(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 119166320 A
(43)申请公布日 2024.12.20

(21)申请号 202310942821.6

(22)申请日 2023.07.28

(71)申请人 京东科技信息技术有限公司
地址 100176 北京市大兴区经济技术开发区科创十一街18号院2号楼6层601

(72)发明人 孟祥滨

(74)专利代理机构 北京英赛嘉华知识产权代理有限责任公司 11204
专利代理师 王达佐 马晓亚

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/48(2006.01)

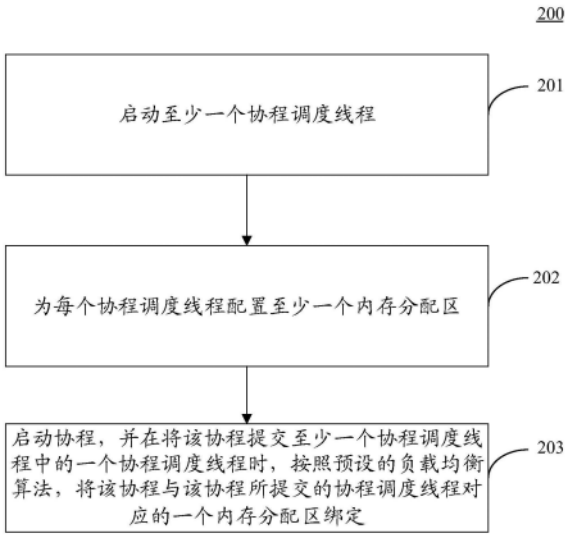
权利要求书2页 说明书10页 附图7页

(54)发明名称

基于协程的内存管理方法、装置

(57)摘要

本申请公开了基于协程的内存管理方法和装置，涉及内存管理技术领域。该方法的一具体实施方式包括：启动至少一个协程调度线程；为每个协程调度线程配置至少一个内存分配区；启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。该实施方式在有效提升系统并发性能的同时降低了内存占用率。



CN 119166320 A

1.一种基于协程的内存管理方法,所述方法包括:

启动至少一个协程调度线程;

为每个协程调度线程配置至少一个内存分配区,其中,所述至少一个内存分配区中的每一内存分配区对应一个线程本地缓存区;

启动协程,并在将该协程提交所述至少一个协程调度线程中的一个协程调度线程时,按照预设的负载均衡算法,将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定,其中,绑定同一内存分配区的协程构成协程组。

2.根据权利要求1所述的方法,所述方法还包括:

在配置允许跨线程调度协程的条件下,在基于协程申请内存请求执行内存分配操作之前,或者在基于协程释放内存请求执行内存合并及归还操作之前,标记原子变量;

响应于确定内存分配操作完成或内存合并及归还操作完成,将所述原子变量清零。

3.根据权利要求1所述的方法,所述方法还包括:

响应于获取到第一协程申请内存的第一请求,在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中搜索与所述第一请求对应的目标内存空间;

响应于搜索成功,分配所述目标内存空间;

响应于搜索失败,在该第一协程所处的协程组绑定的内存分配区中搜索与第一请求对应的目标内存空间并分配。

4.根据权利要求1所述的方法,所述方法还包括:

响应于获取到第二协程释放内存的第二请求,合并该第二协程所处的协程组绑定的内存分配区对应的线程本地缓存区中的空闲内存空间,得到第一内存空间;

响应于确定所述第一内存空间符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件,将所述第一内存空间归还内存分配区,并合并该内存分配区的空闲内存空间,得到第二内存空间;

响应于确定所述第二内存空间符合归还操作系统的第二预设条件,将所述第二内存空间归还操作系统。

5.根据权利要求4所述的方法,所述方法还包括:

响应于确定所述第一内存空间不符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件或所述第二内存空间不符合归还操作系统的第二预设条件,结束操作。

6.根据权利要求3-5任一所述的方法,所述方法还包括:

配置禁止协程跨线程调度。

7.一种基于协程的内存管理装置,所述装置包括:

启动模块,被配置成启动至少一个协程调度线程;

配置模块,被配置成为每个协程调度线程配置至少一个内存分配区,其中,所述至少一个内存分配区中的每一内存分配区对应一个线程本地缓存区;

绑定模块,被配置成启动协程,并在将该协程提交所述至少一个协程调度线程中的一个协程调度线程时,按照预设的负载均衡算法,将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定,其中,绑定同一内存分配区的协程构成协程组。

8.根据权利要求7所述的装置,所述装置还包括:

标记模块,被配置成在配置允许跨线程调度协程的条件下,在基于协程申请内存请求

执行内存分配操作之前，或者在基于协程释放内存请求执行内存合并及归还操作之前，标记原子变量；响应于确定内存分配操作完成或内存合并及归还操作完成，将所述原子变量清零。

9.一种电子设备，其特征在于，包括：

至少一个处理器；以及

与所述至少一个处理器通信连接的存储器；其中，

所述存储器存储有可被所述至少一个处理器执行，以使所述至少一个处理器能够执行权利要求1-6中任一项所述的方法。

10.一种存储有计算机指令的非瞬时计算机可读存储介质，其特征在于，所述计算机指令用于使所述计算机执行权利要求1-6中任一项所述的方法。

基于协程的内存管理方法、装置

技术领域

[0001] 本申请涉及计算机技术领域，具体涉及内存管理技术领域，尤其涉及一种基于协程的内存管理方法和装置。

背景技术

[0002] 现有技术中，为了支持并发，大多数内存分配器都采取了所谓“arena”机制，分摊多线程并发申请内存的负载。内存分配器通过将不同线程“绑定”到同一个arena，并在一个进程中开启多个arena的方式，降低锁冲突，解决并发性能问题。此外，大部分内存分配器还提供线程本地缓存机制。

[0003] 但是，当应用程序的线程数量达到一定规模，或者并发任务中的内存分配/释放(即，malloc/free库函数调用)过于频繁时，这种锁冲突带来的性能损失仍然不可小觑。并且，为了优化性能，大多数内存分配器都引入了线程本地缓存功能；当系统中存在大量线程，且其各自拥有独立的本地缓存时，整体内存占用率的增加也非常明显。

发明内容

[0004] 本申请实施例提供了一种基于协程的内存管理方法、装置、设备以及存储介质。

[0005] 根据第一方面，本申请实施例提供了一种基于协程的内存管理方法，该方法包括：启动至少一个协程调度线程；为每个协程调度线程配置至少一个内存分配区；启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0006] 根据第二方面，本申请实施例提供了一种基于协程的内存管理装置，该装置包括：启动模块，被配置成启动至少一个协程调度线程；配置模块，被配置成为每个协程调度线程配置至少一个内存分配区；绑定模块，被配置成启动协程，并在将该协程提交所述至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0007] 根据第三方面，本申请实施例提供了一种电子设备，该电子设备包括一个或多个处理器；存储装置，其上存储有一个或多个程序，当一个或多个程序被该一个或多个处理器执行，使得一个或多个处理器实现如第一方面的任一实施例的基于协程的内存管理方法。

[0008] 根据第四方面，本申请实施例提供了一种计算机可读介质，其上存储有计算机程序，该程序被处理器执行时实现如第一方面的任一实施例的基于协程的内存管理方法。

[0009] 本申请通过启动至少一个协程调度线程；为每个协程调度线程配置至少一个内存分配区；启动协程，并在将该协程提交所述至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定，充分利用了协程多任务调度分时复用协程调度线程时间片的特性，同属于一个协程调度线程的协程之间共享内存分配区(arena)时，内存申请/释放时无需加锁，从而将因内存管理而产生锁冲突的可能性降至最低，同时，多个协程共用一个线程本地缓存区，

即tcache，上一个协程释放到tcache的内存，调度下一个协程的时候如果要分配内存，就可以被复用，相较于现有方案中每个线程有自己独立的线程本地缓存区，这个tcache不能与其他线程共享，有效降低了系统整体内存占用率，提升了系统的并发性能。

[0010] 应当理解，本部分所描述的内容并非旨在标识本公开的实施例的关键或重要特征，也不用于限制本公开的范围。本公开的其他特征将通过以下的说明书而变得容易理解。

附图说明

[0011] 图1是本申请可以应用于其中的示例性系统架构图；

[0012] 图2是根据本申请的基于协程的内存管理方法的一个实施例的流程图；

[0013] 图3是根据本申请的基于协程的内存管理方法的一个实施例的架构图；

[0014] 图4a是根据本申请的基于协程的内存管理方法的一个实施例的流程图；

[0015] 图4b是根据本申请的基于协程的内存管理方法的一个应用场景的流程图；

[0016] 图4c是根据本申请的基于协程的内存管理方法的又一个应用场景的流程图；

[0017] 图5是根据本申请的基于协程的内存管理装置的一个实施例的流程图；

[0018] 图6是适于用来实现本申请实施例的服务器的计算机系统的结构示意图。

具体实施方式

[0019] 以下结合附图对本申请的示范性实施例做出说明，其中包括本申请实施例的各种细节以助于理解，应当将它们认为仅仅是示范性的。因此，本领域普通技术人员应当认识到，可以对这里描述的实施例做出各种改变和修改，而不会背离本申请的范围和精神。同样，为了清楚和简明，以下的描述中省略了对公知功能和结构的描述。

[0020] 需要说明的是，在不冲突的情况下，本申请中的实施例及实施例中的特征可以相互组合。下面将参考附图并结合实施例来详细说明本申请。

[0021] 图1示出了可以应用本申请的基于协程的内存管理方法的实施例的示例性系统架构100。

[0022] 如图1所示，系统架构100可以包括终端设备101、102、103，网络104和服务器105。网络104用以在终端设备101、102、103和服务器105之间提供通信链路的介质。网络104可以包括各种连接类型，例如有线、无线通信链路或者光纤电缆等等。

[0023] 终端设备101、102、103通过网络104与服务器105交互，以接收或发送消息等。终端设备101、102、103上可以安装有各种通讯客户端应用，例如，存储类应用、通讯类应用等。

[0024] 终端设备101、102、103可以是硬件。当终端设备101、102、103为硬件时，可以是具有显示屏的各种电子设备，包括但不限于手机和笔记本电脑。

[0025] 服务器105可以包括提供以下功能的内存分配器，例如，启动至少一个协程调度线程；为每个协程调度线程配置至少一个内存分配区；启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0026] 需要说明的是，当服务器105为硬件时，可以实现成多个服务器组成的分布式服务器集群，也可以实现成单个服务器。

[0027] 需要指出的是，本公开的实施例所提供的基于协程的内存管理方法可以由服务器

105中的内存分配器执行，也可以由终端设备101、102、103中的内存分配器执行。相应地，基于协程的内存管理装置包括的各个部分(例如各个单元、子单元、模块、子模块)可以全部设置于服务器105中，也可以全部设置于终端设备101、102、103中。

[0028] 应该理解，图1中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要，可以具有任意数目的终端设备、网络和服务器。

[0029] 图2示出了可以应用于本申请的基于协程的内存管理方法的实施例的流程200。在本实施例中，基于协程的内存管理方法包括以下步骤：

[0030] 步骤201，启动至少一个协程调度线程。

[0031] 在本实施例中，当前操作系统都会提供用于内存申请/释放的系统调用，用于满足应用程序对内存使用的要求。以Linux为例，其提供sbrk/brk系统调用，用于在堆区分配内存；提供mmap系统调用在映射区分配内存(一般sbrk/brk用于分配小尺寸内存，而mmap常用于分配大块内存；除Linux以外，其他操作系统也都有类似的系统调用)。

[0032] 但是，如果每次分配/释放内存都使用系统调用，应用程序的性能将受到极大的影响；因而，各类不同的操作系统发行版，都会提供用户态内存分配器，以提升性能。例如，GNU Linux的默认分配器ptmalloc，FreeBSD的jemalloc，以及Google贡献的开源内存分配器tcmalloc等等。

[0033] 为了支持并发，大多数内存分配器都采取了所谓“arena”机制，分摊多线程并发申请内存的负载。一个“arena”就是一个内存分配区。虽然，现有内存分配器都采用多内存分配区(arena)机制，以降低并发冲突；但是，当应用程序的线程数量达到一定规模，或者并发任务中的内存分配/释放(即，malloc/free库函数调用)过于频繁时，这种锁冲突带来的性能损失仍然非常巨大。

[0034] 为克服上述问题，执行主体(如图1中所示的服务器105或终端设备101、102、103中的内存分配器)首先启动至少一个，例如，5个、10个等，协程调度线程

[0035] 其中，协程调度线程用于调度至少一个协程，协程的运行依附于其归属的协程调度线程，协程通过yield方式让出后，协程调度线程才能调度其他协程运行，即协程之间分时复用协程调度线程的时间片。因而，同一个协程调度线程上的协程虽然名义上是“并行”执行，但其分时复用机制保证了这些协程中的任意两个之间，实际上都不会同时运行(即运行时间片不会产生重叠)。基于这个前提，只要保证协程对缓存的访问具备“原子性”(即不会在对缓存的操作未完成时进行yield操作)，就可以实现协程间共享缓存。

[0036] 步骤202，为每个协程调度线程配置至少一个内存分配区。

[0037] 在本实施例中，执行主体可为每个协程调度线程配置一个或多个内存分配区。

[0038] 其中，至少一个内存分配区中的每一内存分配区对应一个线程本地缓存区，各内存分配区对应的线程本地缓存区各不相同。

[0039] 这里，线程本地缓存区，即tcache是隶属于内存分配区，即arena的，他们是一一对应关系。也就是说，tcache中的缓存，是arena中内存的一部分。一般来说，程序分配的内存尺度都比较小，tcache就可以认为是一个小尺度的缓存，大部分情况下，内存分配请求都可以由tcache满足。当申请大尺寸的内存空间时，才有必要向arena申请；此外，当tcache中没有适合尺寸的内存空间时，也是先从arena申请一大块，再裁剪成小块内存使用。因而，tcache就是在一个arena中，对小尺度内存分配和复用的一种加速机制。

[0040] 步骤203，启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0041] 在本实施例中，执行主体可启动一个或多个协程，对于每一协程，在将该协程提交至少一个协程调度线程中的目标协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0042] 在内存分配区上绑定的协程数大于1时，自然产生一组协程，即“协程组”。

[0043] 在上述绑定关系确定后，可根据协程的申请内存、释放内存的请求进行内存分配及内存合并归还。

[0044] 此外，需要指出的是，为避免存在目标协程被从其所属协程调度线程以外的其他协程调度线程执行，并且目标协程访问该其他协程调度线程绑定的内存分配区，产生并发访问，执行主体可以配置禁止协程的跨线程调度，也可以实现乐观锁，例如，版本号机制、CAS算法等，以解决跨线程访问内存分配区。

[0045] 进一步地，在一些可选的方式中，本申请的上述步骤201、202、203可以经由经由指定协程库中的库函数，如，malloc、free、calloc、realloc、memalign、valloc等，钩子提供的回调接口执行。该方式可实现“透明”替换内存分配器，也即在应用无感知的情况下替换内存分配器，提升内存管理性能，同时在协程库提供的高性能并发调度能力的基础上，进一步提升了整体的并发吞吐能力。

[0046] 对于没有callback接口的协程库，例如，libco、libgo等，可增加对基于协程的内存管理操作相关库函数的override，并基于步骤201、202、203实现基于协程的内存管理器。

[0047] 在一些可选的方式中，该方法还包括：响应于获取到第一协程申请内存的第一请求，在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中搜索与第一请求对应的目标内存空间；响应于搜索成功，分配目标内存空间；响应于搜索失败，在该第一协程所处的协程组绑定的内存分配区中搜索与第一请求对应的目标内存空间并分配。

[0048] 在本实现方式中，由于当应用程序调用库函数free释放内存空间时，释放的内存不直接释放回内存分配区，而是放入内存分配区对应的线程本地缓存区，响应于获取到第一协程申请内存的第一请求，执行主体可首先在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中搜索与第一请求对应的目标内存空间，即适合本次请求尺寸的内存空间，若搜索成功，在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中分配目标内存空间；若搜索失败，则进一步在该第一协程所处的协程组绑定的内存分配区中搜索与第一请求对应的目标内存空间并分配。

[0049] 该实现方式通过响应于获取到第一协程申请内存的第一请求，在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中搜索与第一请求对应的目标内存空间；响应于搜索成功，分配目标内存空间；响应于搜索失败，在该第一协程所处的协程组绑定的内存分配区中搜索与第一请求对应的目标内存空间并分配，实现了针对协程申请内存的请求的内存分配。

[0050] 在一些可选的方式中，该方法还包括：响应于获取到第二协程释放内存的第二请求，合并该第二协程所处的协程组绑定的内存分配区对应的线程本地缓存区中的空闲内存空间，得到第一内存空间；响应于确定第一内存空间符合归还该第二协程所处的协程组绑

定的内存分配区的第一预设条件，将第一内存空间归还该内存分配区，并合并该内存分配区的空闲内存空间，得到第二内存空间；响应于确定第二内存空间符合归还操作系统的第二预设条件，将第二内存空间归还操作系统。

[0051] 在本实现方式中，响应于获取到第二协程释放内存的第二请求，执行主体可首先合并该第二协程所处的协程组绑定的内存分配区对应的线程本地缓存区中的空闲内存空间，得到第一内存空间，并判断第一内存空间是否符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件；响应于确定第一内存空间符合归还内存分配区的第一预设条件，将第一内存空间归还该内存分配区，并合并该内存分配区的空闲内存空间，得到第二内存空间；响应于确定第二内存空间符合归还操作系统的第二预设条件，将第二内存空间归还操作系统，响应于不符合，则结束归还操作。

[0052] 这里，第一预设条件、第二预设条件可以相同也可以不同，第一预设条件、第二预设条件可以根据经验、实际需求设定，例如，地址连续且起始地址对齐的连续内存尺寸大于等于预设尺寸阈值、存储区(缓存区或内存分配区)中停留时间大于等于预设时间阈值等，本申请对此不作限定。

[0053] 该实现方式通过响应于获取到第二协程释放内存的第二请求，合并该第二协程所处的协程组绑定的内存分配区对应的线程本地缓存区中的空闲内存空间，得到第一内存空间；响应于确定第一内存空间符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件，将第一内存空间归还该内存分配区，并合并该内存分配区的空闲内存空间，得到第二内存空间；响应于确定第二内存空间符合归还操作系统的第二预设条件，将第二内存空间归还操作系统，实现了针对协程释放内存请求的内存空间归还。

[0054] 在一些可选的方式中，该方法还包括：响应于确定第一内存空间不符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件或第二内存空间不符合归还操作系统的第二预设条件，结束操作。

[0055] 在本实现方式中，执行主体可判断第一内存空间是否符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件，响应于第一内存空间不符合第一预设条件，或第二内存空间不符合归还操作系统的第二预设条件，则结束操作。

[0056] 该实现方式通过响应于确定第一内存空间不符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件，或所述第二内存空间不符合归还操作系统的第二预设条件，结束操作，实现了针对协程释放内存请求的内存空间归还。

[0057] 在一些可选的方式中，配置禁止协程跨线程调度。

[0058] 在本实现方式中，为避免存在目标协程被从其所属协程调度线程以外的其他协程调度线程执行，并且目标协程访问该其他协程调度线程绑定的内存分配区，产生并发访问，执行主体可配置禁止协程跨线程调度。

[0059] 该实现方式通过配置禁止协程跨线程调度，可有效避免产生并发访问，保障内存分配/回收性能。

[0060] 继续参见图3，图3是根据本实施例的基于协程的内存管理方法的应用场景的一个架构图。

[0061] 在图3的应用场景中，执行主体可启动至少一个协程调度线程；为每个协程调度线程配置至少一个内存分配区，如arena0、arena1、arena2……arenaN，其中，至少一个内存分

配区中的每一内存分配区对应一个线程本地缓存区;启动协程,并在将该协程提交至少一个协程调度线程中的协程调度线程时,按照预设的负载均衡算法,将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定,其中,绑定同一内存分配区的协程构成协程组,例如,协程组0绑定arena0,tcache0对应(隶属于)arena0;协程组1绑定arena1,tcache1对应(隶属于)arena1……协程组N绑定arenaN,tcacheN对应(隶属于)arenaN。

[0062] 图4a示出了可以应用于本申请的基于协程的内存管理方法的另一个实施例的流程400。在本实施例中,基于协程的内存管理方法包括以下步骤:

[0063] 步骤401,启动至少一个协程调度线程。

[0064] 在本实施例中,步骤401的实现细节和技术效果,可以参考对步骤201的描述,在此不再赘述。

[0065] 步骤402,为每个协程调度线程配置至少一个内存分配区。

[0066] 在本实施例中,步骤402的实现细节和技术效果,可以参考对步骤202的描述,在此不再赘述。

[0067] 步骤403,启动协程,并在将该协程提交至少一个协程调度线程中的一个协程调度线程时,按照预设的负载均衡算法,将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0068] 在本实施例中,步骤403的实现细节和技术效果,可以参考对步骤203的描述,在此不再赘述。

[0069] 步骤404,在配置允许跨线程调度协程的条件下,在基于协程申请内存请求执行内存分配操作之前,或者在基于协程释放内存请求执行内存合并及归还操作之前,标记原子变量。

[0070] 在本实施例中,为避免存在目标协程被从其所属协程调度线程以外的其他协程调度线程执行,并且目标协程访问该其他协程调度线程绑定的内存分配区,产生并发访问,在配置允许跨线程调度协程的条件下,执行主体可基于CAS(Compare And Swap,比较交换)操作配置一个原子变量。在基于协程申请内存请求执行内存分配操作之前,或者在基于协程释放内存请求执行内存合并及归还操作之前,,标记原子变量。

[0071] 其中,CAS是支持并发的处理器(CPU)提供的一种硬件同步原语。CAS操作包含三个操作数,即内存位置(V)、预期原值(A)和新值(B),写作CAS(V,A,B)。如果内存位置的值与预期原值相匹配,那么处理器会自动将该位置值更新为新值。否则,处理器不做任何操作。

[0072] 步骤405,响应于确定内存分配操作完成或内存合并及归还操作完成,将原子变量清零。

[0073] 在本实施例中,执行主体响应于确定内存分配操作完成或内存合并及归还操作完成,将原子变量清零。

[0074] 具体地,如图4b所示,执行主体可基于CAS操作配置原子变量,在执行内存分配操作之前,标记原子变量,即设置原子变量,内存分配操作可以包括:在线程本地缓存区,即tcache,中搜索与第一请求对应的内存空间,即适合本次请求尺寸的内存空间,若搜索成功,则直接对第一请求对应的目标内存空间进行分配;若搜索失败,则进一步在内存分配区中搜索与第一请求对应的目标内存空间,若搜索成功,直接对第一请求对应的目标内存空间进行分配。

[0075] 响应于内存分配操作完成,将原子变量清零。

[0076] 又如图4c所示,执行主体可基于CAS操作配置原子变量,在进行内存合并及归还操作前标记原子变量,内存合并及归还操作可以包括:合并线程本地缓存区中空闲的内存空间,得到第一内存空间,并判断第一内存空间是否符合归还内存分配区的第一预设条件;响应于第一内存空间不符合第一预设条件则结束归合并还操作。响应于第一内存空间符合归还内存分配区的第一预设条件,将第一内存空间归还内存分配区,并合并内存分配区的空闲内存空间,得到第二内存空间;响应于第二内存空间符合归还操作系统的第二预设条件,将第二内存空间归还操作系统;响应于不符合,则结束操作。

[0077] 响应于确定内存合并及归还操作完成,将原子变量清零。

[0078] 进一步地,若标记原子变量失败,且失败次数大于等于预设的次数阈值,不再做busy loop,而是通过挂起协程的方式进行延迟,换出一次再换入,然后重新尝试CAS操作。也就是“yield&resume”,即先换出,然后再次被换入时继续后面的操作。

[0079] 从图4中可以看出,与图2对应的实施例相比,本实施例中的基于协程的内存管理方法的流程400体现了在配置允许跨线程调度协程的条件下,在基于协程申请内存请求执行内存分配操作之前,或者在基于协程释放内存请求执行内存合并及归还操作之前,标记原子变量;响应于确定内存分配操作完成或内存合并及归还操作完成,将原子变量清零(即支持负载均衡,如当某些协程调度线程上大量协程结束造成负载不均时,可以把其他负载较重的协程调度线程上的协程调度至相对负载较轻的调度线程)的同时避免了并发访问。

[0080] 进一步参考图5,作为对上述各图所示方法的实现,本申请提供了一种基于协程的内存管理装置的一个实施例,该装置实施例与图2所示的方法实施例相对应,该装置具体可以应用于各种电子设备中。

[0081] 如图5所示,本实施例的基于协程的内存管理装置500包括:启动模块501、配置模块502和绑定模块503。

[0082] 其中,启动模块501,可被配置成启动至少一个协程调度线程。

[0083] 配置模块502,可被配置成为每个协程调度线程配置至少一个内存分配区。

[0084] 绑定模块503,可被配置成启动协程,并在将该协程提交至少一个协程调度线程中的一个协程调度线程时,按照预设的负载均衡算法,将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定。

[0085] 在本实施例的一些可选的方式中,该装置还包括标记模块,被配置成在配置允许跨线程调度协程的条件下,在基于协程申请内存请求执行内存分配操作之前,或者在基于协程释放内存请求执行内存合并及归还操作之前,标记原子变量;响应于确定内存分配完成或内存合并及归还完成,将原子变量清零。

[0086] 在本实施例的一些可选的方式中,该装置还包括分配模块,被配置成响应于获取到第一协程申请内存的第一请求,在该第一协程所处的协程组绑定的内存分配区对应的线程本地缓存区中搜索与第一请求对应的目标内存空间;响应于搜索成功,分配目标内存空间;响应于搜索失败,在该第一协程所处的协程组绑定的内存分配区中搜索与第一请求对应的目标内存空间并分配。

[0087] 在本实施例的一些可选的方式中,该装置还包括释放模块,被配置成响应于获取到第二协程释放内存的第二请求,合并该第二协程所处的协程组绑定的内存分配区对应的

线程本地缓存区中的空闲内存空间，得到第一内存空间；响应于确定第一内存空间符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件，将第一内存空间归还该内存分配区，并合并该内存分配区的空闲内存空间，得到第二内存空间；响应于确定第二内存空间符合归还操作系统的第二预设条件，将第二内存空间归还操作系统。

[0088] 在本实施例的一些可选的方式中，该装置还包括归还模块，被配置成响应于确定第一内存空间不符合归还该第二协程所处的协程组绑定的内存分配区的第一预设条件，或第二内存空间不符合归还操作系统的第二预设条件，结束操作。

[0089] 在本实施例的一些可选的方式中，该装置还包括：配置模块，被配置成配置禁止协程跨线程调度。

[0090] 根据本申请的实施例，本申请还提供了一种电子设备和一种可读存储介质。

[0091] 如图6所示，是根据本申请实施例的基于协程的内存管理方法的电子设备的框图。

[0092] 600是根据本申请实施例的基于协程的内存管理方法的电子设备的框图。电子设备旨在表示各种形式的数字计算机，诸如，膝上型计算机、台式计算机、工作台、个人数字助理、服务器、刀片式服务器、大型计算机、和其它适合的计算机。电子设备还可以表示各种形式的移动装置，诸如，个人数字处理、蜂窝电话、智能电话、可穿戴设备和其它类似的计算装置。本文所示的部件、它们的连接和关系、以及它们的功能仅仅作为示例，并且不意在限制本文中描述的和/或者要求的本申请的实现。

[0093] 如图6所示，该电子设备包括：一个或多个处理器601、存储器602，以及用于连接各部件的接口，包括高速接口和低速接口。各个部件利用不同的总线互相连接，并且可以被安装在公共主板上或者根据需要以其它方式安装。处理器可以对在电子设备内执行的指令进行处理，包括存储在存储器中或者存储器上以在外部输入/输出装置(诸如，耦合至接口的显示设备)上显示GUI的图形信息的指令。在其它实施方式中，若需要，可以将多个处理器和/或多条总线与多个存储器和多个存储器一起使用。同样，可以连接多个电子设备，各个设备提供部分必要的操作(例如，作为服务器阵列、一组刀片式服务器、或者多处理器系统)。图6中以一个处理器601为例。

[0094] 存储器602即为本申请所提供的非瞬时计算机可读存储介质。其中，所述存储器存储有可由至少一个处理器执行的指令，以使所述至少一个处理器执行本申请所提供的基于协程的内存管理方法。本申请的非瞬时计算机可读存储介质存储计算机指令，该计算机指令用于使计算机执行本申请所提供的基于协程的内存管理方法。

[0095] 存储器602作为一种非瞬时计算机可读存储介质，可用于存储非瞬时软件程序、非瞬时计算机可执行程序以及模块，如本申请实施例中的基于协程的内存管理方法对应的程序指令/模块(例如，附图5所示的启动模块501、配置模块502和绑定模块503)。处理器601通过运行存储在存储器602中的非瞬时软件程序、指令以及模块，从而执行服务器的各种功能应用以及数据处理，即实现上述方法实施例中的基于协程的内存管理方法。

[0096] 存储器602可以包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需要的应用程序；存储数据区可存储基于协程的内存管理的电子设备的使用所创建的数据等。此外，存储器602可以包括高速随机存取存储器，还可以包括非瞬时存储器，例如至少一个磁盘存储器件、闪存器件、或其他非瞬时固态存储器件。在一些实施例中，存储器602可选包括相对于处理器601远程设置的存储器，这些远程存储器可以通

过网络连接至基于协程的内存管理的电子设备。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0097] 基于协程的内存管理方法的电子设备还可以包括:输入装置603和输出装置604。处理器601、存储器602、输入装置603和输出装置604可以通过总线或者其他方式连接,图6中以通过总线连接为例。

[0098] 输入装置603可接收输入的数字或字符信息,例如触摸屏、小键盘、鼠标、轨迹板、触摸板、指示杆、一个或者多个鼠标按钮、轨迹球、操纵杆等输入装置。输出装置604可以包括显示设备、辅助照明装置(例如,LED)和触觉反馈装置(例如,振动电机)等。该显示设备可以包括但不限于,液晶显示器(LCD)、发光二极管(LED)显示器和等离子体显示器。在一些实施方式中,显示设备可以是触摸屏。

[0099] 此处描述的系统和技术的各种实施方式可以在数字电子电路系统、集成电路系统、专用ASIC(专用集成电路)、计算机硬件、固件、软件、和/或它们的组合中实现。这些各种实施方式可以包括:实施在一个或者多个计算机程序中,该一个或者多个计算机程序可在包括至少一个可编程处理器的可编程系统上执行和/或解释,该可编程处理器可以是专用或者通用可编程处理器,可以从存储系统、至少一个输入装置、和至少一个输出装置接收数据和指令,并且将数据和指令传输至该存储系统、该至少一个输入装置、和该至少一个输出装置。

[0100] 这些计算程序(也称作程序、软件、软件应用、或者代码)包括可编程处理器的机器指令,并且可以利用高级过程和/或面向对象的编程语言、和/或汇编/机器语言来实施这些计算程序。如本文使用的,术语“机器可读介质”和“计算机可读介质”指的是用于将机器指令和/或数据提供给可编程处理器的任何计算机程序产品、设备、和/或装置(例如,磁盘、光盘、存储器、可编程逻辑装置(PLD)),包括,接收作为机器可读信号的机器指令的机器可读介质。术语“机器可读信号”指的是用于将机器指令和/或数据提供给可编程处理器的任何信号。

[0101] 为了提供与用户的交互,可以在计算机上实施此处描述的系统和技术,该计算机具有:用于向用户显示信息的显示装置(例如,CRT(阴极射线管)或者LCD(液晶显示器)监视器);以及键盘和指向装置(例如,鼠标或者轨迹球),用户可以通过该键盘和该指向装置来将输入提供给计算机。其它种类的装置还可以用于提供与用户的交互;例如,提供给用户的反馈可以是任何形式的传感反馈(例如,视觉反馈、听觉反馈、或者触觉反馈);并且可以用任何形式(包括声输入、语音输入或者、触觉输入)来接收来自用户的输入。

[0102] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如,作为数据服务器)、或者包括中间件部件的计算系统(例如,应用服务器)、或者包括前端部件的计算系统(例如,具有图形用户界面或者网络浏览器的用户计算机,用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术的实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如,通信网络)来将系统的部件相互连接。通信网络的示例包括:局域网(LAN)、广域网(WAN)和互联网。

[0103] 计算机系统可以包括客户端和服务器。客户端和服务器一般远离彼此并且通常通过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计

算机程序来产生客户端和服务器的关系。

[0104] 根据本申请实施例的技术方案，在有效提升系统并发性能的同时降低了内存占用率。

[0105] 应该理解，可以使用上面所示的各种形式的流程，重新排序、增加或删除步骤。例如，本发申请中记载的各步骤可以并行地执行也可以顺序地执行也可以不同的次序执行，只要能够实现本申请公开的技术方案所期望的结果，本文在此不进行限制。

[0106] 上述具体实施方式，并不构成对本申请保护范围的限制。本领域技术人员应该明白的是，根据设计要求和其他因素，可以进行各种修改、组合、子组合和替代。任何在本申请的精神和原则之内所作的修改、等同替换和改进等，均应包含在本申请保护范围之内。

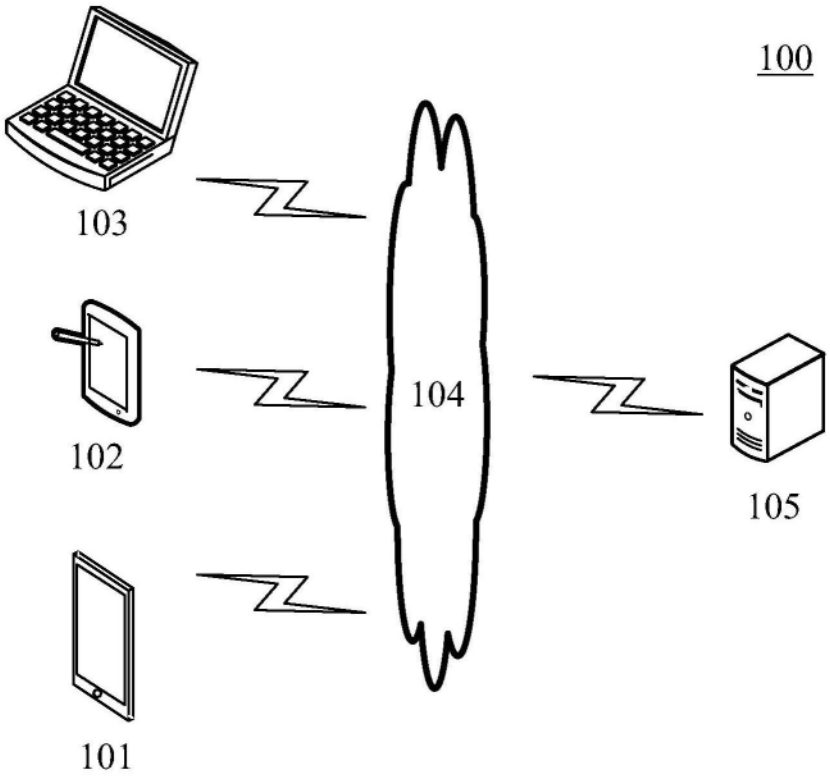


图1

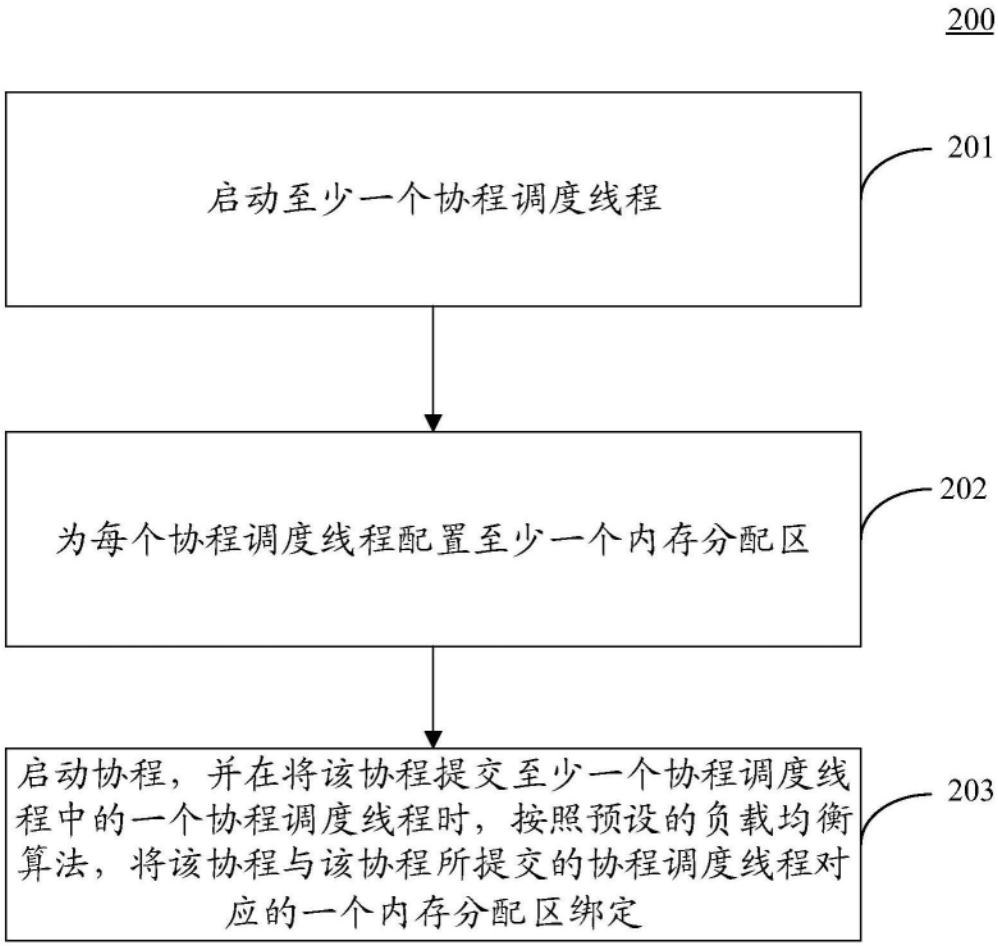

200
启动至少一个协程调度线程
201
为每个协程调度线程配置至少一个内存分配区
202
启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定
203

图2

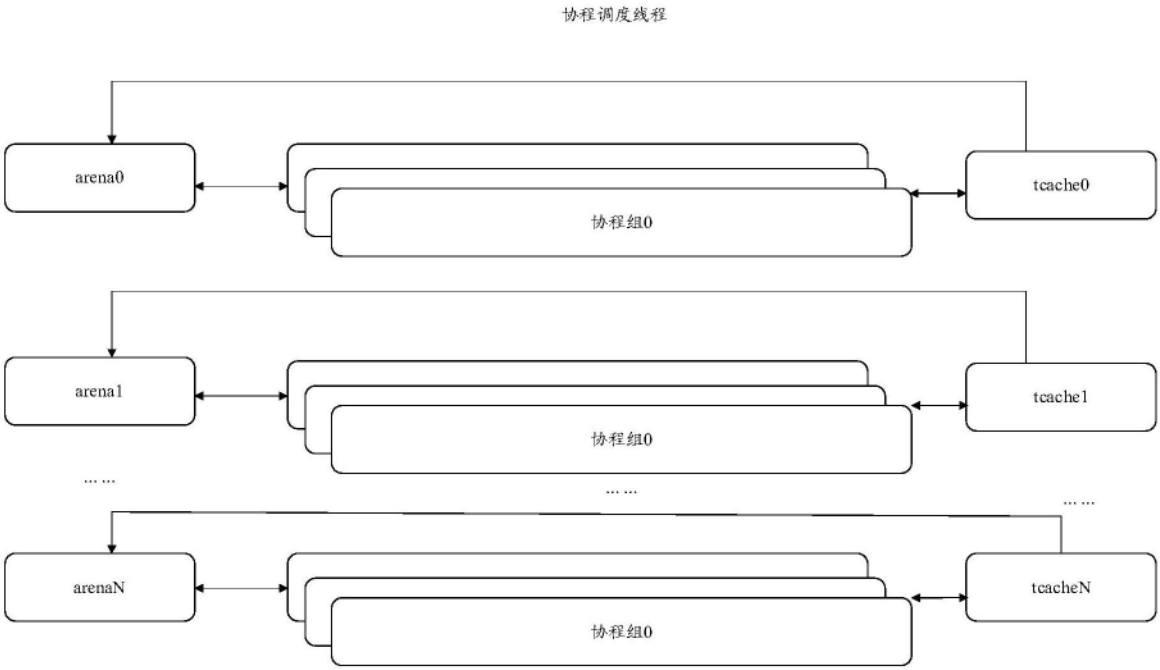

协程调度线程
arena0
协程组0
tcache0
arena1
协程组0
tcache1
……
……
……
arenaN
协程组0
tcacheN

图3

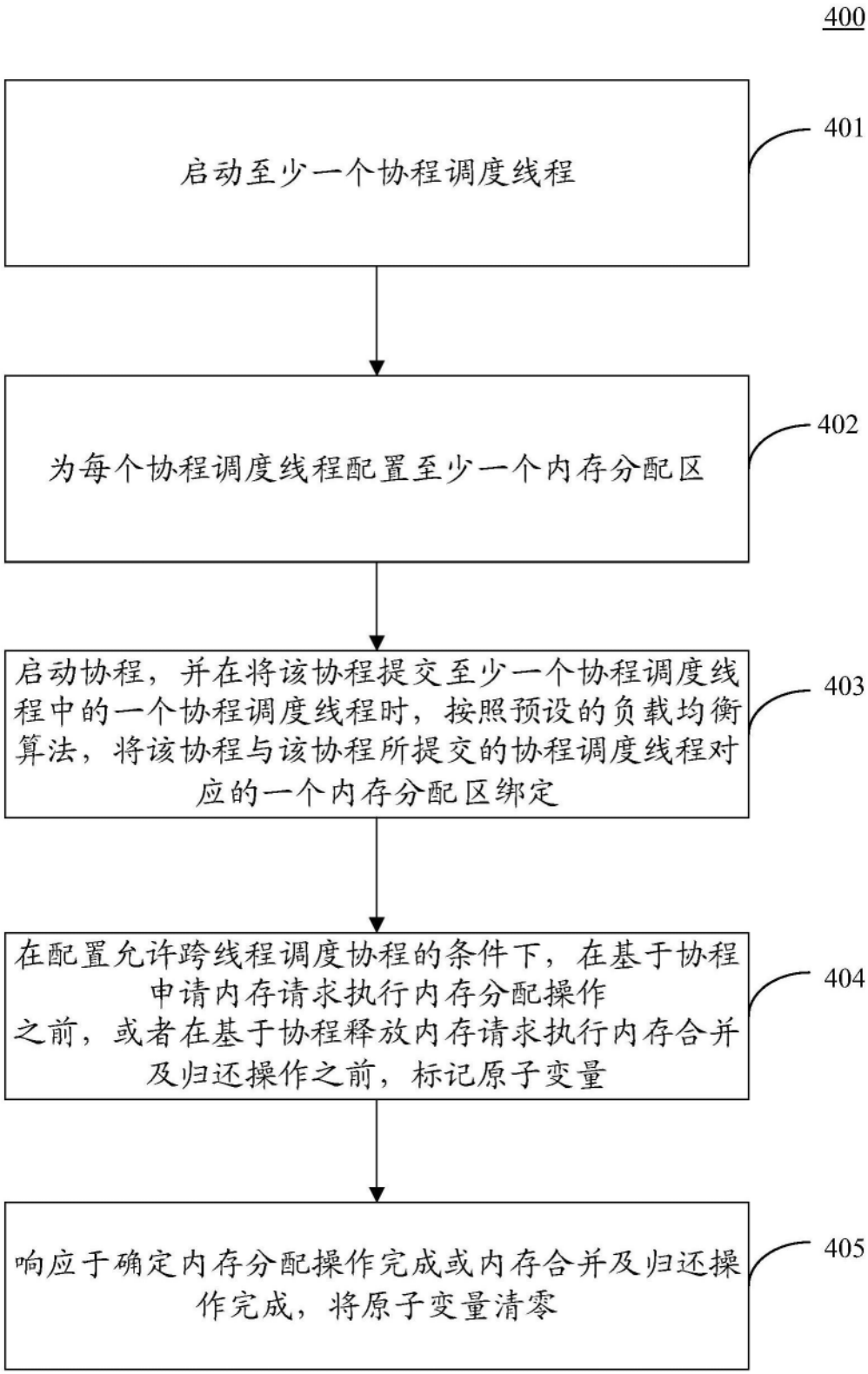

400
启动至少一个协程调度线程
401
为每个协程调度线程配置至少一个内存分配区
402
启动协程，并在将该协程提交至少一个协程调度线程中的一个协程调度线程时，按照预设的负载均衡算法，将该协程与该协程所提交的协程调度线程对应的一个内存分配区绑定
403
在配置允许跨线程调度协程的条件下，在基于协程申请内存请求执行内存分配操作之前，或者在基于协程释放内存请求执行内存合并及归还操作之前，标记原子变量
404
响应于确定内存分配操作完成或内存合并及归还操作完成，将原子变量清零
405

图4a

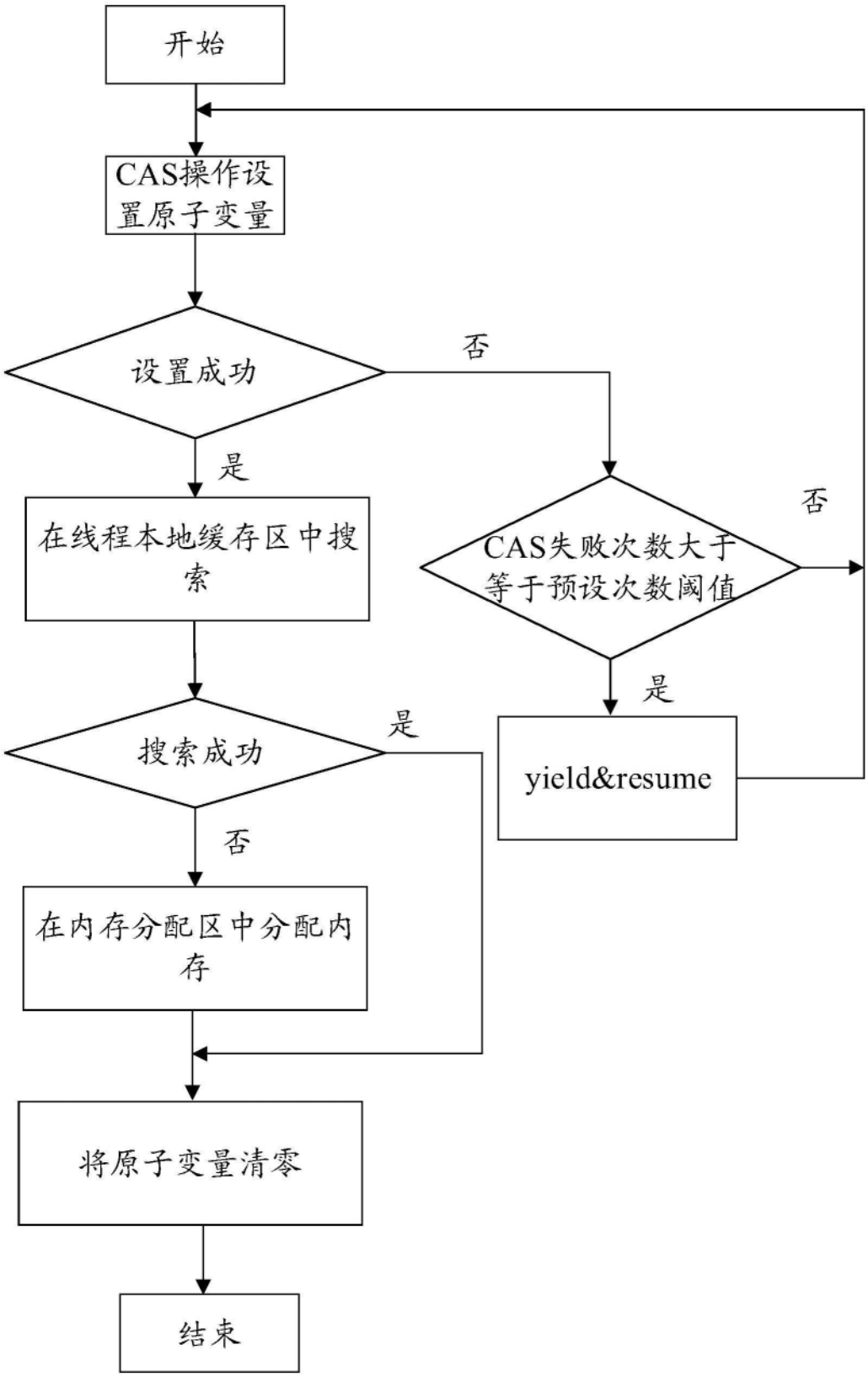

开始
CAS操作设置原子变量
设置成功
否
是
在线程本地缓存区中搜索
CAS失败次数大于等于预设次数阈值
否
是
搜索成功
是
yield&resume
否
在内存分配区中分配内存
将原子变量清零
结束

图4b

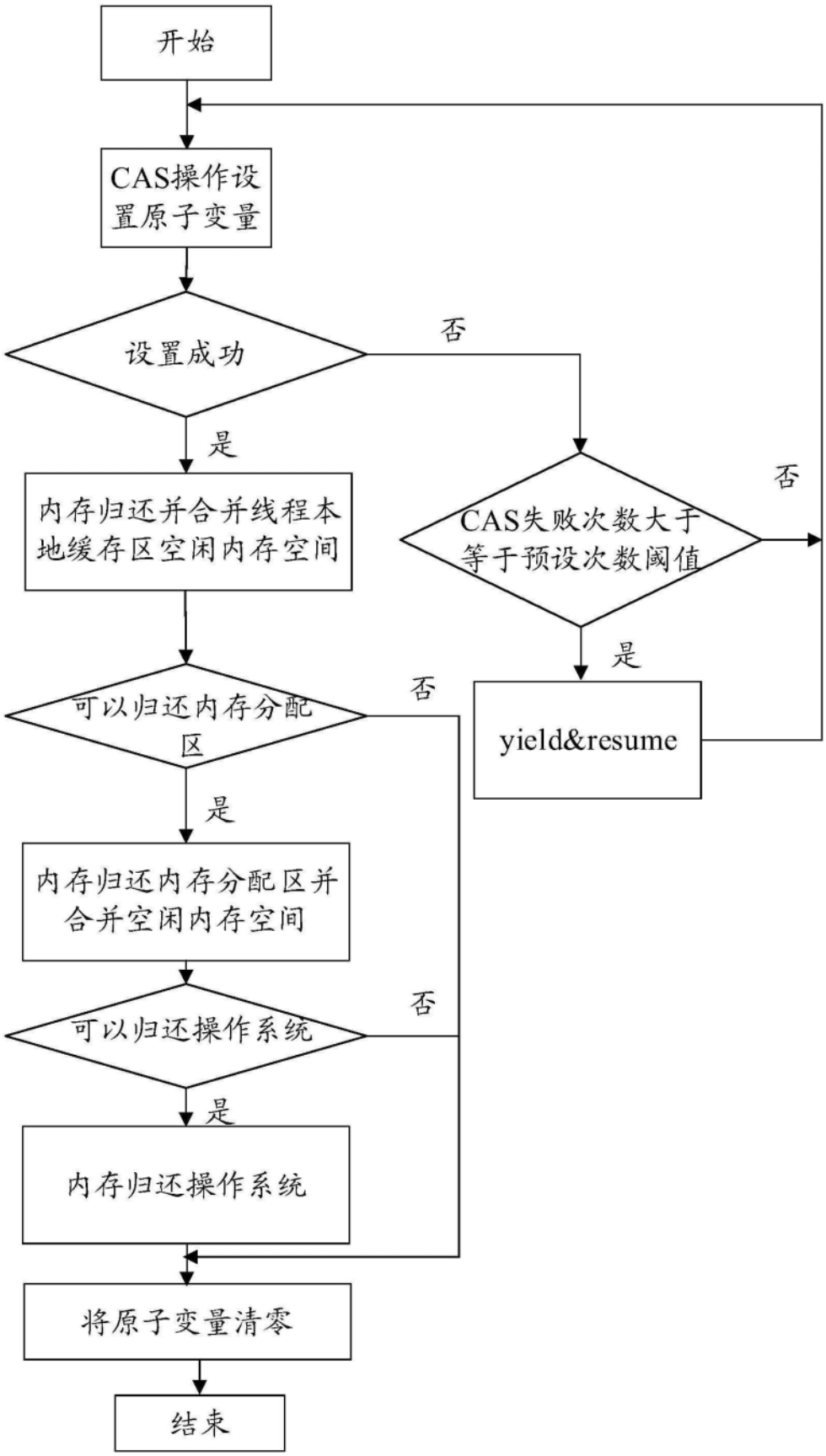

开始
CAS操作设置原子变量
设置成功
否
是
内存归还并合并线程本地缓存区空闲内存空间
CAS失败次数大于等于预设次数阈值
否
是
yield&resume
可以归还内存分配区
否
是
内存归还内存分配区并合并空闲内存空间
可以归还操作系统
否
是
内存归还操作系统
将原子变量清零
结束

图4c

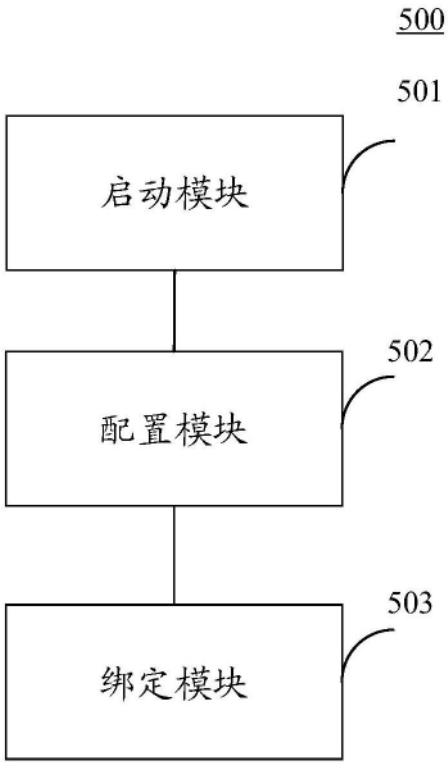

500
501
启动模块
502
配置模块
503
绑定模块

图5

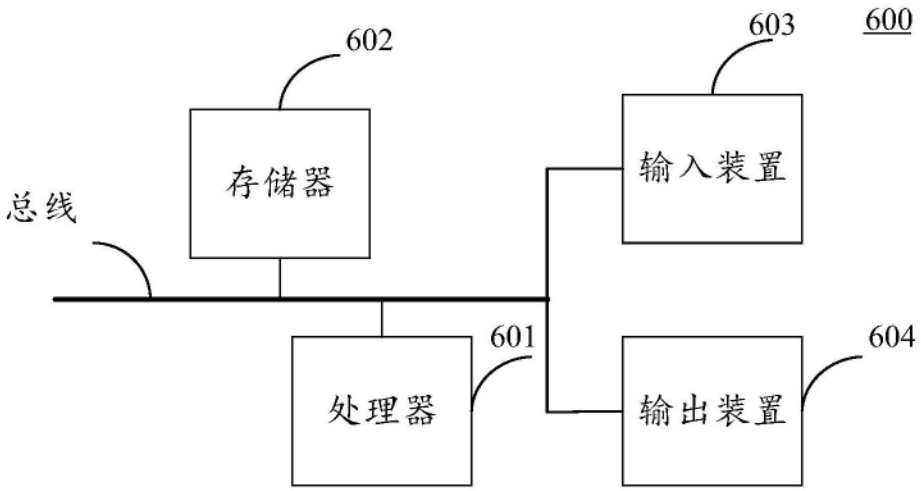

602
603
600
存储器
输入装置
总线
601
604
处理器
输出装置

图6