的标记线程编号。

6、如权利要求 1 所述的方法，其特征在于，所述根据所述第一指针的压入情况对所述第一对象进行第一标记处理，包括：

5 如果将所述第一指针成功压入与当前第一对象所在的内存块对应的标记线程的私有栈，则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为压入状态；

如果未能将所述第一指针成功压入与当前第一对象所在的内存块对应的标记线程的私有栈，则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为溢出状态。

7、一种用于垃圾回收的并行标记处理方法，其特征在于，预设的 N 个标记线程中的每个标记线程至少包括：1 个私有栈，所述方法应用在每个标记线程中，其中，应用在所述第一标记线程中的所述方法包括以下步骤：

15 第一标记线程根据线程启动指令，从第一私有栈中取出预先压入的第一对象的第一指针；

所述第一标记线程遍历所述第一指针指示的所述第一对象引用的第二对象，并根据预设的 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。

8、如权利要求 7 所述的方法，其特征在于，所述根据 N 个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理，包括：

所述第一标记线程根据所述对应关系，判断是否负责对当前处理的第二对象所在的内存块 P1 进行标记处理；

25 如果所述第一标记线程负责对所述内存块 P1 进行标记处理，则将当前处理的第二对象的第二指针压入到所述第一私有栈中；

所述第一标记线程判断是否成功将所述第二指针压入到所述第一私有栈中；

30 如果所述第一标记线程未能成功将所述第二指针压入到所述第一私有栈中，则将所述内存块 P1 的位图中、与该第二对象对应的状态标记为溢出状态。

9、如权利要求 8 所述的方法，其特征在于，还包括：

设置与所述内存块 P1 对应的溢出标志。

10、如权利要求 8 所述的方法，其特征在于，所述第一标记线程判断是否成功将所述第二指针压入到所述第一私有栈中之后，还包括：

5 如果所述第一标记线程成功将所述第二指针压入到所述第一私有栈中，则继续对所述第一指针对应的下一个第二对象进行处理。

11、如权利要求 8 所述的方法，其特征在于，预设的 N 个标记线程中每个标记线程还包括：与其他 (N-1) 个标记线程对应的 (N-1) 个公用输出队列；

10 所述判断是否负责对当前处理的第二对象所在的内存块 P1 进行标记处理之后，还包括：

如果所述第一标记线程不负责对所内存块 P1 进行标记处理，确定标记线程 M 负责对所内存块 P1 进行标记处理；

15 将所述第二指针压入到所述第一标记线程中与所述标记线程 M 对应的公用输出队列 M 中；

所述第一标记线程判断是否成功将所述第二指针压入到所述公用输出队列 M 中；

如果所述第一标记线程成功将所述第二指针压入到所述公用输出队列 M 中，则继续对所述第一指针对应的下一个第二对象进行处理。

20 12、如权利要求 11 所述的方法，其特征在于，预设的 N 个标记线程中每个标记线程还包括：1 个缓存队列；

所述第一标记线程判断是否成功将所述第二指针压入到所述公用输出队列 M 中之后，还包括：

25 如果所述第一标记线程未能成功将所述第二指针压入到所述公用输出队列 M 中，则把所述第二指针压入到所述第一标记线程中的第一缓存队列中；

所述第一标记线程判断是否成功将所述第二指针压入到所述第一缓存队列中；

30 如果所述第一标记线程未能成功将所述第二指针压入到所述第一缓存队列中，则将所述内存块 P1 的位图中、与当前处理的第二对象对应的

状态标记为溢出状态。

13、如权利要求 12 所述的方法，其特征在于，还包括：

设置与所述内存块 P1 对应的溢出标志。

14、如权利要求 13 所述的方法，其特征在于，所述第一标记线程判断是否成功将所述第二指针压入到所述第一缓存队列中之后，还包括：

如果所述第一标记线程成功将所述第二指针压入到所述第一缓存队列中，则继续对所述第一指针对应的下一个第二对象进行处理。

15、如权利要求 8-14 任一所述的方法，其特征在于，所述第一标记线程根据所述对应关系，判断是否负责对当前处理的第二对象所在的内存块 P1 进行标记处理之前，还包括：

查询所述内存块 P1 的位图中与该第二对象对应的状态标记；

如果与该第二对象对应的状态标记为未标记状态，则将与该第二对象对应的状态修改为压入状态。

16、如权利要求 7 所述的方法，其特征在于，预设的 N 个标记线程中每个标记线程还包括：1 个缓存队列，以及与其他 (N-1) 个标记线程对应的 (N-1) 个公用输出队列；所述方法还包括：

判断本线程的第一私有栈是否为空；

如果所述第一私有栈为空，所述第一标记线程判断本线程的第一缓存队列是否为空；

如果所述第一缓存队列不为空，所述第一标记线程从所述第一缓存队列中取出预先压入的第三对象的第三指针，其中，所述第一标记线程不负责当前处理的第三对象所在的内存块 P2 进行标记处理；

所述第一标记线程根据所述对应关系，确定标记线程 W 负责对当前处理的第三对象所在的内存块 P2 进行标记处理；

所述第一标记线程将所述第三指针压入到所述第一标记线程中与所述标记线程 W 对应的公用输出队列 W 中；

所述第一标记线程判断是否成功将当前处理的第三指针压入到所述公用输出队列 W 中；

如果所述第一标记线程成功将所述第三指针压入到所述公用输出队列 W 中，则继续对所述第一缓存队列中的下一个第三对象向对应的公用输

出队列进行压入处理。

17、如权利要求 16 所述的方法，其特征在于，所述第一标记线程判断是否成功将当前处理的第三指针压入到所述公用输出队列 W 中之后，还包括：

5 如果所述第一标记线程未能成功将所述第三指针压入到所述公用输出队列 W 中，则将所述第三指针重新压入所述第一缓存队列中。

18、如权利要求 16 所述的方法，其特征在于，所述第一标记线程判断本线程的第一缓存队列是否为空之后，还包括：

10 如果所述第一缓存队列为空，所述第一标记线程判断其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列是否为空；

如果当前处理的标记线程 T 中与所述第一标记线程对应的公用输出队列不为空，所述第一标记线程从当前处理的公用输出队列中取出预先压入的第四对象的第四指针，其中，所述第四对象为所述标记线程 T 预先压入的、由所述第一标记线程负责对所述第四对象所在的内存块 P3 进行标记
15 处理；

所述第一标记线程将所述第四指针压入到所述第一私有栈中；

所述第一标记线程判断是否成功将所述第四指针压入到所述第一私有栈中；

20 如果所述第一标记线程未能成功将所述第四指针压入到所述第一私有栈中，则将所述内存块 P3 的位图中、与该第四对象对应的状态标记为溢出状态。

19、如权利要求 18 所述的方法，其特征在于，还包括：

设置与所述内存块 P3 对应的溢出标志。

20、如权利要求 18 所述的方法，其特征在于，所述第一标记线程判断是否成功将所述第四指针压入到所述第一私有栈中之后，还包括：
25

如果所述第一标记线程成功将所述第四指针压入到所述第一私有栈中，则继续对下一个第四对象向所述第一私有栈中进行压入处理。

21、如权利要求 18 所述的方法，其特征在于，所述第一标记线程判断其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列是否
30 是否为空之后，还包括：

如果其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列都为空, 所述第一标记线程判断本线程负责标记的内存块中是否有溢出状态的第五对象;

5 如果所述第一标记线程负责标记的内存块中具有溢出状态的第五对象, 所述第一标记线程将当前处理的内存块 P4 中标记溢出状态的第五对象修改为压入状态, 将当前第五对象的第五指针压入到所述第一私有栈中;

所述第一标记线程判断是否成功将所述第五指针压入到所述第一私有栈中;

10 如果所述第一标记线程未能成功将所述第五指针压入到所述第一私有栈中, 则将所述内存块 P4 的位图中、与该第五对象对应的状态标记为溢出状态。

22、如权利要求 21 所述的方法, 其特征在于, 还包括:

设置与所述内存块 P5 对应的溢出标志。

15 23、如权利要求 21 所述的方法, 其特征在于, 所述第一标记线程判断是否成功将所述第五指针压入到所述第一私有栈中之后, 还包括:

如果所述第一标记线程成功将所述第五指针压入到所述第一私有栈中, 则继续对下一个第五对象进行处理。

20 24、如权利要求 21 所述的方法, 其特征在于, 所述第一标记线程判断本线程负责标记的内存块中是否有溢出状态的第五对象之后, 还包括:

如果所述第一标记线程负责标记的内存块中不具有溢出状态的第五对象, 则标记结束, 退出线程。

25、一种用于垃圾回收的并行标记处理装置, 其特征在于, 包括:

25 遍历模块, 用于根据虚拟机堆中的内存块与 N 个标记线程的对应关系, 遍历所述虚拟机堆中的第一对象, 其中, N 为大于 1 的整数, N 个标记线程占用的内存容量是预先设置的, 每个标记线程至少包括: 1 个私有栈;

第一标记模块, 用于将当前处理的第一对象的第一指针压入与该第一对象所在的内存块对应的标记线程的私有栈, 根据所述第一指针的压入情况对所述第一对象进行第一标记处理;

30 启动模块, 用于对所述第一对象遍历完成后, 向所述 N 个标记线程发

送线程启动指令,以使所述 N 个标记线程根据各自私有栈中所述第一指针的压入情况,同步进行用于垃圾回收的标记处理。

26、如权利要求 25 所述的装置,其特征在于,还包括:

申请模块,用于申请所述 N 个标记线程;

5 建立模块,用于遍历所述虚拟机堆中的内存块,建立内存块与 N 个标记线程的对应关系。

27、如权利要求 26 所述的装置,其特征在于,所述建立模块用于:

为每个内存块指定对应的标记线程;或者,

为每个标记线程指定负责标记的内存块。

10 28、如权利要求 26 所述的装置,其特征在于,所述建立模块包括:

分配单元,用于遍历虚拟机堆中的内存块,为每个内存块分配编号;

确定单元,用于根据内存块的编号和标记线程总数 N,确定与每个内存块编号对应的标记线程编号。

29、如权利要求 28 所述的装置,其特征在于,所述确定单元用于:

15 获取标记线程总数 N 被内存块编号整除后的余数,确定与每个内存块编号对应的标记线程编号。

30、如权利要求 25 所述的装置,其特征在于,所述第一标记模块用于:

20 如果将所述第一指针成功压入与当前第一对象所在的内存块对应的标记线程的私有栈,则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为压入状态;

如果未能将所述第一指针成功压入与当前第一对象所在的内存块对应的标记线程的私有栈,则将当前第一对象所在内存块的位图中、与该第一对象对应的状态标记为溢出状态。

25 31、一种用于垃圾回收的并行标记处理装置,其特征在于,预设的 N 个标记线程中的每个标记线程至少包括: 1 个私有栈,所述装置应用在每个标记线程中,其中,应用在所述第一标记线程中的所述装置包括:

获取模块,用于根据线程启动指令,从第一私有栈中取出预先压入的第一对象的第一指针;

30 第二标记模块,用于遍历所述第一指针指示的所述第一对象引用的第

二对象,并根据预设的N个标记线程与虚拟机堆中内存块的对应关系对所述第二对象进行第二标记处理。

32、如权利要求31所述的装置,其特征在于,所述第二标记模块包括:

5 第一判断单元,用于根据所述对应关系,判断是否负责对当前处理的第二对象所在的内存块P1进行标记处理;

第一处理单元,用于如果所述第一标记线程负责对所述内存块P1进行标记处理,则将当前处理的第二对象的第二指针压入到所述第一私有栈中;

10 第二判断单元,用于判断是否成功将所述第二指针压入到所述第一私有栈中;

第二处理单元,用于如果所述第一标记线程未能成功将所述第二指针压入到所述第一私有栈中,则将所述内存块P1的位图中、与该第二对象对应的状态标记为溢出状态。

15 33、如权利要求32所述的装置,其特征在于,所述第二处理单元还用于:

设置与所述内存块P1对应的溢出标志。

34、如权利要求32所述的装置,其特征在于,所述第二处理单元还用于:

20 如果所述第一标记线程成功将所述第二指针压入到所述第一私有栈中,则继续对所述第一指针对应的下一个第二对象进行处理。

35、如权利要求32所述的装置,其特征在于,预设的N个标记线程中每个标记线程还包括:与其他(N-1)个标记线程对应的(N-1)个公用输出队列;所述第二标记模块还包括:

25 所述第一处理单元,还用于如果所述第一标记线程不负责对所述内存块P1进行标记处理,确定标记线程M负责对所述内存块P1进行标记处理,将所述第二指针压入到所述第一标记线程中与所述标记线程M对应的公用输出队列M中;

30 第三判断单元,用于判断是否成功将所述第二指针压入到所述公用输出队列M中;

第三处理单元，用于如果所述第一标记线程成功将所述第二指针压入到所述公用输出队列 M 中，则继续对所述第一指针对应的下一个第二对象进行处理。

36、如权利要求 35 所述的装置，其特征在于，预设的 N 个标记线程
5 中每个标记线程还包括：1 个缓存队列；所述第二标记模块还包括：

所述第三处理单元，还用于如果所述第一标记线程未能成功将所述第二指针压入到所述公用输出队列 M 中，则把所述第二指针压入到所述第一标记线程中的第一缓存队列中；

10 第四判断单元，用于判断是否成功将所述第二指针压入到所述第一缓存队列中；

第四处理单元，用于如果所述第一标记线程未能成功将所述第二指针压入到所述第一缓存队列中，则将所述内存块 P1 的位图中、与当前处理的第二对象对应的状态标记为溢出状态。

15 37、如权利要求 36 所述的装置，其特征在于，所述第四处理单元还用于：

设置与所述内存块 P1 对应的溢出标志。

38、如权利要求 36 所述的装置，其特征在于，

20 所述第四处理单元，还用于如果所述第一标记线程成功将所述第二指针压入到所述第一缓存队列中，则继续对所述第一指针对应的下一个第二对象进行处理。

39、如权利要求 31-38 任一所述的装置，其特征在于，

所述第二处理单元，还用于查询所述内存块 P1 的位图中与该第二对象对应的状态标记；

25 如果与该第二对象对应的状态标记为未标记状态，则将与该第二对象对应的状态修改为压入状态。

40、如权利要求 31 所述的装置，其特征在于，预设的 N 个标记线程中每个标记线程还包括：1 个缓存队列，以及与其他 (N-1) 个标记线程对应的 (N-1) 个公用输出队列；所述第二标记模块还包括：

第五判断单元，用于判断所述第一私有栈是否为空；

30 第六判断单元，用于如果所述第一私有栈为空，判断第一标记线程的

第一缓存队列是否为空；

第五处理单元，用于如果所述第一缓存队列不为空，所述第一标记线程从所述第一缓存队列中取出预先压入的第三对象的第三指针，其中，所述第一标记线程不负责当前处理的第三对象所在的内存块 P2 进行处理；根据所述对应关系，确定标记线程 W 负责当前处理的第三对象所在的内存块 P2 进行处理；将所述第三指针压入到所述第一标记线程中与所述标记线程 W 对应的公用输出队列 W 中；

第七判断单元，用于判断是否成功将当前处理的第三指针压入到所述公用输出队列 W 中；

第六处理单元，用于如果所述第一标记线程成功将所述第三指针压入到所述公用输出队列 W 中，则继续对所述第一缓存队列中的下一个第三对象向对应的公用输出队列进行压入处理。

41、如权利要求 40 所述的装置，其特征在于，

所述第六处理单元，还用于如果所述第一标记线程未能成功将所述第三指针压入到所述公用输出队列 W 中，则将所述第三指针重新压入所述第一缓存队列中。

42、如权利要求 40 所述的装置，其特征在于，所述第二标记模块还包括：

第八判断单元，用于如果所述第一缓存队列为空，判断其他 (N-1) 个标记线程中与所述第一标记线程对应的公用输出队列是否为空；

第七处理单元，用于如果当前处理的标记线程 T 中与所述第一标记线程对应的公用输出队列不为空，所述第一标记线程从当前处理的公用输出队列中取出预先压入的第四对象的第四指针，其中，所述第四对象为所述标记线程 T 预先压入的、由所述第一标记线程负责对所述第四对象所在的内存块 P3 进行处理；将所述第四指针压入到所述第一私有栈中；

第九判断单元，用于判断是否成功将所述第四指针压入到所述第一私有栈中；

第八处理单元，用于如果所述第一标记线程未能成功将所述第四指针压入到所述第一私有栈中，则将所述内存块 P3 的位图中、与该第四对象对应的状态标记为溢出状态。

43、如权利要求 42 所述的装置，其特征在于，

所述第八处理单元，还用于设置与所述内存块 P3 对应的溢出标志。

44、如权利要求 42 所述的装置，其特征在于，

所述第八处理单元，还用于如果所述第一标记线程成功将所述第四指针压入到所述第一私有栈中，则继续对下一个第四对象向所述第一私有栈中进行压入处理。

45、如权利要求 42 所述的装置，其特征在于，所述第二标记模块还包括：

第十判断单元，用于如果其他（N-1）个标记线程中与所述第一标记线程对应的公用输出队列都为空，所述第一标记线程判断本线程负责标记的内存块中是否有溢出状态的第五对象；

第九处理单元，用于如果所述第一标记线程负责标记的内存块中具有溢出状态的第五对象，所述第一标记线程将当前处理的内存块 P4 中标记溢出状态的第五对象修改为压入状态，将当前第五对象的第五指针压入到所述第一私有栈中；

第十一判断单元，用于所述第一标记线程判断是否成功将所述第五指针压入到所述第一私有栈中；

第十处理单元，用于如果所述第一标记线程未能成功将所述第五指针压入到所述第一私有栈中，则将所述内存块 P4 的位图中、与该第五对象对应的状态标记为溢出状态。

46、如权利要求 45 所述的装置，其特征在于，

所述第十处理单元，还用于设置与所述内存块 P5 对应的溢出标志。

47、如权利要求 45 所述的装置，其特征在于，

所述第十处理单元，还用于如果所述第一标记线程成功将所述第五指针压入到所述第一私有栈中，则继续对下一个第五对象进行处理。

48、如权利要求 45 所述的装置，其特征在于，

所述第九处理单元，还用于如果所述第一标记线程负责标记的内存块中不具有溢出状态的第五对象，则标记结束，退出线程。

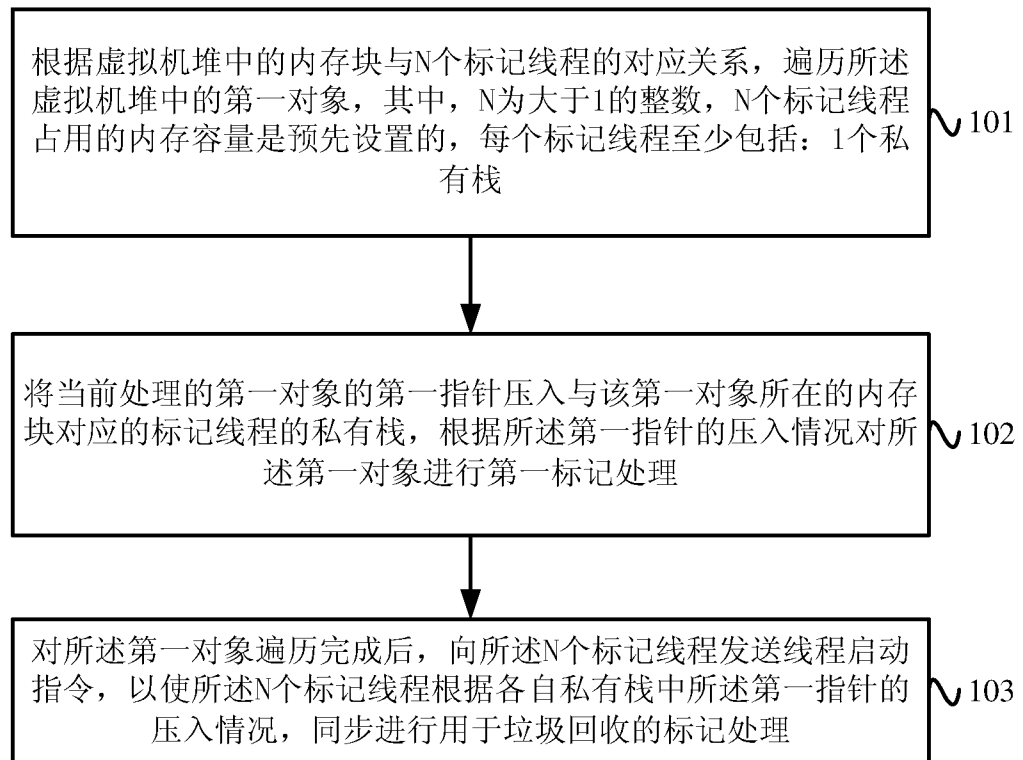


图 1

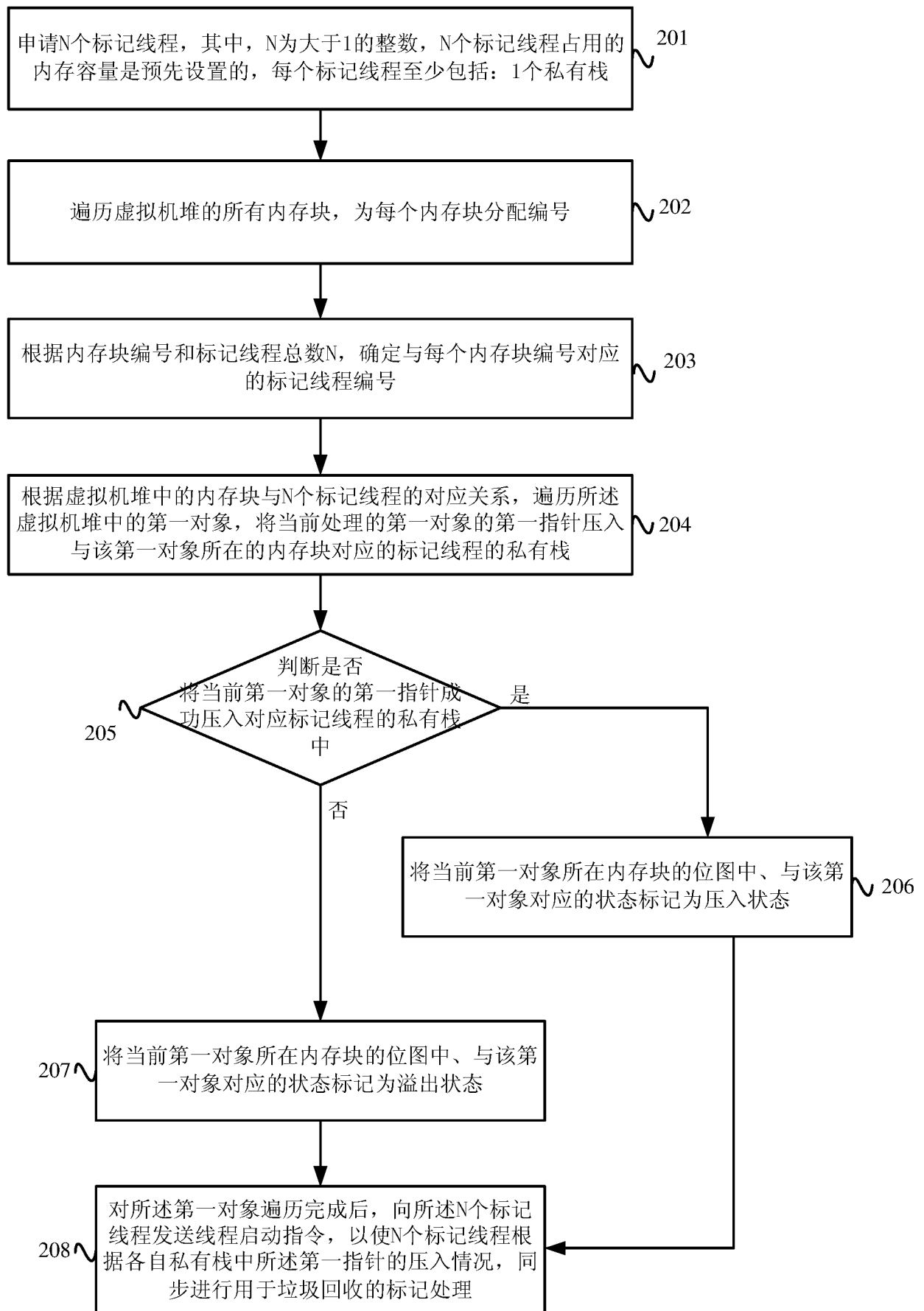


图 2

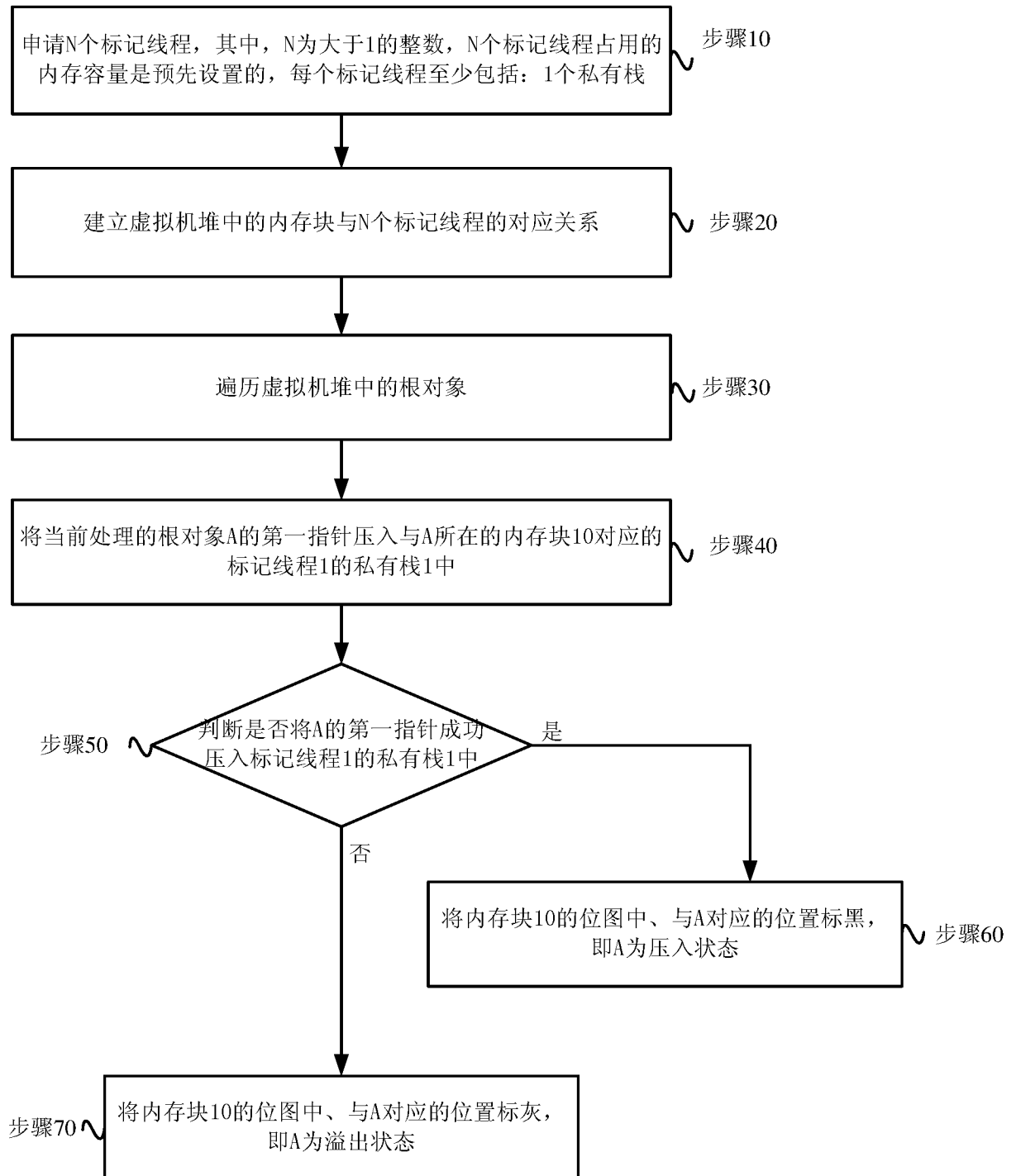


图 3

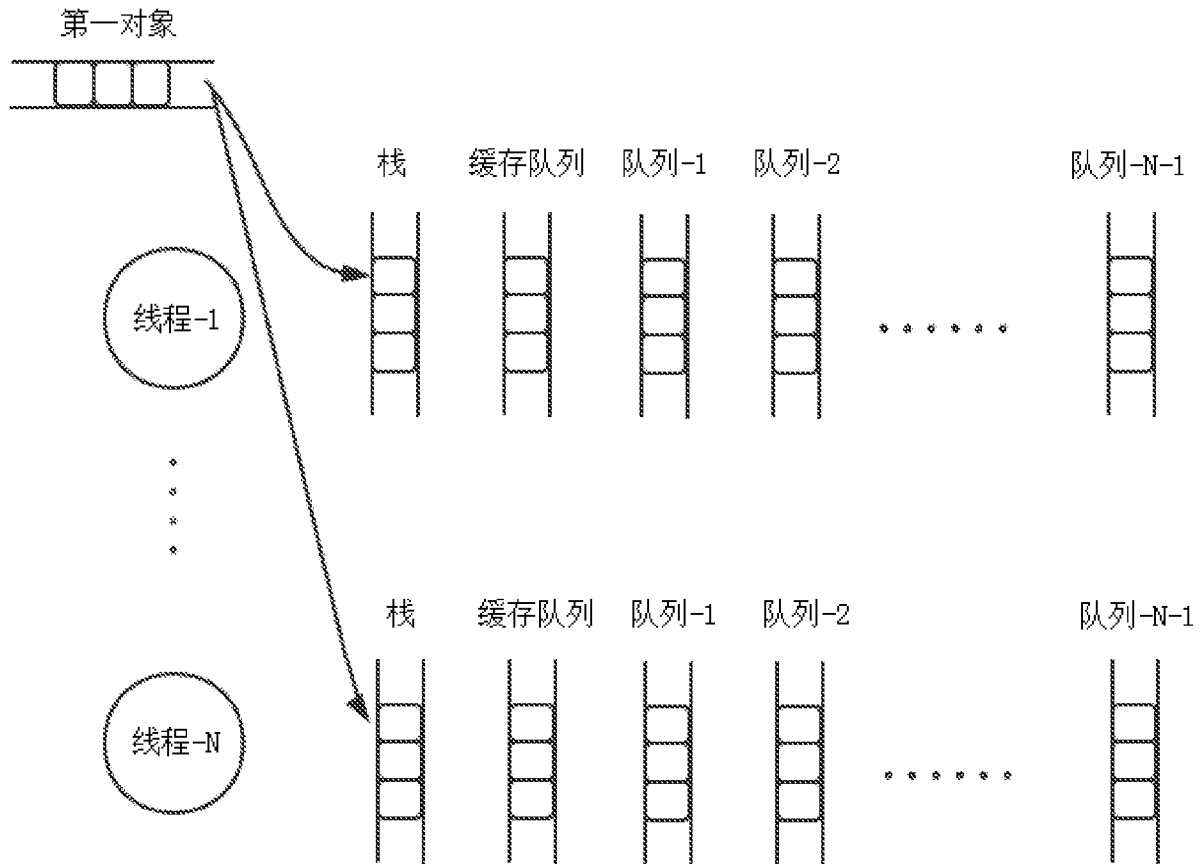


图 4

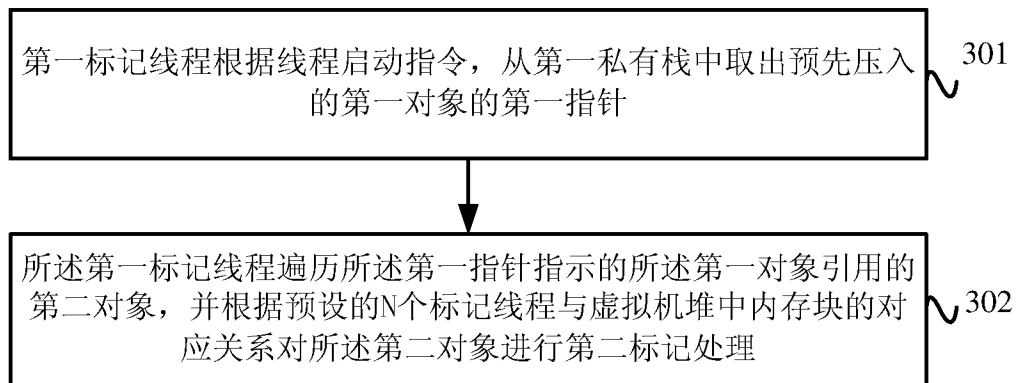


图 5

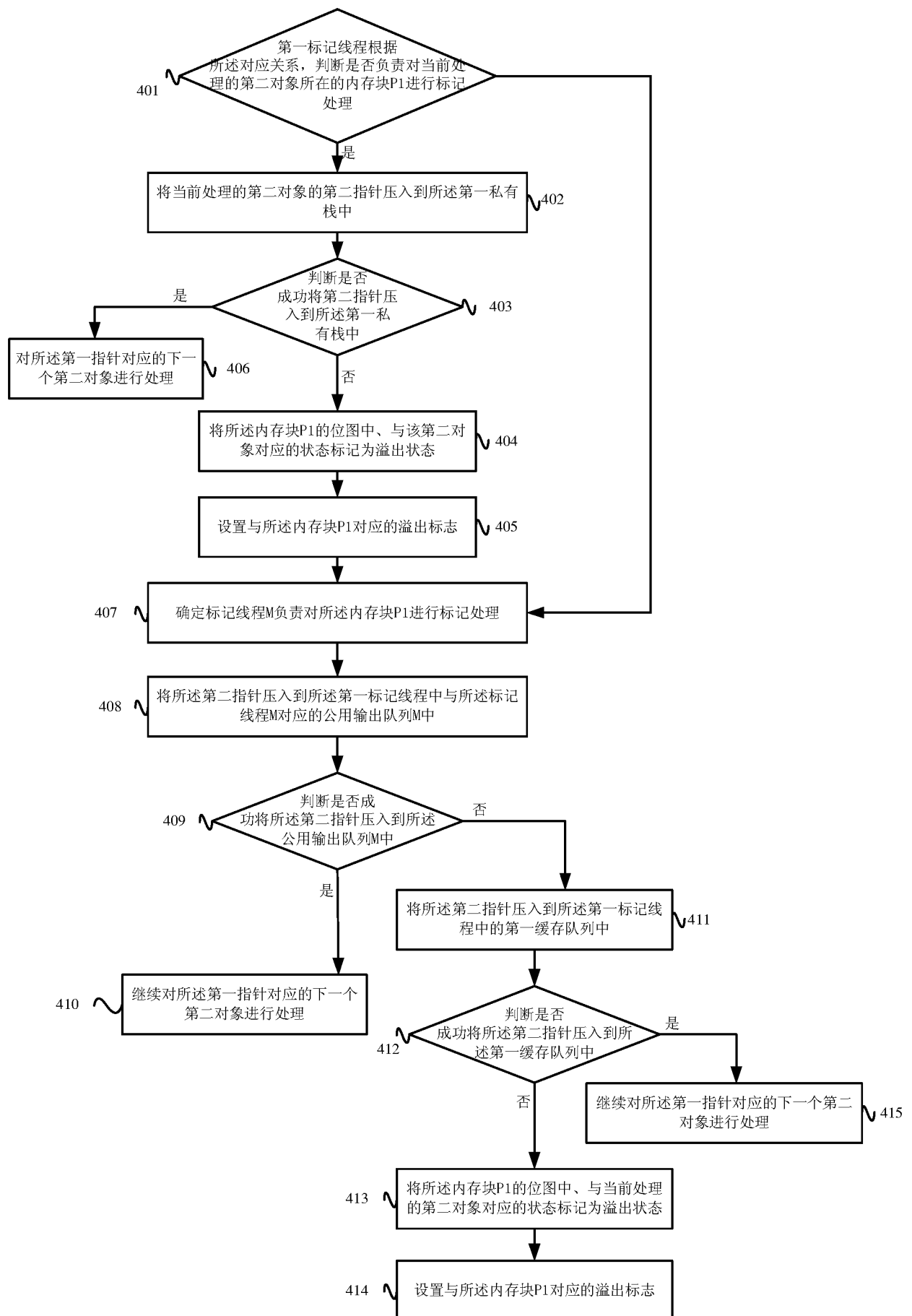


图 6

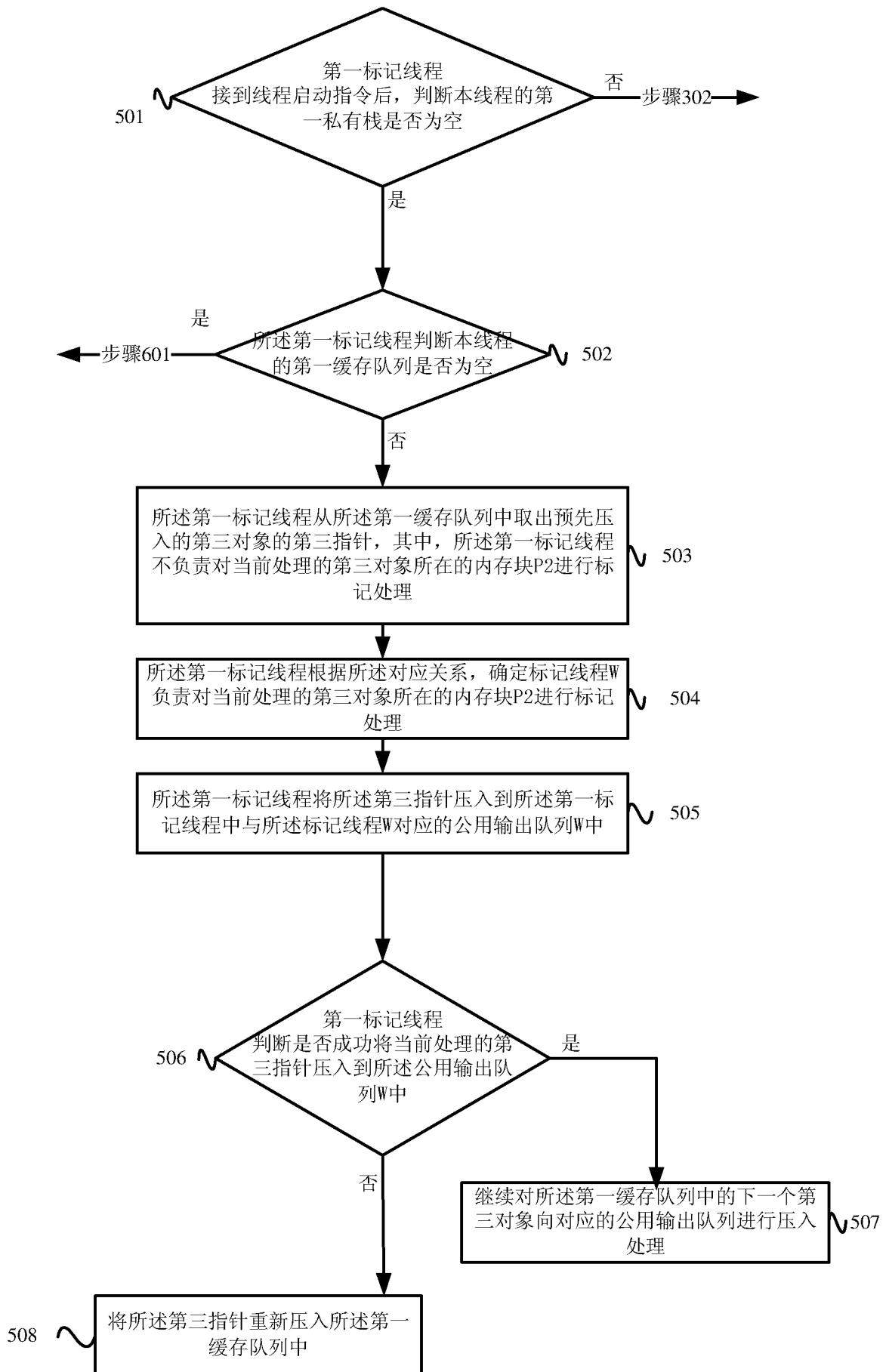


图 7

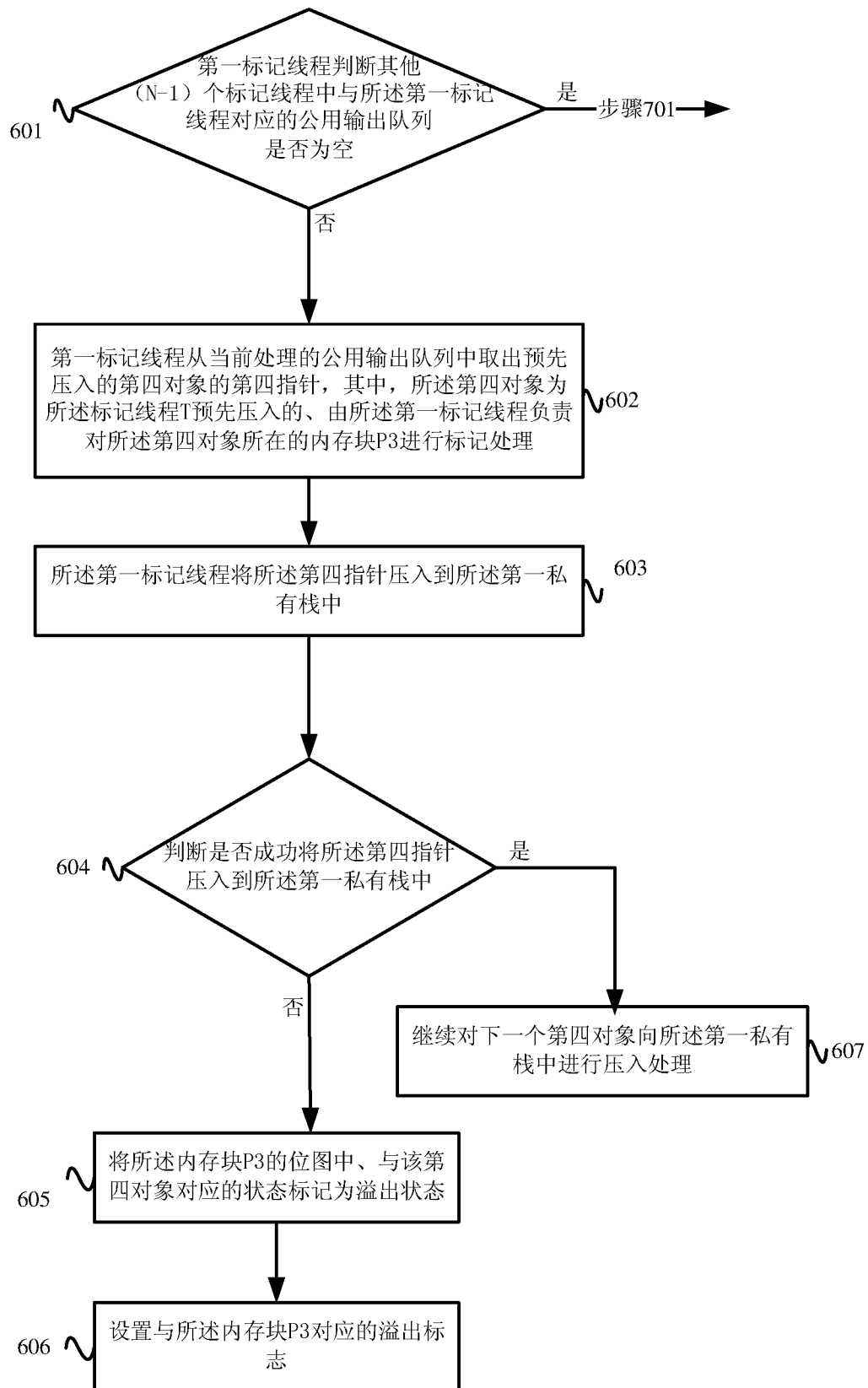


图 8

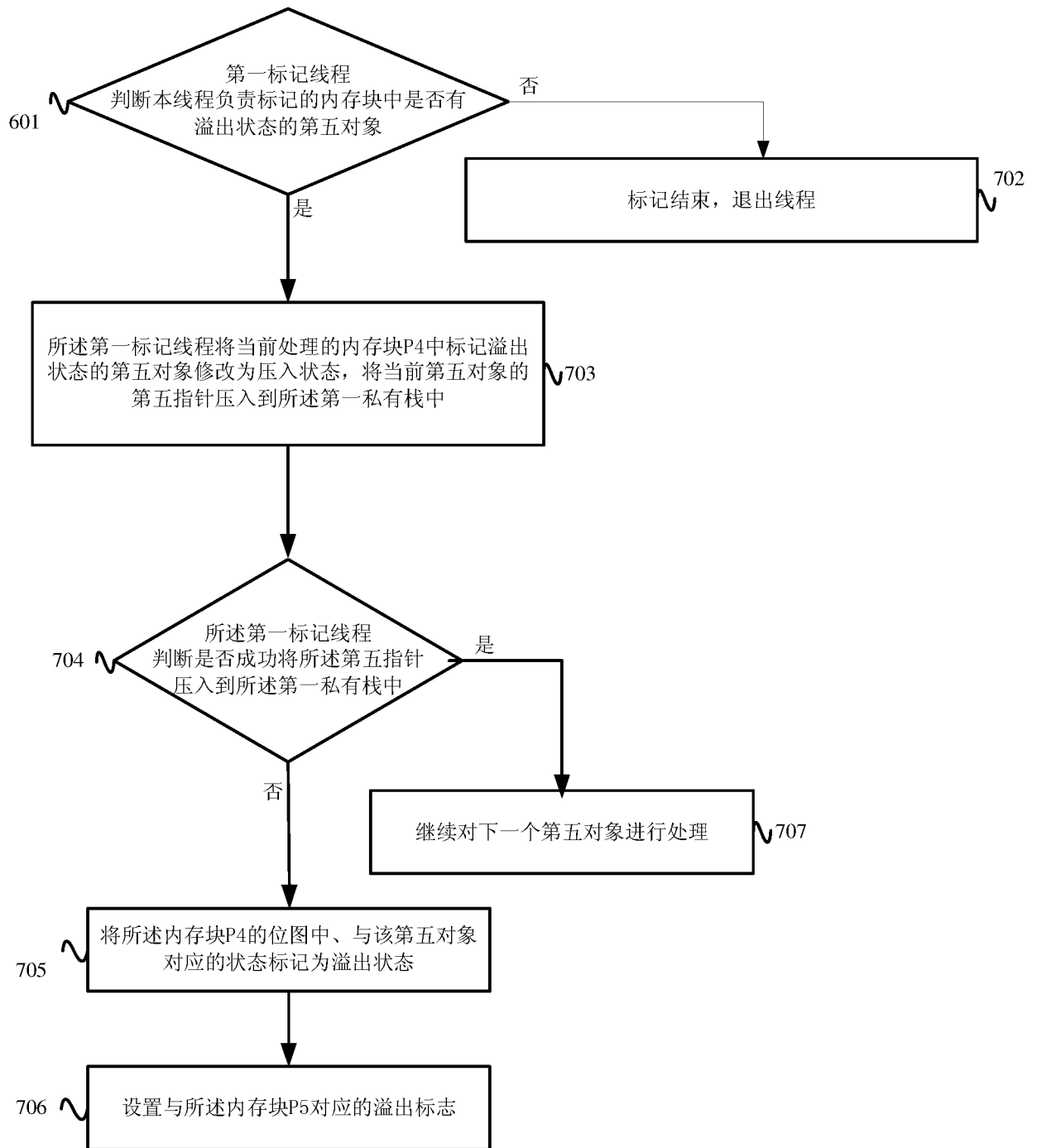


图 9

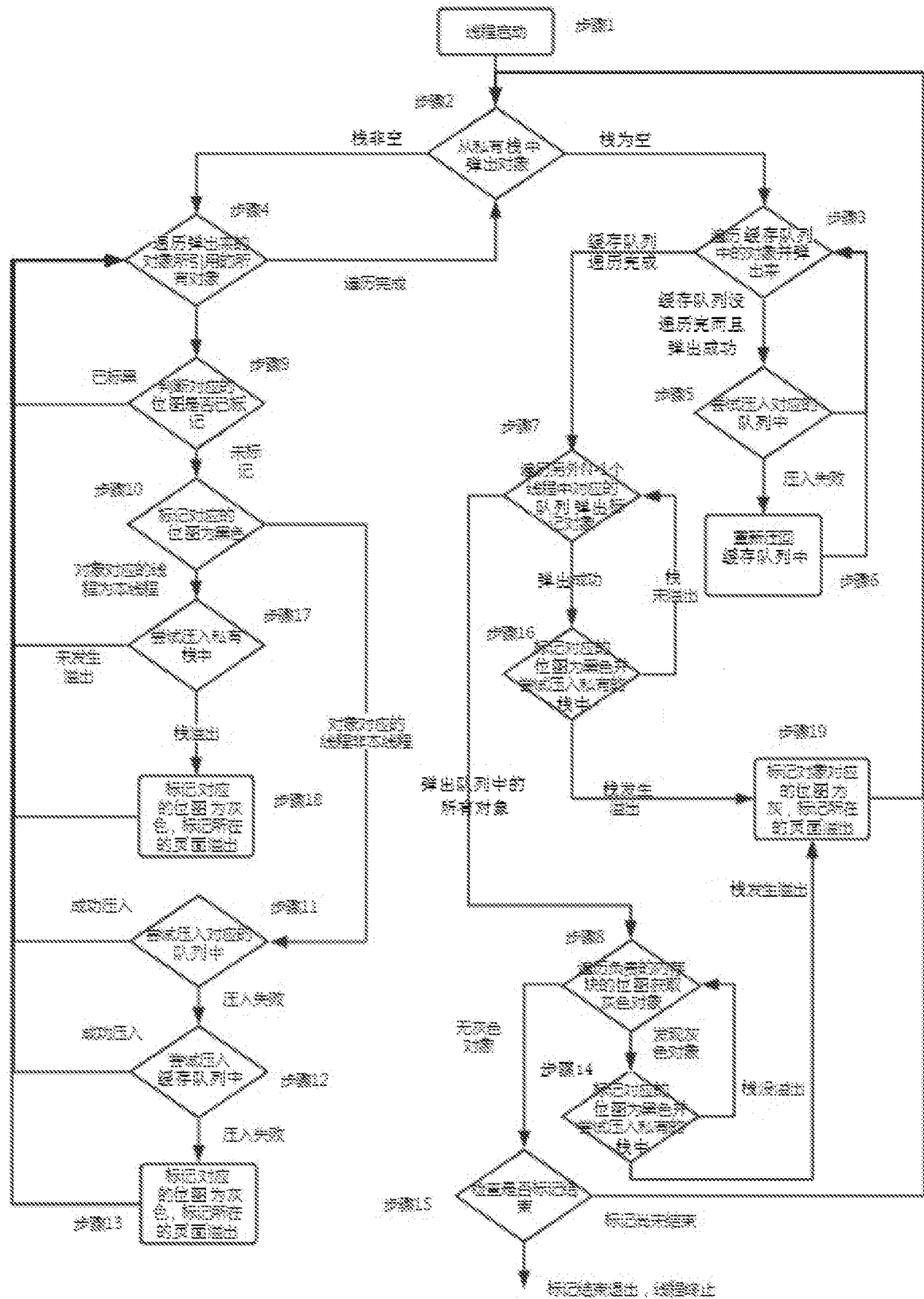


图 10

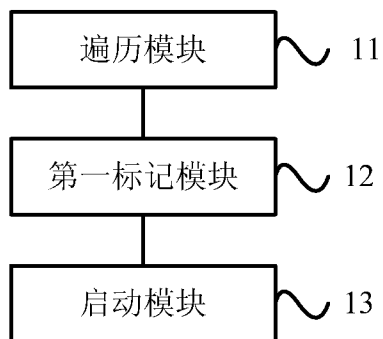


图 11

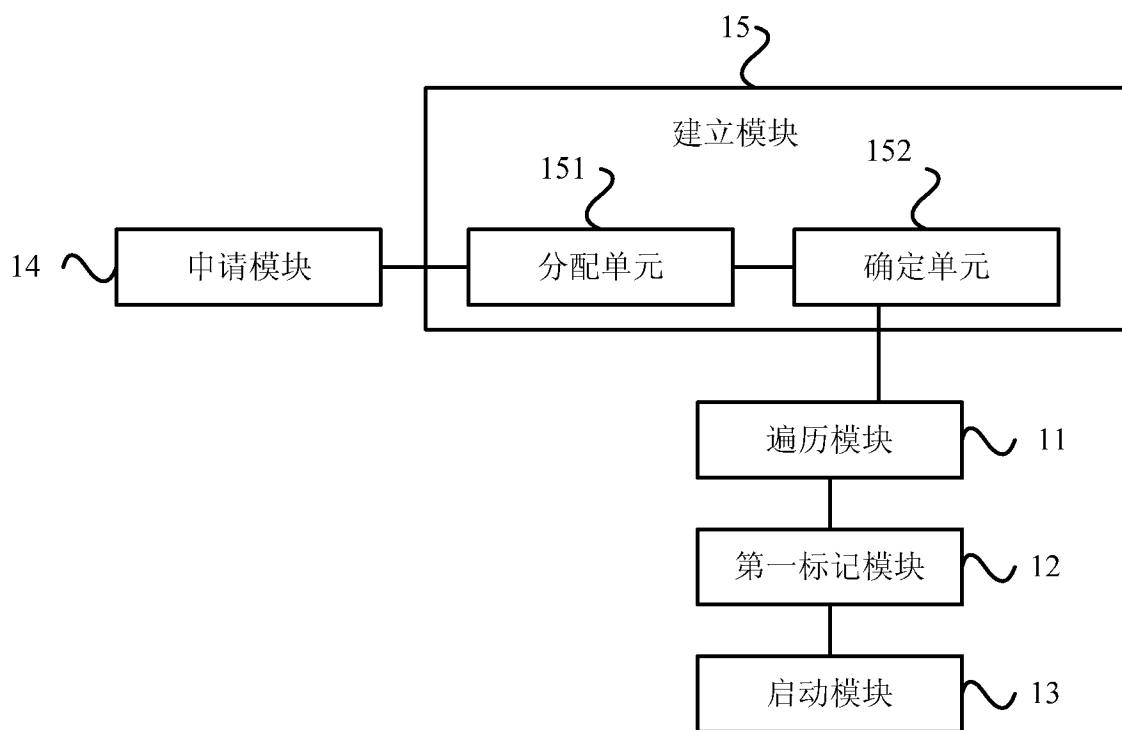


图 12

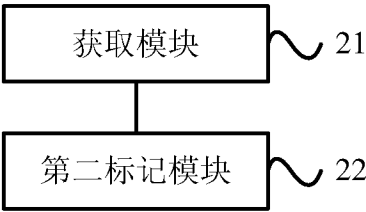


图 13

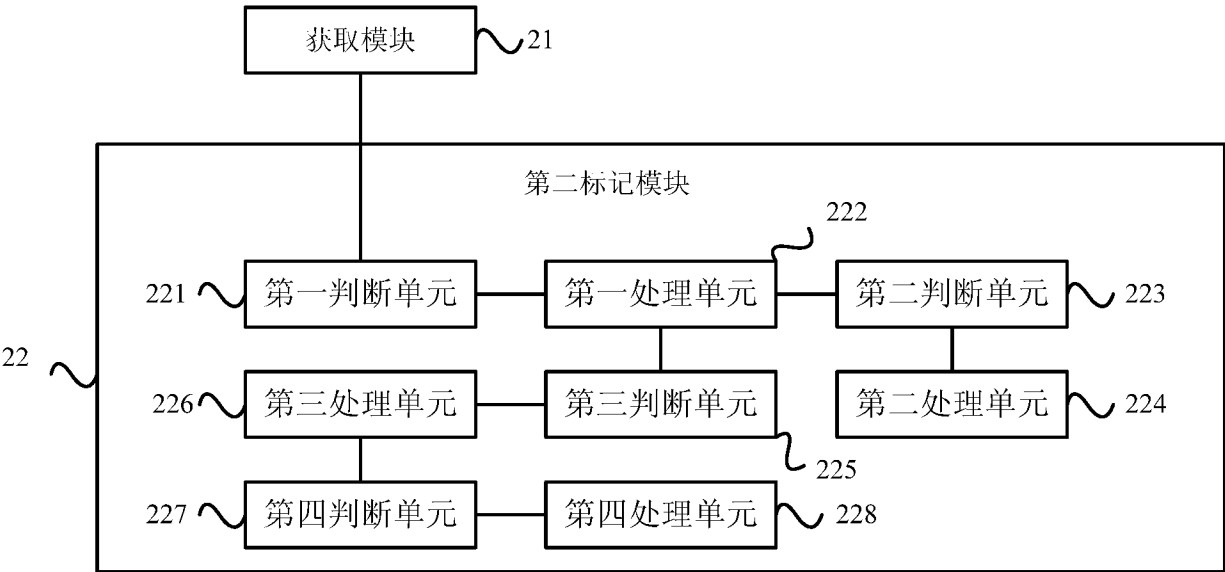


图 14

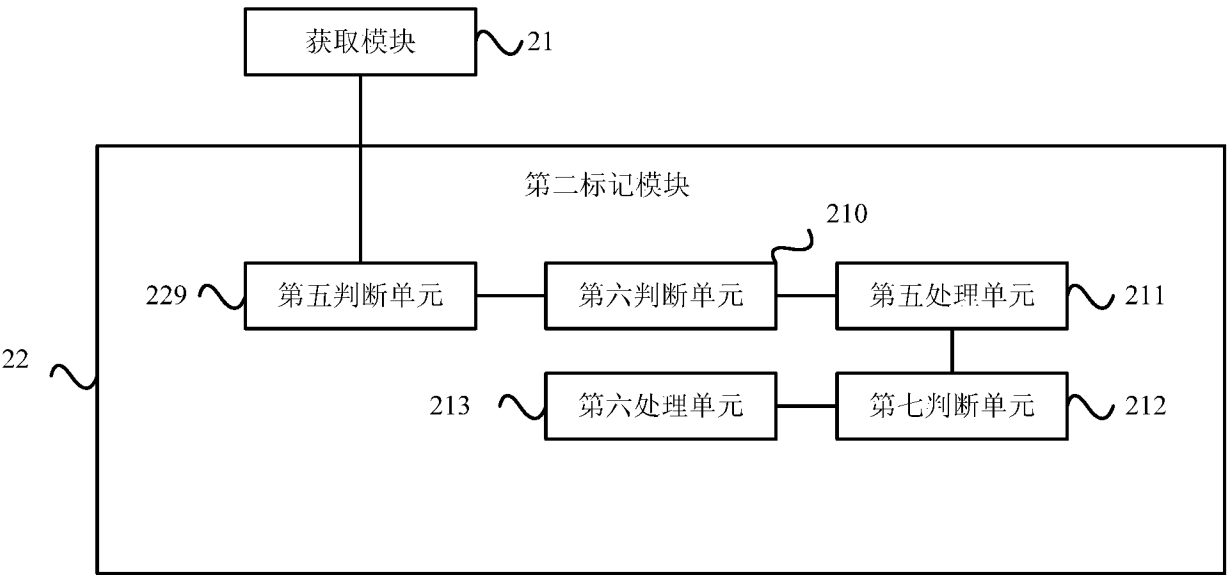


图 15

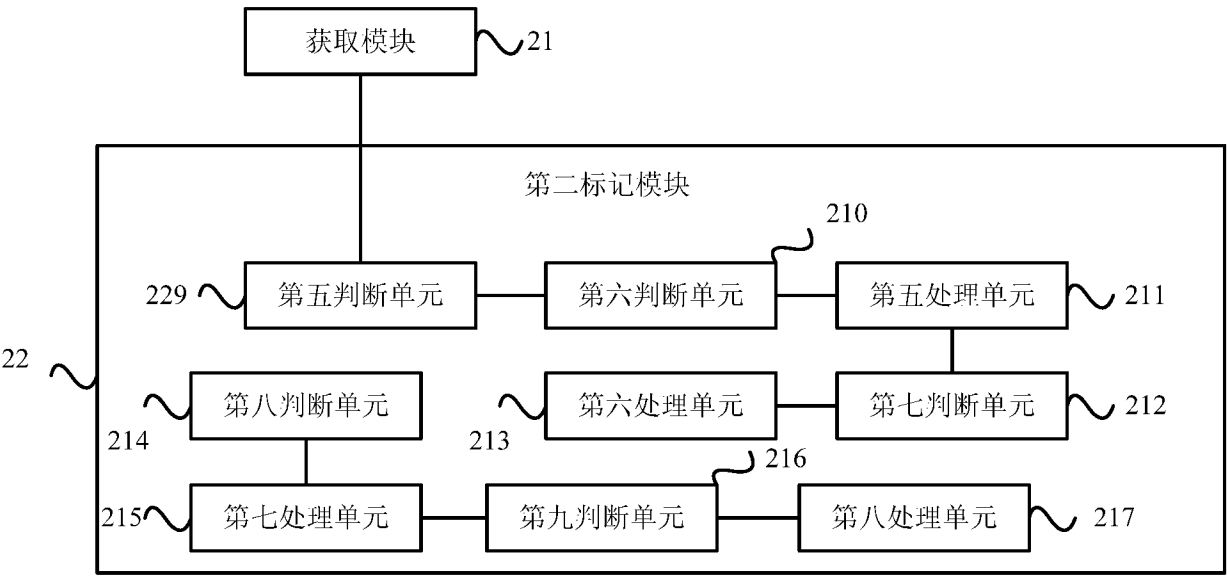


图 16

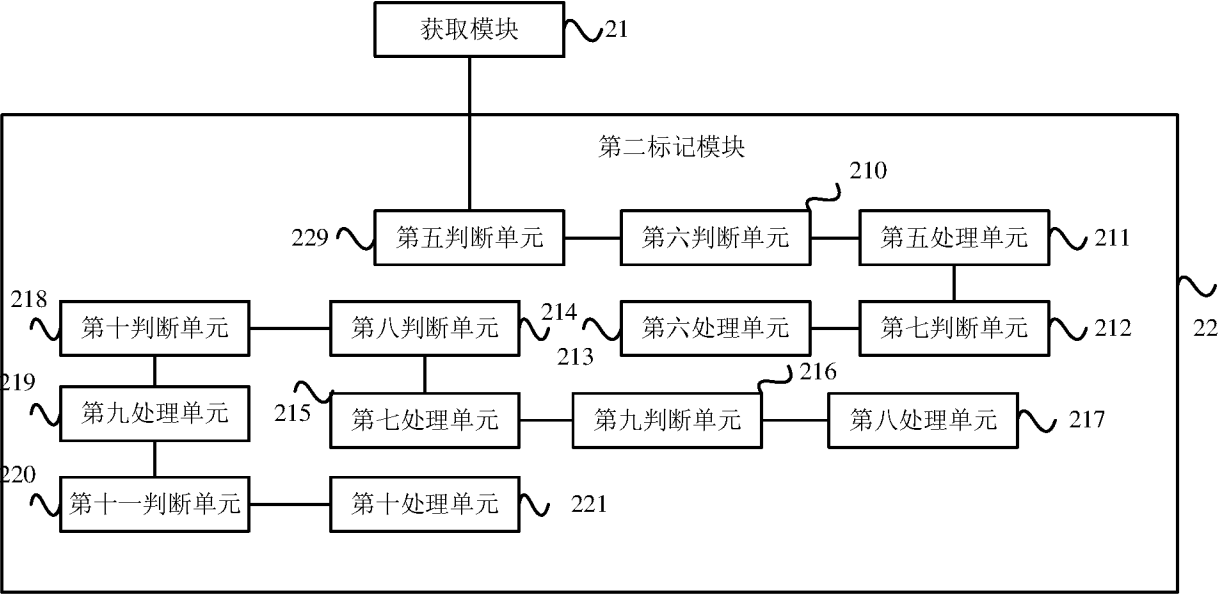


图 17

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2017/077446

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/02 (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)

EPODOC, WPI, CNPAT, CNKI, IEEE: recovery, press in, garbage, collect, parallel, mark, memory, block, thread, map, stack, synchronize, pointer, push, pop, private, common, queue

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WU, Ming et al., "Task-Pushing: a Scalable Parallel GC Marking Algorithm without Synchronization Operations", PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, 2007. IPDPS 2007. IEEE INTERNATIONAL, vol./, no./, 11 June 2007 (11.06.2007), ISSN: 1530-2075, the abstract, page 3, left-hand column, last paragraph to page 5, right-hand column, paragraph 3, and figures 1-3	1-11, 15, 25-35, 39
A	CN 102722415 A (GUANZHOU KINGRAY INFORMATION TECHNOLOGY CO., LTD.), 10 October 2012 (10.10.2012), the whole document	1-48
A	CN 102023891 A (FUDAN UNIVERSITY), 20 April 2011 (20.04.2011), the whole document	1-48
A	US 2009248766 A1 (SUN MICROSYSTEMS, INC.), 01 October 2009 (01.10.2009), the whole document	1-48
A	US 2015058381 A1 (ORACLE INTERNATIONAL CORPORATION), 26 February 2015 (26.02.2015), the whole document	1-48

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:	"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
"A" document defining the general state of the art which is not considered to be of particular relevance	"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
"E" earlier application or patent but published on or after the international filing date	"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
"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)	"&" document member of the same patent family
"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	

Date of the actual completion of the international search 07 June 2017 (07.06.2017)	Date of mailing of the international search report 28 June 2017 (28.06.2017)
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 JIANG, Xiaoqing Telephone No.: (86-10) 61648093

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2017/077446

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ZHANG, Shijun et al., "Improved Garbage Collection Algorithm for Incremental JVM", COMPUTER ENGINEERING, vol.38, no.1, 31 January 2012 (31.01.2012), ISSN: 1000-3428, the whole document	1-48

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2017/077446

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 102722415 A	10 October 2012	None	
CN 102023891 A	20 April 2011	None	
US 2009248766 A1	01 October 2009	None	
US 2015058381 A1	26 February 2015	EP 3036638 A1	29 June 2016
		WO 2015026966 A1	26 February 2015
		CN 105453054 A	30 March 2016

国际检索报告

国际申请号

PCT/CN2017/077446

A. 主题的分类

G06F 12/02 (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))

EPODOC, WPI, CNPAT, CNKI, IEEE: 垃圾, 回收, 收集, 并行, 标记, 内存, 块, 线程, 映射, 栈, 同步, 指针, 压入, 弹出, 私有, 公共, 队列, garbage, collect, parallel, mark, memory, block, thread, map, stack, synchronize, pointer, push, pop, private, common, queue

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
X	WU, Ming 等. "Task-Pushing: a Scalable Parallel GC Marking Algorithm without Synchronization Operations" Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, 第/卷, 第/期, 2007年 6月 11日 (2007 - 06 - 11), ISSN: 1530-2075, 摘要, 第3页左栏最后1段-第5页右栏第3段, 图1-3	1-11, 15, 25-35, 39
A	CN 102722415 A (广州晶锐信息技术有限公司) 2012年 10月 10日 (2012 - 10 - 10) 全文	1-48
A	CN 102023891 A (复旦大学) 2011年 4月 20日 (2011 - 04 - 20) 全文	1-48
A	US 2009248766 A1 (SUN MICROSYSTEMS, INC.) 2009年 10月 1日 (2009 - 10 - 01) 全文	1-48
A	US 2015058381 A1 (ORACLE INTERNATIONAL CORPORATION) 2015年 2月 26日 (2015 - 02 - 26) 全文	1-48

☒ 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

2017年 6月 7日

国际检索报告邮寄日期

2017年 6月 28日

ISA/CN的名称和邮寄地址

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

传真号 (86-10) 62019451

受权官员

姜晓庆

电话号码 (86-10) 61648093

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	张嗜军 等. “一种改进的增量式JVM垃圾收集算法” 计算机工程, 第38卷, 第1期, 2012年 1月 31日 (2012 - 01 - 31), ISSN: 1000-3428, 全文	1-48

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2017/077446

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	102722415	A	2012年 10月 10日	无			
CN	102023891	A	2011年 4月 20日	无			
US	2009248766	A1	2009年 10月 1日	无			
US	2015058381	A1	2015年 2月 26日	EP	3036638	A1	2016年 6月 29日
				WO	2015026966	A1	2015年 2月 26日
				CN	105453054	A	2016年 3月 30日

表 PCT/ISA/210 (同族专利附件) (2009年7月)