(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106980541 B
(45)授权公告日 2019.11.19

(21)申请号 201710142827.X

(22)申请日 2017.03.10

(65)同一申请的已公布的文献号
申请公布号 CN 106980541 A

(43)申请公布日 2017.07.25

(73)专利权人 浙江大学
地址 310013 浙江省杭州市西湖区余杭塘路866号

(72)发明人 王总辉 陈文智 梁平 李国玺

(74)专利代理机构 杭州天勤知识产权代理有限公司 33224
代理人 蒋琼

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

(56)对比文件
CN 101526923 A,2009.09.09,
CN 101572552 A,2009.11.04,
CN 106462494 A,2017.02.22,
CN 103620564 A,2014.03.05,
US 2006/0123035 A1,2006.06.08,

审查员 陈敏

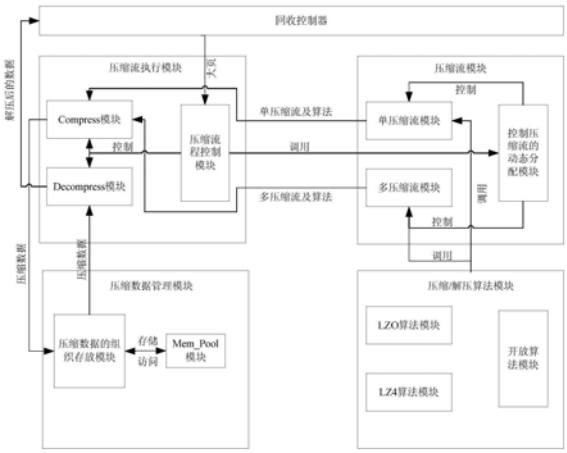
权利要求书2页 说明书7页 附图4页

(54)发明名称

一种大页内存压缩回收系统及方法

(57)摘要

本发明公开了一种大页内存压缩回收系统，包括：回收控制器，用于对大页进行解映射处理；压缩执行流模块，用于控制大页中内存数据的压缩和解压缩的执行流程，负责调用压缩流模块的单压缩或多压缩流压缩内存数据；压缩流模块，负责单压缩流或多压缩流的控制和调度，实时地根据CPU的负载情况分配空闲的单压缩流或多压缩流压缩内存数据；压缩/解压缩算法模块，用于对大页中内存数据的压缩或解压；压缩数据管理模块，用于压缩数据的动态存储管理；本发明还公开了一种利用上述大页内存压缩回收系统今进行大页内存压缩回收方法，本发明不仅可以极大地减少系统内存开销合提升CPU检索cache的速度，还能极大的拓展内存复用率。



CN 106980541 B

1.一种大页内存压缩回收系统,其特征在于,包括:

回收控制器,用于对接收的大页进行解映射处理,将压缩执行流模块发送来的解压缩数据转送给操作系统;

压缩执行流模块,用于控制解映射处理后大页中内存数据的压缩执行流程,负责调用压缩流模块的单压缩或多压缩流压缩内存数据,得到压缩数据;用于控制压缩数据管理模块中的压缩数据的解压缩执行流程,负责调用压缩流模块的单压缩或多压缩流解压缩压缩数据,得到解压缩数据;

压缩流模块,用于负责单压缩流或多压缩流的控制和调度,实时地根据CPU的负载情况分配空闲的单压缩流或多压缩流压缩内存数据或解压缩压缩数据;

压缩/解压缩算法模块,用于对大页中内存数据的压缩和压缩数据的解压缩;

压缩数据管理模块,用于压缩数据的动态存储管理;

具体地,所述的压缩执行流模块包括Compress模块、Decompress模块以及压缩流程控制模块,其中,Compress模块用于执行大页中内存数据的压缩,Decompress模块用于执行压缩数据的解压,压缩流程控制模块用于调用单压缩或多压缩流,并控制Compress模块执行大页中内存数据的压缩或控Decompress模块制执行压缩数据的解压;

所述的压缩流模块包括单压缩流模块、多压缩流模块以及控制压缩流动态分配模块,其中,控制压缩流动态分配模块用于根据CPU的负载情况分配单压缩流模块或多压缩流模块去执行压缩内存数据操作;

所述的压缩/解压缩算法模块包括LZO算法模块、LZ4算法模块以及开放算法模块,其中,LZO算法模块内置有压缩或解压的LZO算法,且该算法被单压缩流或多压缩流调用去执行压缩或解压,LZ4算法模块内置有压缩或解压的LZ4算法,且该算法被单压缩流或多压缩流调用去执行压缩或解压,开放算法模块用于将其他开源算法直接移植于该模块内,并被单压缩流或多压缩流调用;

所述的压缩数据管理模块包括压缩数据的组织存放模块和Mem_pool模块,其中,压缩数据的组织存放模块用于将接收的压缩数据存放于Mem_pool模块,Mem_pool模块用于压缩数据;

所述的压缩数据的组织存放模块包含用于对大页数据进行管理的红黑树,所述的红黑树的每个节点是三元组mapping、index、address,其中,mapping和index为每个大页page中独有的元素,以mapping为第一关键字和以index作为第二关键字在红黑树中进行排序,用于对大页进行标记;address保存存储压缩数据4k小页的虚拟地址。

2.一种应用权利要求1 所述的 大页内存压缩回收系统进行大页内存压缩回收方法,具体包括:

大页内存数据的压缩:回收控制器对大页进行加锁和unmap操作处理,并将处理后的大页发送至压缩执行流模块,压缩执行流模块调用压缩流模块中的压缩流和压缩/解压算法模块中的压缩算法对大页进行压缩,并将得到的压缩数据发送至压缩数据管理模块,利用红黑树将压缩数据存储于小页中;

压缩数据的解压缩:压缩执行流模块接收来自回收控制器的解压缩命令,通过压缩数据管理模块中的红黑树查找压缩数据,并调用压缩流模块中的压缩流和压缩/解压算法模块中的解压算法对小页进行解压缩,然后将解压缩后的数据写入当前大页,发送至回收流

控制器。

3.根据权利要求2所述的大页内存压缩回收方法,其特征在于,所述的大页内存数据的压缩具体包括以下步骤:

(a-1)利用回收控制器对冷热页追踪隔离出来的大页进行加锁;

(a-2)利用回收控制器对该大页进行unmap操作,清空所有映射此大页的页表项pte;

(a-3)Compress模块判断大页是否超出压缩限制,若是,则不进行压缩,若否,执行步骤(a-4);

(a-4)Compress模块检查大页是否为全0页,若是,将该大页进行特殊标记,并将数据直接扔掉,不进行压缩;若否,执行步骤(a-5);

(a-5)压缩流程控制模块根据回收控制器发送来的大页向控制压缩流的动态分配模块发出调用压缩流的指令;

(a-6)控制压缩流的动态分配模块根据CPU的负载情况分配空闲的单压缩流或多压缩流供压缩流程控制模块调用,若没有空闲的单压缩流或多压缩流,控制压缩流的动态分配模块则进行睡眠等待,直到存在空闲的单压缩流或多压缩流再进行分配;

(a-7)被分配的单压缩流或多压缩流调用压缩/解压缩算法模块中的一个或多个压缩算法随同自身被压缩流程控制模块调用;

(a-8)压缩流程控制模块根据调用的单压缩流或多压缩流以及压缩算法控制Compress模块对大页中的内存数据进行压缩,得到压缩数据;

(a-9)压缩数据的组织存放模块将接收到的压缩流中的压缩数据拷贝至Mem_pool模块的4k小页中,并将4k小页的address以及当前大页的mapping和index插入到红黑树里;

(a-10)压缩流程控制模块释放单压缩流或多压缩流,将当前大页从页缓存Page Cache中移除;

(a-11)回收控制器接触页锁,并释放该大页。

4.根据权利要求2所述的大页内存压缩回收方法,其特征在于,所述的压缩数据的解压缩具体包括以下步骤:

(b-1)当访问大页触发缺页异常时,回收控制器判断大页的页表项pte是否为空,若是,执行步骤(b-2),若否,则不是由压缩造成的缺页异常,结束解压缩;

(b-2)Decompress模块判断页缓存Page Cache中是否有对应的文件页,若是,将文件页映射给当前产生缺页异常的页表项pte;若否,执行步骤(b-3);

(b-3)Decompress模块申请一个新大页,并将该新大页加入至页缓存Page Cache;

(b-4)Decompress模块利用当前产生缺页异常的vma找到相应的mapping和index,并利用二元组mapping和index到红黑树中查找是否存在压缩数据,若是,执行步骤(b-5);若否,则是第一次访问,执行步骤(b-6);

(b-5)Decompress模块从红黑树中找到压缩数据的地址,通过标志位判断大页是否是全0页,若是,则直接对大页写0;若否,执行步骤(b-6);

(b-6)压缩流程控制模块调用压缩/解压缩算法模块中的一个或多个压缩解压缩算法并控制Decompress模块对4k小页中的压缩数据进行解压缩,并将解压后的数据写入当前大页;

(b-7)把当前的页表项pte映射到该大页。

一种大页内存压缩回收系统及方法

技术领域

[0001] 本发明属于计算机操作系统技术领域，具体涉及一种大页内存压缩回收系统及方法。

背景技术

[0002] 随着计算机技术的发展，特别是巨型机服务器的内存越来越大。而现有的服务器普遍采用Linux内核对内存按4k一个页的粒度进行管理，对于大内存就显得捉襟见肘。比如对于256G的大内存，用4k页机制的话，单页表描述符就需要2G的内存空间。同时其对应的页表也很巨大，导致CPU的cache因不能全部缓存页表而造成CPU的检索过慢，从而影响性能。

[0003] 对此Linux内核采用了一种Hugetlb机制把多个连续4k页拼接成2M大页来对大内存进行管理。但是Hugetlb机制不支持大页压缩回收机制，以至于大页内存的利用率不高。因为在大页内存中会存在许多活跃度不高的大页，对其进行压缩存储可以提高内存的利用率。

[0004] 4k页机制虽然能对内存进行压缩回收但是开销太大以及Hugetlb机制虽然提供2M大页支持但是不能对其进行压缩回收。

[0005] 现有的典型的内存复用技术，主要有以下几种：内存压缩、内存去重、ballooning技术、transcendent memory技术以及SWAP技术等。

[0006] 内存去重(memory dedulplication)，就是把内存中的重复内容消除以提高内存利用率的技术。由于现代计算机的内存是分页管理的，所以内存去重技术往往指的就是页共享(page sharing)技术。它通过共享相同内容的物理内存页从而降低物理内存消耗。内存去重是去掉重复的物理内存，以节省物理内存。

[0007] Ballooning技术和transcendent memory技术是基于虚拟化平台上的内存复用技术。简而言之，就是通过监测虚拟化平台上各个虚拟机的实际内存使用情况，来对各个虚拟机的内存进行管理，即将实际使用内存少于分配的内存的虚拟机内存进行回收，来供给其余用途。

[0008] SWAP技术是Linux内核最常用的一种内存扩容技术。该技术把不常用的冷页内存交换到磁盘的swap分区中，需要访问的时候再把其读取到内存中。该机制由于需要读写磁盘，导致系统的响应特别慢。

[0009] 内存压缩则比较好理解：由于现今计算机的CPU利用率比较空闲，因此，通过使CPU变得“忙碌”起来，将内存中的一些“冷页”进行压缩然后重新存放到内存中，从而来使我们有更多的可用内存。

[0010] 内存压缩可以有效的提高内存有效容量，降低页故障率，同时不会有大容量内存的使用带来的能耗和空间消耗。同时可以提高内存带宽使用率。但是压缩和解压缩会带来延迟，如果处理不当可能把压缩带来的好处抵消掉。因此有效使用压缩技术需要谨慎选择实现框架、压缩算法等等。

发明内容

[0011] 鉴于上述，本发明提供了一种大页内存压缩回收系统及方法，主要是基于全新的大页内存管理机制对大页内存进行压缩回收，把不经常访问的大页内存进行压缩回收，腾出的空间让其他用户进程利用，以提高内存的复用率。

[0012] 一种大页内存压缩回收系统，包括：

[0013] 回收控制器，用于对接收的大页进行解映射处理，将压缩执行流模块发送来的解压缩数据转送给操作系统；

[0014] 压缩执行流模块，用于控制解映射处理后大页中内存数据的压缩执行流程，负责调用压缩流模块的单压缩或多压缩流压缩内存数据，得到压缩数据；用于控制压缩数据管理模块中的压缩数据的解压缩执行流程，负责调用压缩流模块的单压缩或多压缩流解压缩压缩数据，得到解压缩数据；

[0015] 压缩流模块，用于负责单压缩流或多压缩流的控制和调度，实时地根据CPU的负载情况分配空闲的单压缩流或多压缩流压缩内存数据或解压缩压缩数据；

[0016] 压缩/解压缩算法模块，用于对大页中内存数据的压缩和压缩数据的解压缩；

[0017] 压缩数据管理模块，用于压缩数据的动态存储管理。

[0018] 所述的回收控制器用于对从冷热页跟踪模块隔离的冷大页进行解映射处理。因为在压缩期间，进程的页表项还映射了该页，因此必须对其进行解除映射，保证在压缩数据期间不能有其他用户进程修改大页内存里的数据。

[0019] 所述的压缩执行流模块包括Compress模块、Decompress模块以及压缩流程控制模块，其中，Compress模块用于执行大页中内存数据的压缩，Decompress模块用于执行压缩数据的解压，压缩流程控制模块用于调用单压缩或多压缩流，并控制Compress模块执行大页中内存数据的压缩或控Decompress模块制执行压缩数据的解压。

[0020] 所述的压缩流模块包括单压缩流、多压缩流以及控制压缩流动态分配模块，其中，控制压缩流动态分配模块用于根据CPU的负载情况分配单压缩流或多压缩流去执行压缩内存数据操作。

[0021] 压缩流是缓存buffer和算法私有的，每个压缩操作都会独享压缩流。由于上层调用者kswap是多线程的，如果采用单压缩流，将限制kswap的并发。同时如果单压缩流出现数据奔溃或卡住的情况，所有的其他压缩操作将一直处于等待状态，这样将造成效率低下，而多流压缩架构可以让多个压缩操作并行执行，这大大提高了压缩的效率和稳定性。

[0022] 所述的压缩/解压缩算法模块包括LZO算法模块、LZ4算法模块以及开放算法模块，其中，LZO算法模块内置有压缩或解压的LZO算法，且该算法被单压缩流或多压缩流调用去执行压缩或解压，LZ4算法模块内置有压缩或解压的LZ4算法，且该算法被单压缩流或多压缩流调用去执行压缩或解压，开放算法模块用于将其他开源算法直接移植于该模块内，并被单压缩流或多压缩流调用。

[0023] LZO和LZ4算法是比较成熟的压缩算法。目前Linux内核3.10版本只有LZO算法，Linux内核3.15版本引入了LZ4算法。本系统直接移植这两种算法，LZO算法比LZ4算法有更好的压缩比和更快的压缩速度，但是LZ4在解压速度上比LZO算法更优秀。由于大页有2M大，必须对其进行快速的解压，因此LZ4算法在本系统中更具优势。这两个模块是独立模块，可以用开放算法模块中引入的其他算法替换。

[0024] 所述的压缩数据管理模块包括压缩数据的组织存放模块和Mem_pool模块，其中，压缩数据的组织存放模块用于将接收的压缩数据存放于Mem_pool模块，Mem_pool模块用于压缩数据。

[0025] 如果将2M大页进行压缩后的压缩数据还是存放于2M大页内，这会导致空间利用率低和数据迁移慢的问题。处于空间率用率和安全性通用性的考虑，本发明运用vmalloc函数将压缩数据放在内核空间的4k小页内，占用整数个4k小页，不满4k小页也占用整个4k小页。这样每个2M大页压缩后的压缩数据最多就浪费4k的内存空间，空间浪费率为1/512。对于解压缩以后，系统将整个4k小页直接释放掉，不会造成空间碎片化。

[0026] 所述的压缩数据的组织存放模块包含用于对大页数据进行管理的红黑树，所述的红黑树的每个节点是三元组mapping、index、address，其中，mapping和index为每个大页page中独有的元素，以mapping为第一关键字和以index作为第二关键字在红黑树中进行排序，用于对大页进行标记；address保存存储压缩数据4k小页的虚拟地址。这样每次解压缩的时候只需要log(n)的复杂度就可以找到压缩数据的地址进行解压缩。

[0027] 压缩数据的组织存放模块改变了页表项的映射，压缩前虚拟地址映射的是一块连续的2M物理页，压缩以后将虚拟地址映射到一组4K页。本质就是开始时虚拟地址指向2M页，压缩以后将虚拟地址直指向新存放的地址。

[0028] 本发明还提供了一种大页内存压缩回收方法，具体包括：

[0029] 大页内存数据的压缩：回收控制器对大页进行枷锁和unmap操作处理，并将处理后的大页发送至压缩执行流模块，压缩执行流模块调用压缩流模块中的压缩流和压缩/解压算法模块中的压缩算法对大页进行压缩，并将得到的压缩数据发送至压缩数据管理模块，利用红黑树将压缩数据存储于小页中。

[0030] 压缩数据的解压缩：压缩执行流模块接收来自回收控制器的解压缩命令，通过压缩数据管理模块中的红黑树查找压缩数据，并调用压缩流模块中的压缩流和压缩/解压算法模块中的解压算法对小页进行解压缩，然后将解压缩后的数据写入当前大页，发送至回收流控制器。

[0031] 所的大页内存数据的压缩具体包括以下步骤：

[0032] (a-1)利用回收控制器对冷热页追踪隔离出来的大页进行加锁，防止其他内核模块或进程占用该页；

[0033] (a-2)利用回收控制器对该大页进行unmap操作，清空所有映射此大页的页表项pte，保证在压缩过程中用户进程不能修改该大页的数据；

[0034] (a-3)Compress模块判断大页是否超出压缩限制，若是，则不进行压缩，若否，执行步骤(a-4)；

[0035] (a-4)Compress模块检查大页是否为全0页，若是，将该大页进行特殊标记，并将数据直接扔掉，不进行压缩；若否，执行步骤(a-5)；

[0036] (a-5)压缩流程控制模块根据回收控制器发送来的大页向控制压缩流的动态分配模块发出调用压缩流的指令；

[0037] (a-6)控制压缩流的动态分配模块根据CPU的负载情况分配空闲的单压缩流或多压缩流供压缩流程控制模块调用，若没有空闲的单压缩流或多压缩流，控制压缩流的动态分配模块则进行睡眠等待，直到存在空闲的单压缩流或多压缩流再进行分配；

[0038] (a-7)被分配的单压缩流或多压缩流调用压缩/解压缩算法模块中的一个或多个压缩算法随同自身被压缩流程控制模块调用；

[0039] (a-8)压缩流程控制模块根据调用的单压缩流或多压缩流以及压缩算法控制Compress模块对大页中的内存数据进行压缩，得到压缩数据；

[0040] (a-9)压缩数据的组织存放模块将接收到的压缩流中的压缩数据拷贝至Mem_pool模块的4k小页中，并将4k小页的address以及当前大页的mapping和index插入到红黑树里；

[0041] (a-10)压缩流程控制模块释放单压缩流或多压缩流，将当前大页从页缓存Page Cache中移除；

[0042] (a-11)回收控制器接触页锁，并释放该大页。

[0043] 所述的压缩数据的解压缩具体包括以下步骤：

[0044] (b-1)当访问大页触发缺页异常时，回收控制器判断大页的页表项pte是否为空，若是，执行步骤(b-2)，若否，则不是由压缩造成的缺页异常，结束解压缩；

[0045] (b-2)Decompress模块判断页缓存Page Cache中是否有对应的文件页，若是，将文件页映射给当前产生缺页异常的页表项pte；若否，执行步骤(b-3)；

[0046] (b-3)Decompress模块申请一个新大页，并将该新大页加入至页缓存Page Cache；

[0047] (b-4)Decompress模块利用当前产生缺页异常的vma找到相应的mapping和index，并利用该二元组mapping和index到红黑树中查找是否存在压缩数据，若是，执行步骤(b-5)；若否，则是第一次访问，执行步骤(b-6)；

[0048] (b-5)Decompress模块从红黑树中找到压缩数据的地址，通过标志位判断大页是否是全0页，若是，则直接对大页写0。若否，执行步骤(b-6)；

[0049] (b-6)压缩流程控制模块调用压缩/解压缩算法模块中的一个或多个压缩解压缩算法并控制Decompress模块对4k小页中的压缩数据进行解压缩，并将解压后的数据写入当前大页；

[0050] (b-7)把当前的页表项pte映射到该大页。

[0051] 本发明基于全新的4k小页和2M大页分开管理框架，对2M大页进行压缩回收，不仅可以极大地减少系统内存开销合提升CPU检索cache的速度，还能极大的拓展内存复用率。经测试，可以对内存进行“扩容”2倍以上。比如256G的物理内存，用户能够用的内存可达500G以上。同时本压缩系统采用模块化设计，支持热插拔，和原生的Linux内核耦合性低，可方便修改适应不同的内核版本。

附图说明

[0052] 图1是本发明实施例大页内存压缩构架概要图；

[0053] 图2是本发明实施例大页内存压缩回收系统的结构示意图；

[0054] 图3是本发明实施例改变页表项的映射原理图；

[0055] 图4是本发明实施例大页内存数据的压缩方法流程图；

[0056] 图5是本发明实施例压缩数据的解压缩方法流程图。

具体实施方式

[0057] 为了更为具体地描述本发明，下面结合附图及具体实施方式对本发明的技术方案

进行详细说明。

[0058] 图1所示的是大页内存压缩构架概要图，该架构把物理内存分为两个部分：4k空间和2M空间。利用全新的大页内存独立管理机制管理2M大页内存。而原生的Linux内存管理机制负责管理4k页空间，针对于冷热页追踪模块筛选出来的冷大页，采用本发明提出的大页内存压缩回收系统对其进行压缩存储管理。

[0059] 图2所示的是实施例大页内存压缩回收系统的结构示意图，如图2所示，本实施例大页内存压缩回收系统，包括：回收控制器、压缩执行流模块、压缩流模块、压缩/解压缩算法模块以及压缩数据管理模块。

[0060] 回收控制器用于对从冷热页跟踪模块隔离的冷大页进行解映射处理。因为在压缩期间，进程的页表项还映射了该页，因此必须对其进行解除映射，保证在压缩数据期间不能有其他用户进程修改大页内存里的数据。

[0061] 压缩执行流模块包括Compress模块、Decompress模块以及压缩流程控制模块，其中，Compress模块用于执行大页中内存数据的压缩，Decompress模块用于执行压缩数据的解压，压缩流程控制模块用于调用单压缩或多压缩流，并控制Compress模块执行大页中内存数据的压缩或控Decompress模块制执行压缩数据的解压。

[0062] 压缩流模块包括单压缩流、多压缩流以及控制压缩流动态分配模块，其中，控制压缩流动态分配模块用于根据CPU的负载情况分配单压缩流或多压缩流去执行压缩内存数据操作。

[0063] 压缩流是缓存buffer和算法私有的，每个压缩操作都会独享压缩流。由于上层调用者kswap是多线程的，如果采用单压缩流，将限制kswap的并发。同时如果单压缩流出现数据奔溃或卡住的情况，所有的其他压缩操作将一直处于等待状态，这样将造成效率低下，而多流压缩架构可以让多个压缩操作并行执行，这大大提高了压缩的效率和稳定性。

[0064] 压缩/解压缩算法模块包括LZO算法模块、LZ4算法模块以及开放算法模块，其中，LZO算法模块内置有压缩或解压的LZO算法，且该算法被单压缩流或多压缩流调用去执行压缩或解压，LZ4算法模块内置有压缩或解压的LZ4算法，且该算法被单压缩流或多压缩流调用去执行压缩或解压，开放算法模块用于将其它开源算法直接移植于该模块内，并被单压缩流或多压缩流调用。

[0065] LZO和LZ4算法是比较成熟的压缩算法。目前Linux内核3.10版本只有LZO算法，Linux内核3.15版本引入了LZ4算法。本系统直接移植这两种算法，LZO算法比LZ4算法有更好的压缩比和更快的压缩速度，但是LZ4在解压速度上比LZO算法更优秀。由于大页有2M大，必须对其进行快速的解压，因此LZ4算法在本系统中更具优势。这两个模块是独立模块，可以用开放算法模块中引入的其他算法替换。

[0066] 压缩数据管理模块包括压缩数据的组织存放模块和Mem_pool模块，其中，压缩数据的组织存放模块用于将接收的压缩数据存放于Mem_pool模块，Mem_pool模块用于压缩数据。

[0067] 如果将2M大页进行压缩后的压缩数据还是存放于2M大页内，这会导致空间利用率低和数据迁移慢的问题。处于空间率用率和安全性通用性的考虑，本发明运用vmalloc函数将压缩数据放在内核空间的4k小页内，占用整数个4k小页，不满4k小页也占用整个4k小页。这样每个2M大页压缩后的压缩数据最多就浪费4k的内存空间，空间浪费率为1/512。对于解

压缩以后，系统将整个4k小页直接释放掉，不会造成空间碎片化。

[0068] 压缩数据的组织存放模块包含用于对大页数据进行管理的红黑树，所述的红黑树的每个节点是三元组mapping、index、address，其中，mapping和index为每个大页page中独有的元素，以mapping为第一关键字和以index作为第二关键字在红黑树中进行排序，用于对大页进行标记；address保存存储压缩数据4k小页的虚拟地址。这样每次解压缩的时候只需要log(n)的复杂度就可以找到压缩数据的地址进行解压缩。

[0069] 压缩数据的组织存放模块改变了页表项的映射，压缩前虚拟地址映射的是一块连续的2M物理页，压缩以后将虚拟地址映射到一组4K页。本质就是开始时虚拟地址指向2M页，压缩以后将虚拟地址直指向新存放的地址，其原理图如图3所示。

[0070] 本实施例还提供了一种利用上述大页内存压缩回收系统今进行大页内存压缩回收方法，包括大页内存数据的压缩方法和压缩数据的解压缩方法。

[0071] 如图4所示的是本发明提供的大页内存数据的压缩方法实施例流程示意图，本实施例所描述的大页内存数据的压缩方法，具体包括以下步骤：

[0072] S101，利用回收控制器对冷热页追踪隔离出来的大页进行加锁，防止其他内核模块或进程占用该页。

[0073] S102，利用回收控制器对该大页进行unmap操作，清空所有映射此大页的页表项pte，保证在压缩过程中用户进程不能修改该大页的数据。

[0074] S103，Compress模块判断大页是否超出压缩限制，若是，即大页没有空间存放数据，则不进行压缩，若否，执行步骤S104。

[0075] S104，Compress模块判断大页是否为全0页，若是，将该大页进行特殊标记，并将数据直接扔掉，不进行压缩；若否，执行S105。

[0076] S105，压缩流程控制模块根据回收控制器发送来的大页向控制压缩流的动态分配模块发出调用压缩流的指令。

[0077] S106，控制压缩流的动态分配模块根据CPU的负载情况分配空闲的多压缩流供压缩流程控制模块调用。

[0078] S107，被分配的多压缩流调用压缩/解压缩算法模块中的LZ4算法随同自身被压缩流程控制模块调用。

[0079] S108，压缩流程控制模块根据调用的多压缩流以及LZ4算法控制Compress模块对大页中的内存数据进行压缩，得到压缩数据。

[0080] S109，压缩数据的组织存放模块将接收到的压缩流中的压缩数据拷贝至Mem_pool模块的4k小页中，并将4k小页的address以及当前大页的mapping和index插入到红黑树里。

[0081] S110，压缩流程控制模块释放多压缩流，将当前大页从页缓存Page Cache中移除。

[0082] S111，回收控制器接触页锁，并释放该大页。

[0083] 如图5所示是本发明提供的压缩数据的解压缩方法实施例流程示意图，本实施例所描述的压缩数据的解压缩方法，具体包括以下步骤；

[0084] S201，当访问大页触发缺页异常时，回收控制器判断大页的页表项pte是否为空，若是，执行S202，若否，则不是由压缩造成的缺页异常，结束解压缩。

[0085] S202，Decompress模块判断页缓存Page Cache中是否有对应的文件页，若是，将文件页映射给当前产生缺页异常的页表项pte；若否，执行S203。

[0086] S203,Decompress模块申请一个新大页,并将该新大页加入至页缓存Page Cache。

[0087] S204,Decompress模块利用当前产生缺页异常的vma找到相应的mapping和index,并利用该二元组mapping和index到红黑树中查找是否存在压缩数据,若是,执行S205;若否,则是第一次访问,执行S206。

[0088] S205,Decompress模块从红黑树中找到压缩数据的地址,通过标志位判断大页是否是全0页,若是,则直接对大页写0。若否,执行S206。

[0089] S206,压缩流程控制模块调用压缩/解压缩算法模块中的LZO算法并控制Decompress模块对4k小页中的压缩数据进行解压缩,并将解压后的数据写入当前大页。

[0090] S207,把当前的页表项pte映射到该大页。

[0091] 以上所述的具体实施方式对本发明的技术方案和有益效果进行了详细说明,应理解的是以上所述仅为本发明的最优选实施例,并不用于限制本发明,凡在本发明的原则范围内所做的任何修改、补充和等同替换等,均应包含在本发明的保护范围之内。

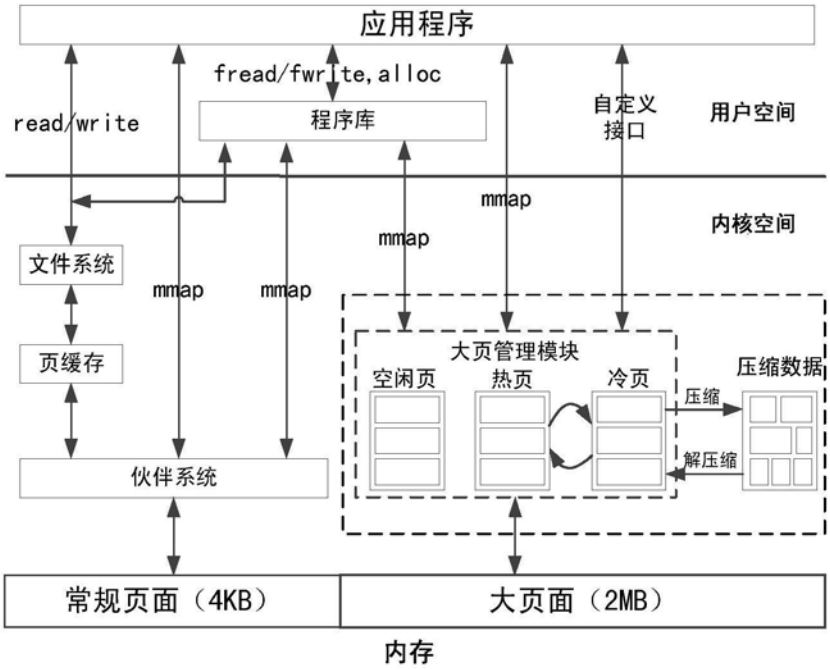

应用程序
fread/fwrite,alloc
read/write
程序库
自定义
接口
用户空间
mmap
内核空间
mmap
文件系统
mmap
mmap
页缓存
大页管理模块
空闲页
热页
冷页
压缩数据
压缩
解压缩
伙伴系统
常规页面（4KB）
大页面（2MB）
内存

图1

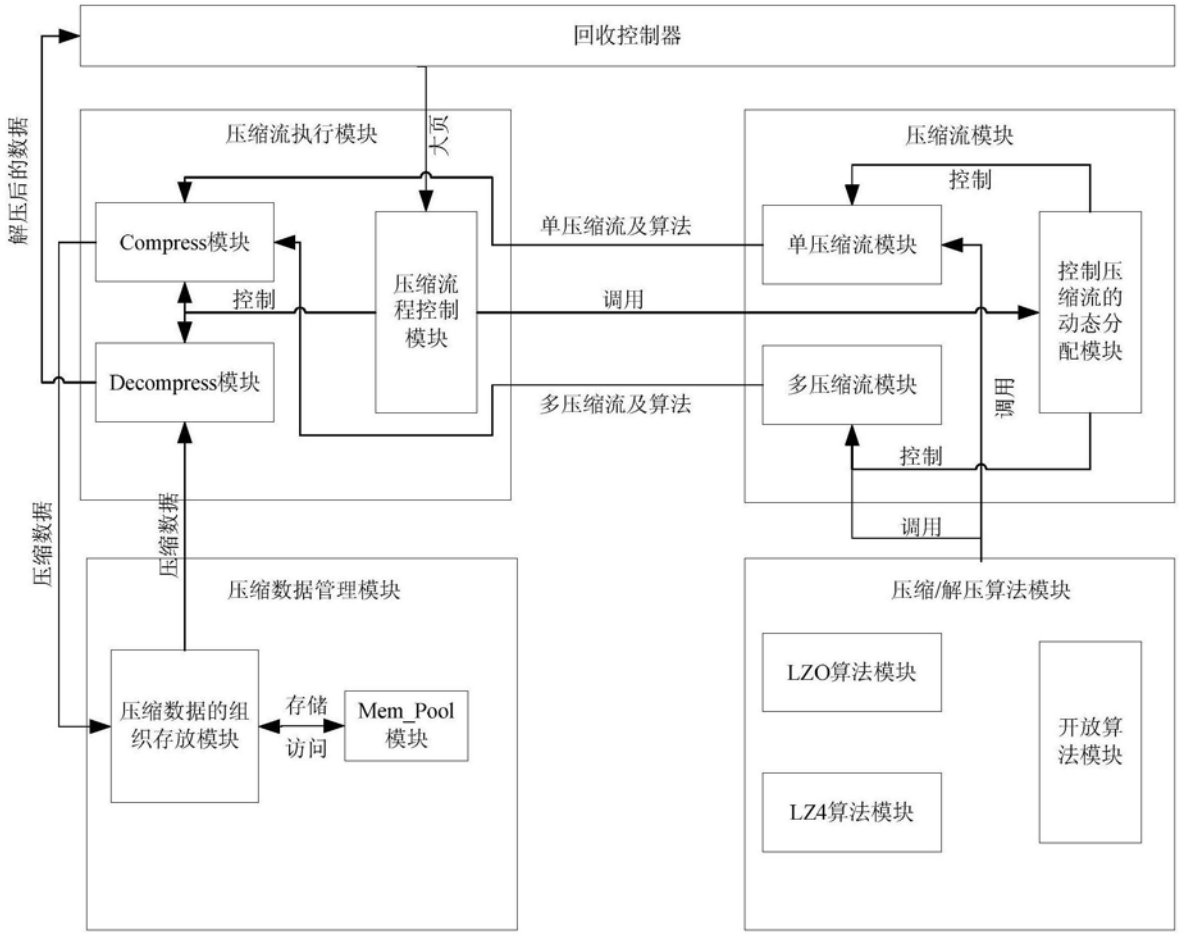

回收控制器
压缩流执行模块
大页
解压后的数据
Compress模块
控制
Decompress模块
压缩流程控制模块
单压缩流及算法
调用
多压缩流及算法
压缩流模块
控制
单压缩流模块
控制压缩流的动态分配模块
多压缩流模块
调用
控制
调用
压缩数据
压缩数据管理模块
压缩数据的组织存放模块
存储
访问
Mem_Pool模块
压缩/解压算法模块
LZO算法模块
LZ4算法模块
开放算法模块

图2

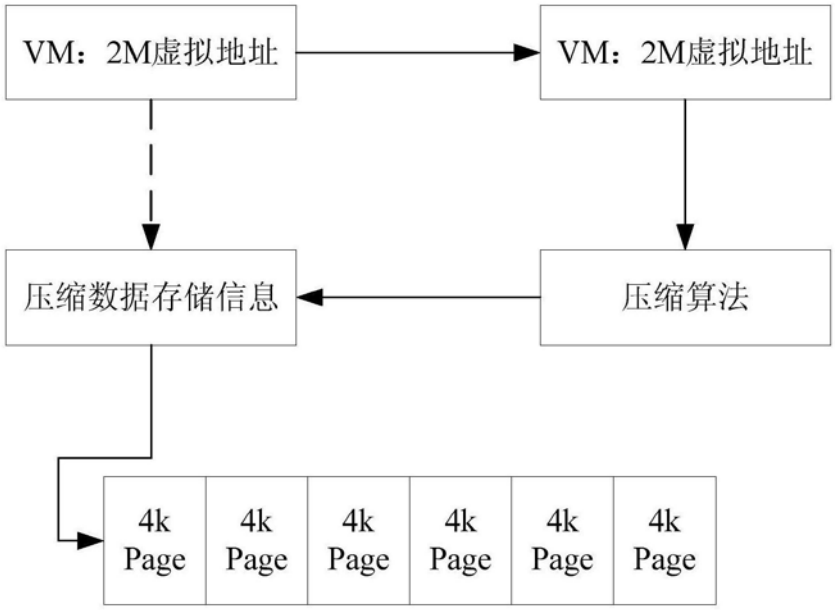

VM：2M虚拟地址
VM：2M虚拟地址
压缩数据存储信息
压缩算法
4k Page
4k Page
4k Page
4k Page
4k Page
4k Page

图3

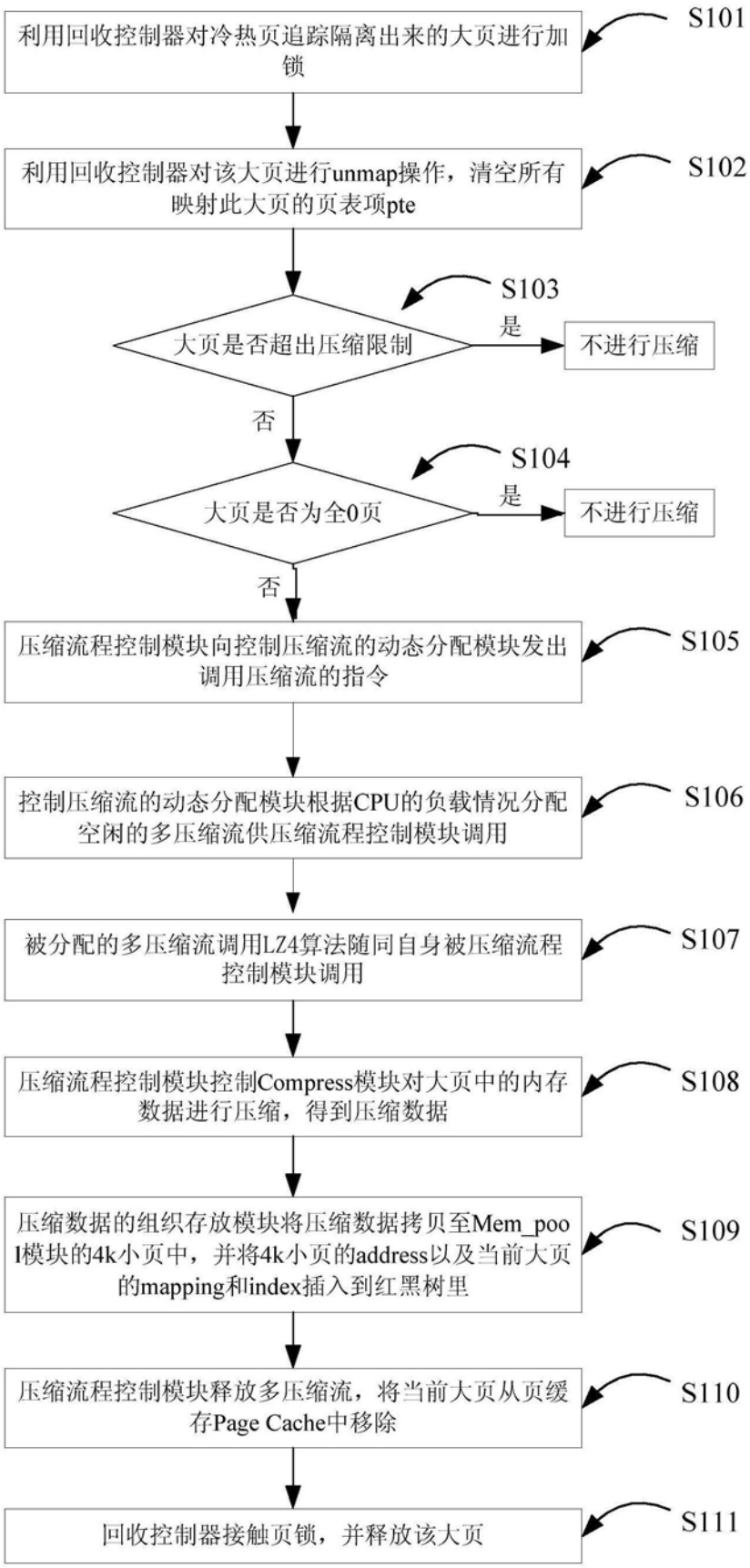

利用回收控制器对冷热页追踪隔离出来的大页进行加锁
S101
利用回收控制器对该大页进行unmap操作，清空所有映射此大页的页表项pte
S102
S103
大页是否超出压缩限制
是
不进行压缩
否
S104
大页是否为全0页
是
不进行压缩
否
压缩流程控制模块向控制压缩流的动态分配模块发出调用压缩流的指令
S105
控制压缩流的动态分配模块根据CPU的负载情况分配空闲的多压缩流供压缩流程控制模块调用
S106
被分配的多压缩流调用LZ4算法随同自身被压缩流程控制模块调用
S107
压缩流程控制模块控制Compress模块对大页中的内存数据进行压缩，得到压缩数据
S108
压缩数据的组织存放模块将压缩数据拷贝至Mem_pool模块的4k小页中，并将4k小页的address以及当前大页的mapping和index插入到红黑树里
S109
压缩流程控制模块释放多压缩流，将当前大页从页缓存Page Cache中移除
S110
回收控制器接触页锁，并释放该大页
S111

图4

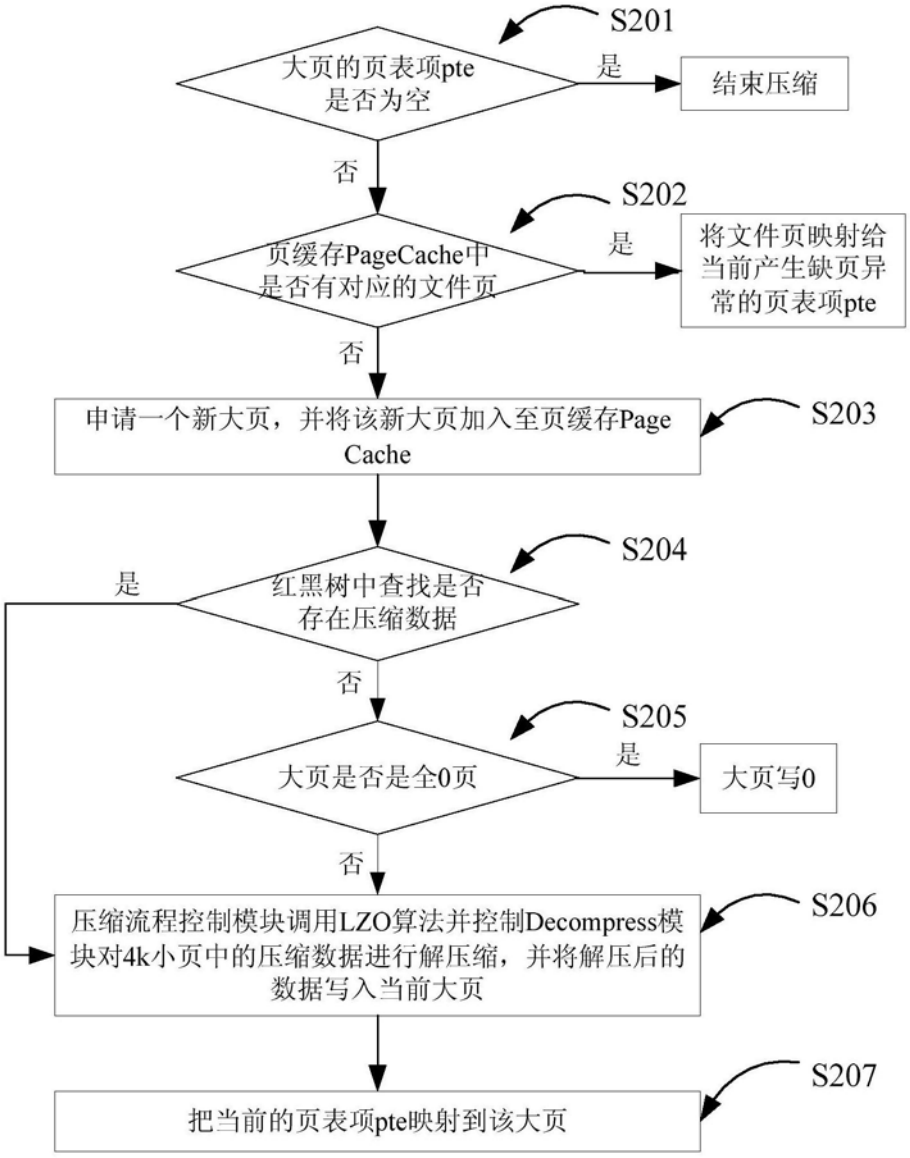

S201
大页的页表项pte是否为空
是
结束压缩
否
S202
页缓存PageCache中是否有对应的文件页
是
将文件页映射给当前产生缺页异常的页表项pte
否
申请一个新大页，并将该新大页加入至页缓存Page Cache
S203
S204
是
红黑树中查找是否存在压缩数据
否
S205
大页是否是全0页
是
大页写0
否
S206
压缩流程控制模块调用LZO算法并控制Decompress模块对4k小页中的压缩数据进行解压缩，并将解压后的数据写入当前大页
S207
把当前的页表项pte映射到该大页

图5