



(12) 发明专利申请

(10) 申请公布号 CN 104317734 A

(43) 申请公布日 2015.01.28

(21) 申请号 201410707583.1

(22) 申请日 2014.11.28

(71) 申请人 迈普通信技术股份有限公司

地址 610041 四川省成都市高新技术开发区  
九兴大道 16 号

(72) 发明人 范恒英

(74) 专利代理机构 北京中博世达专利商标代理  
有限公司 11274

代理人 申健

(51) Int. Cl.

G06F 12/06 (2006.01)

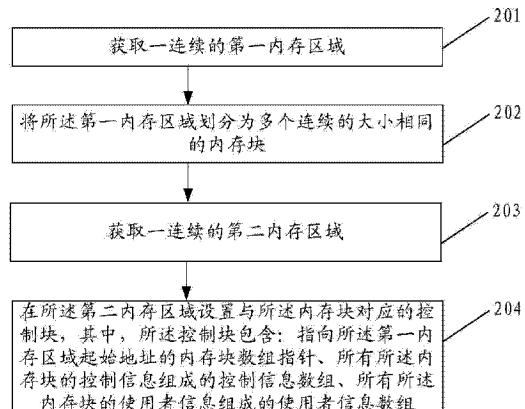
权利要求书1页 说明书4页 附图3页

(54) 发明名称

一种适用于 SLAB 的内存分配方法及装置

(57) 摘要

本发明公开了一种适用于 SLAB 的内存分配方法及装置,涉及计算机领域,通过将内存块的使用者信息与分配给用户使用的内存块分离,避免了越界写时使用者信息被改写的问题。本发明实施例提供的 SLAB 内存分配方法包括:获取一连续的第一内存区域;将所述第一内存区域划分为多个连续的大小相同的内存块;获取一连续的第二内存区域;在所述第二内存区域设置与所述内存块对应的控制块;其中,所述控制块包含:指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。



1. 一种适用于 SLAB 的内存分配方法,其特征在于,包括 :

获取一连续的第一内存区域;

将所述第一内存区域划分为多个连续的大小相同的内存块;

获取一连续的第二内存区域;

在所述第二内存区域设置与所述内存块对应的控制块;其中,所述控制块包含:指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。

2. 根据权利要求 1 所述的 SLAB 内存分配方法,其特征在于,在所述第二内存区域设置与所述内存块对应的控制块,包括 :

在所述第二内存区域设置与所述内存块数组指针的大小、所述控制信息数组的大小、所述使用者信息数组的大小之和相等的控制块。

3. 根据权利要求 1 或 2 所述的 SLAB 内存分配方法,其特征在于,

所述控制块还包含指向所述使用者信息数组起始地址的使用者信息数组指针。

4. 根据权利要求 1-3 任一项所述的 SLAB 内存分配方法,其特征在于,在获取一连续的第一内存区域之前,所述方法还包括 :

接收上层用户发送的分配请求,其中,所述分配请求包含用户请求分配的内存大小;

将所述第一内存区域划分为多个连续的大小相同的内存块包括:

将所述内存区域划分为多个连续的、大小与所述用户请求分配的内存大小相等的内存块。

5. 一种适用于 SLAB 的内存分配装置,其特征在于,包括 :

获取单元,用于获取一连续的第一内存区域;

划分单元,用于将所述第一内存区域划分为多个连续的大小相同的内存块;

所述获取单元,还用于获取一连续的第二内存区域;

配置单元,用于在所述第二内存区域设置与所述内存块对应的控制块,其中,所述控制块包含:指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。

6. 根据权利要求 5 所述的 SLAB 内存分配装置,其特征在于,所述配置单元,具体用于:

在所述第二内存区域设置与所述内存块数组指针的大小、所述控制信息数组的大小、所述使用者信息数组的大小之和相等的控制块。

7. 根据权利要求 5 和 6 所述的 SLAB 内存分配装置,其特征在于,

所述控制块还包含指向所述使用者信息数组起始地址的使用者信息数组指针。

8. 根据权利要求 1-3 任一项所述的 SLAB 内存分配装置,其特征在于,

所述获取单元,还用于在获取一连续的第一内存区域之前,接收上层用户发送的分配请求,其中,所述分配请求包含用户请求分配的内存大小;

相应的,所述划分单元,具体用于将所述内存区域划分为多个连续的、大小与所述用户请求分配的内存大小相等的内存块。

## 一种适用于 SLAB 的内存分配方法及装置

### 技术领域

[0001] 本发明涉及计算机技术领域，尤其涉及一种适用于 SLAB 的内存分配方法及装置。

### 背景技术

[0002] SLAB 是一种在 LINUX 和类 UNIX 操作系统内核中广泛使用的内存管理机制，该机制创建一系列 SLAB 内存池，将从底层内存管理机制分出的大块内存划分为特定大小的内存块放入 SLAB 内存池中，上层用户请求分配内存时直接从 SLAB 内存池中分配，上层用户释放内存时将内存释放回 SLAB 内存池，并通过与该 SLAB 内存池对应的 SLAB 控制块以链表方式对 SLAB 内存池中的每个内存块进行管理。

[0003] 其中，LINUX 内核中的 SLAB 内存管理机制还可以记录内存块的使用者信息（如使用者调用内存分配接口函数或内存释放接口函数时的指令地址或者调用堆栈信息），以便于使用者利用内存块的使用者信息分析定位内存泄露、越界写、内存未初始化就使用、内存释放后继续使用等内存使用的相关问题（bug）。例如，图 1 为现有 SLAB 内存的结构图。如图 1 所示，包含内存块数组以及控制该内存块数组的 SLAB 控制块，其中，内存块数组为一段连续的物理内存区域，包含多个内存大小相等的内存块，以及，与内存块相邻的内存块的使用者信息的记录区域，SLAB 控制块包含指向内存块数组起始地址的内存块数组指针以及内存块对应的控制信息，其中，控制信息包含内存块的状态信息（如已分配、空闲）。

[0004] 由于现有 SLAB 内存管理机制中记录内存块的使用者信息的区域与内存块区域相邻接，当使用者发生内存越界写时，内存块的使用者信息记录通常会被改写，导致记录的信息无效，使得定位内存 bug 困难。例如，如图 1 所示，假设用户分配了内存块 1，且内存块 1 的大小为 32 字节，此时，用户若向内存块 1 中写入 36 字节的数据，则会使得多出的 4 字节数据写入内存块 1 的使用者信息的区域，使得内存块的使用者信息的记录被改写。

### 发明内容

[0005] 本发明的实施例提供一种适用于 SLAB 的内存分配方法及装置，以解决现有 SLAB 内存管理机制中，在使用者越界写的情况下，内存块的使用者信息记录被改写的问题。

[0006] 为达到上述目的，本发明的实施例采用如下技术方案：

[0007] 第一方面，本发明实施例提供一种适用于 SLAB 的内存分配方法，包括：

[0008] 获取一连续的第一内存区域；

[0009] 将所述第一内存区域划分为多个连续的大小相同的内存块；

[0010] 获取一连续的第二内存区域；

[0011] 在所述第二内存区域设置与所述内存块对应的控制块，其中，所述控制块包含：指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。

[0012] 第二方面，本发明实施例提供一种适用于 SLAB 的内存分配装置，包括：

[0013] 获取单元，用于获取一连续的第一内存区域；

[0014] 划分单元,用于将所述第一内存区域划分为多个连续的大小相同的内存块；  
[0015] 所述获取单元,还用于获取一连续的第二内存区域；  
[0016] 配置单元,用于在所述第二内存区域设置与所述内存块对应的控制块,其中,所述控制块包含:指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。  
[0017] 由上可知,本发明实施例提供的 SLAB 内存分配方法及装置,与现有技术相比,通过将内存块的使用者信息配置在与内存块对应的控制块中,来实现将内存块区域与记录内存块的使用者信息的区域分离。因为本发明实施例中内存块的使用者信息与分配给用户使用的内存块不相邻,所以,当使用者越界写时,使用者信息不会被破坏,从而有效避免了现有 SLAB 内存管理机制中,使用者越界写的情况下,内存块的使用者信息记录被改写的问题。

## 附图说明

[0018] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0019] 图 1 为现有 SLAB 内存的结构图；  
[0020] 图 2 为本发明实施例提供的一种适用于 SLAB 的内存分配方法的流程图；  
[0021] 图 3 为本发明实施例提供的 SLAB 内存的结构图；  
[0022] 图 3A 为本发明实施例提供的 SLAB 内存的结构图；  
[0023] 图 4 为本发明实施例提供的一种适用于 SLAB 的内存分配装置的结构图。

## 具体实施方式

[0024] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都属于本发明保护的范围。

[0025] 需要说明的是,本发明提供的 SALB 内存分配方法适用于 Linux 操作系统,还适用于类 UNIX 等其他操作系统,本发明对此不进行限制,仅以 Linux 操作系统的 SLAB 内存分配方法为例进行说明。在 Linux 系统中,SLAB 内存管理有两部分组成:一是缓存 keme\_cache(即 SLAB 内存池),二是 SLAB,每一种 keme\_cache 对应一种数据结构,每一个 keme\_cache 包含多个 SLAB,每个 SLAB 包含多个特定大小的内存块,可以缓存多个对象,并且每个 SLAB 中的对象由与该 SLAB 对应的控制块通过索引链表方式进行管理。本发明实施例中仅以对一个 SLAB 的内存分配为例进行说明。

[0026] 图 2 为本发明实施例提供的一种适用于 SLAB 的内存分配方法的流程图,如图 2 所示,所述方法可以包括:

[0027] 201、获取一连续的第一内存区域。  
[0028] 优选的,可以从底层内存管理机制(如 Linux 中的 buddy 内存管理机制)获取一

连续的第一内存区域。

[0029] 202、将所述第一内存区域划分为多个连续的大小相同的内存块。

[0030] 其中，每块的大小由该 SLAB 所属的 kmem\_cache 的对象大小确定。

[0031] 203、获取一连续的第二内存区域。

[0032] 优选的，该第二内存区域可以和第一内存区域一起从底层内存管理机制（如 Linux 中的 buddy 内存管理机制），也可以从另一 kmem\_cache 获取。

[0033] 204、在所述第二内存区域设置与所述内存块对应的控制块，其中，所述控制块包含：指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。

[0034] 其中，所述控制块是一种数据结构，用于记录所述内存块的相关信息；所述内存块的控制信息用于标识内存块的使用情况，包含：已分配、空闲；所述内存块的使用者信息包含使用者调用内存分配的接口函数或释放内存的接口函数时的指令地址、和 / 或调用堆栈信息；图 3 为本发明实施例提供的 SLAB 内存的结构图，如图 3 所示，控制块中包含指向所述第一内存区域起始地址的内存块数组指针、内存块 1 的控制信息 – 内存块 4 的控制信息、以及内存块 1 的使用者信息 – 内存块 4 的使用者信息；控制头包含第一内存区域（内存块 1– 内存块 4）的起始地址。如此，将内存块的使用者信息存放在控制块中，以避免与内存块区域的连续存放。

[0035] 优选的，本发明实施例中，在配置控制块时，可以配置与所述内存块数组指针的大小、所述控制信息数组的大小、所述使用者信息数组的大小之和相等的控制块。

[0036] 此外，为了后期方便读取信息，在配置控制块时，将内存块的控制信息连续存放，组成控制信息数组；将内存块的使用者信息连续存放，组成使用者信息数组。

[0037] 进一步的，本发明实施例中为了快速的查找到内存块的使用者信息，如图 3A 所示，在控制块的控制头中还设置有一指向使用者信息数组的起始地址的使用者信息数组指针，以便通过该指针快速查看使用者信息。需要说明的是，该使用者信息数组指针不是必需，因为还可以通过其它方式计算出使用者信息数组的起始地址。

[0038] 由上可知，本发明实施例提供的 SLAB 内存分配方法，与现有技术相比，通过将内存块的使用者信息配置在与内存块对应的控制块中，来实现将内存块区域与记录内存块的使用者信息的区域分离。因为本发明实施例中内存块的使用者信息与分配给用户使用的内存块不相邻，所以，当使用者越界写时，使用者信息不会被破坏，从而有效避免了现有 SLAB 内存管理机制中，使用者越界写的情况下，内存块的使用者信息记录被改写的问题。

[0039] 图 4 为本发明实施例提供的一种适用于 SLAB 的内存分配装置的结构图，如图 4 所示，所述装置可以包括：

[0040] 获取单元 401，用于获取一连续的第一内存区域。

[0041] 划分单元 402，用于将所述第一内存区域划分为多个连续的大小相同的内存块。

[0042] 所述获取单元 401，用于获取一连续的第二内存区域。

[0043] 配置单元 403，用于在所述第二内存区域设置与所述内存块对应的控制块，其中，所述控制块包含：指向所述第一内存区域起始地址的内存块数组指针、所有所述内存块的控制信息组成的控制信息数组、所有所述内存块的使用者信息组成的使用者信息数组。

[0044] 其中，所述控制块是一种数据结构，用于记录所述内存块的相关信息；所述内存块

的控制信息用于标识内存块的使用情况,包含:已分配、空闲;所述内存块的使用者信息包含使用者调用内存分配的接口函数或释放内存的接口函数时的指令地址、和 / 或调用堆栈信息;图 3 为本发明实施例提供的 SLAB 内存的结构图,如图 3 所示,控制块中包含指向所述第一内存区域起始地址的内存块数组指针、内存块 1 的控制信息 - 内存块 4 的控制信息、以及内存块 1 的使用者信息 - 内存块 4 的使用者信息。如此,将内存块的使用者信息存放在控制块中,以避免与内存块区域的连续存放。

[0045] 进一步的,所述获取单元 401,具体用于从底层内存管理机制(如 Linux 中的 buddy 内存管理机制)获取一连续的第一内存区域以及第二内存区域;其中,第二内存区域也可从另一 kmem\_cache 中获取。

[0046] 进一步的,所述划分单元 402,具体用于将所述内存区域划分为多个连续的大小相同的内存块;每块的大小由该 SLAB 所属的 kmem\_cache 的对象大小确定。

[0047] 进一步的,配置单元 403,具体用于配置与所述内存块数组指针的大小、所述控制信息数组的大小、所述使用者信息数组的大小之和相等的控制块。

[0048] 此外,为了后期方便读取信息,在配置控制块时,将内存块的控制信息连续存放,组成控制信息数组;将内存块的使用者信息连续存放,组成使用者信息数组。

[0049] 进一步的,本发明实施例中为了快速的查找到内存块的使用者信息,如图 3A 所示,还在控制块中设置有使用者信息数组指针,以便通过该指针快速查看使用者信息。需要说明的是,该使用者信息数组指针不是必需,因为还可以通过其它方式计算出使用者信息数组的起始地址。

[0050] 由上可知,本发明实施例提供的 SLAB 内存分配装置,通过将内存块的使用者信息配置在与内存块对应的控制块中,来实现将内存块区域与记录内存块的使用者信息的区域分离。因为本发明实施例中内存块的使用者信息与分配给用户使用的内存块不相邻,所以,当使用者越界写时,使用者信息不会被破坏,从而有效避免了现有 SLAB 内存管理机制中,使用者越界写的情况下,内存块的使用者信息记录被改写的问题。

[0051] 以上所述,仅为本发明的具体实施方式,但本发明的保护范围并不局限于此,任何熟悉本技术领域的技术人员在本发明揭露的技术范围内,可轻易想到变化或替换,都应涵盖在本发明的保护范围之内。因此,本发明的保护范围应以所述权利要求的保护范围为准。

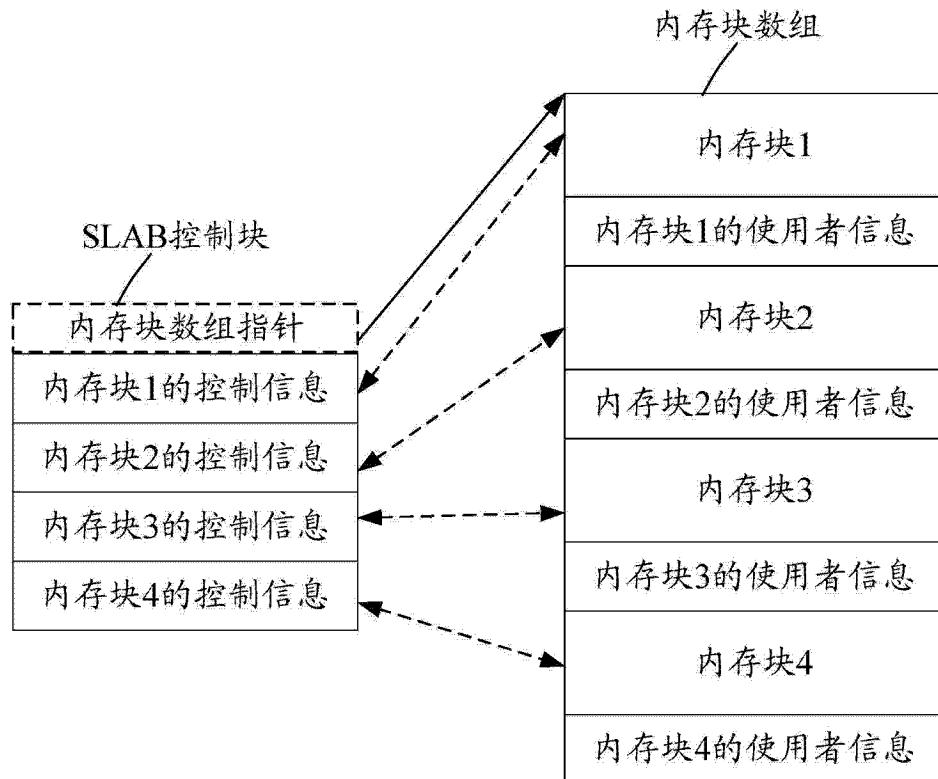


图 1

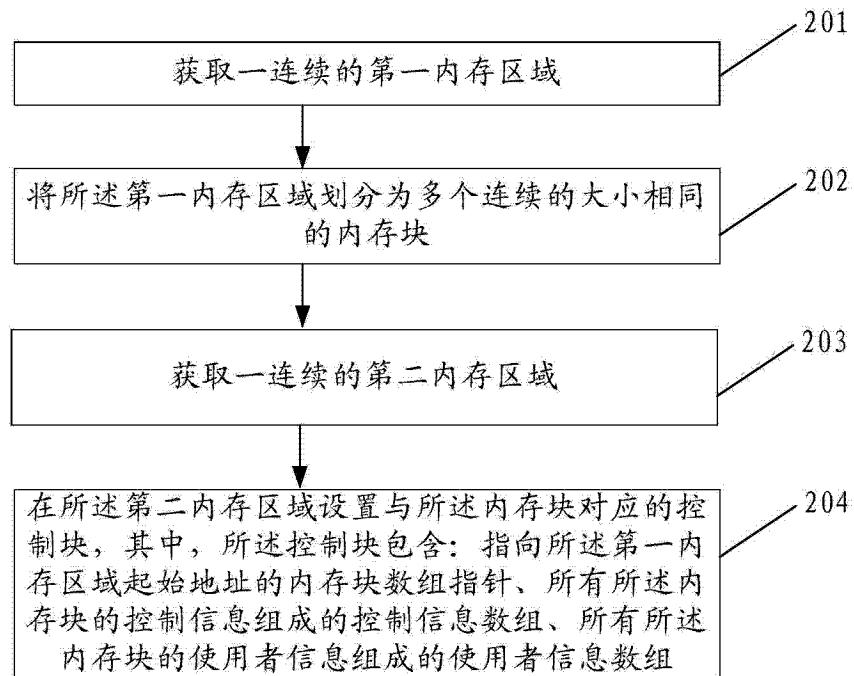


图 2

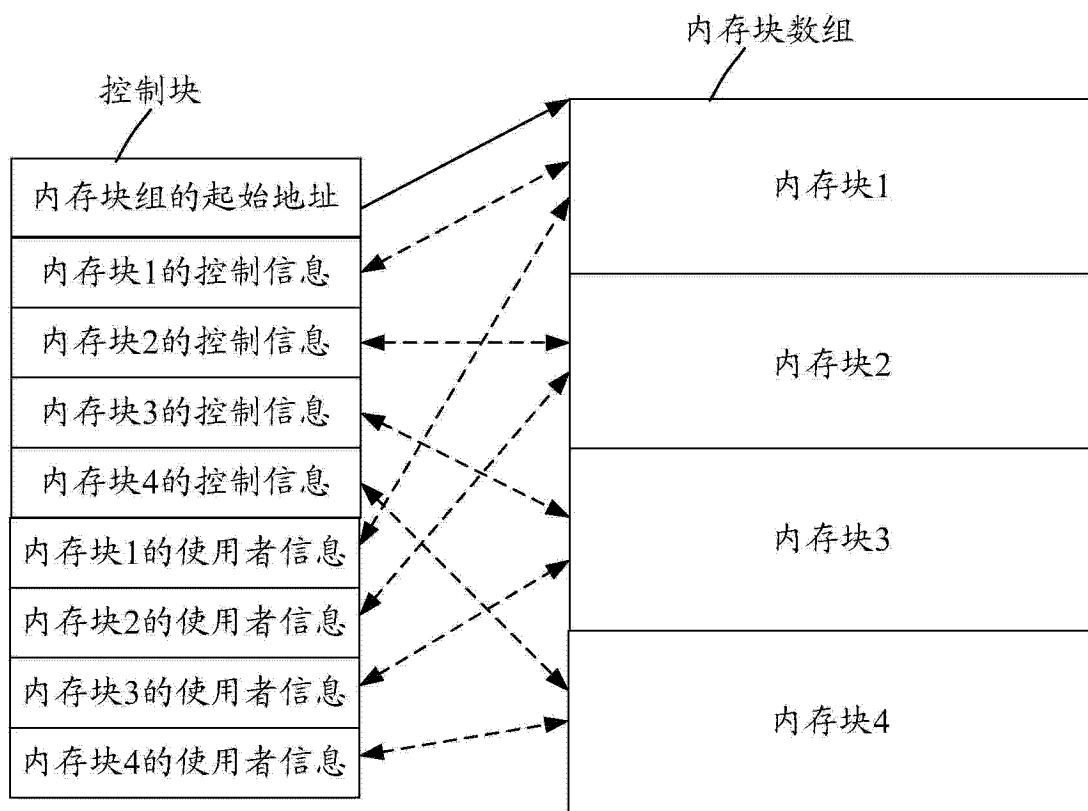


图 3

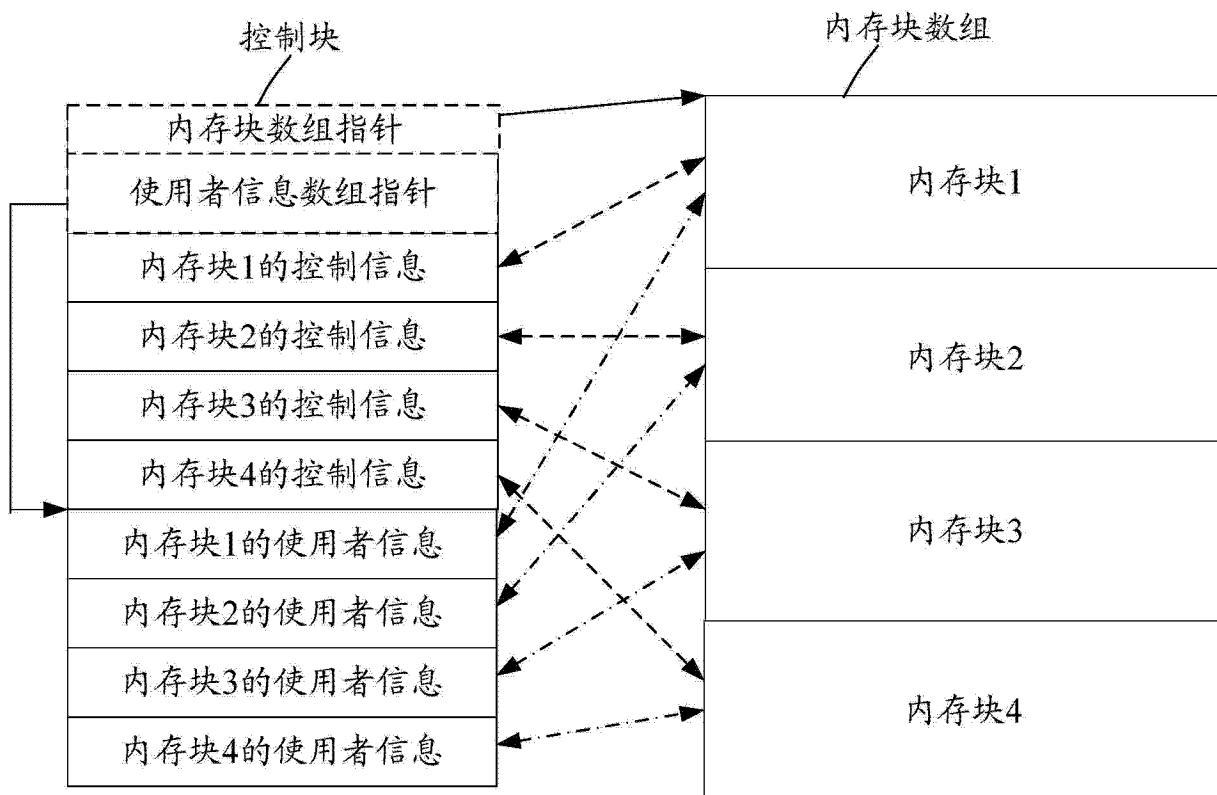


图 3A

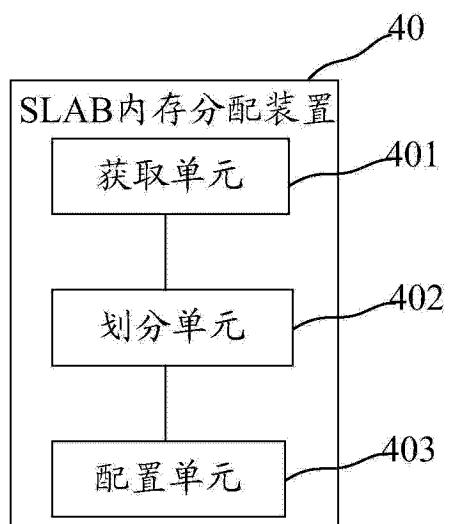


图 4