

说明书

内存页的处理方法、设备、装置以及存储介质

本申请要求于 2023 年 06 月 20 日提交的申请号为 202310740304.0、发明名称为“一种内存设置方法”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

本申请要求于 2023 年 08 月 25 日提交的申请号为 202311086477.1、发明名称为“内存页的处理方法、设备、装置以及存储介质”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及通信技术领域，特别涉及一种内存页的处理方法、设备、装置以及存储介质。

背景技术

为了缓解内存墙导致的性能瓶颈，在相关技术中，尝试从微架构层面的优化来弥补中央处理器（central processing unit, CPU）与内存之间的性能差距，例如，芯片微架构的设计尝试通过频繁模式压缩（frequent pattern compression, FPC）等缓存压缩技术，来减少片外通信的数据量，具体地，当 CPU 的缓存中的数据从缓存行（cache line）被淘汰到内存时，内存管理单元（memory management unit, MMU）按照缓存行中的数据所匹配的缓存压缩模式，对缓存行中的数据进行压缩，得到压缩数据，将压缩数据存储到内存，以实现将缓存行的数据淘汰到内存。

但是，缓存行中除了应用程序写入的数据以外还可能存在一些随机数值，导致缓存行中的数据不能很好的匹配数据压缩模式，没有匹配到数据压缩模式的数据在被淘汰出缓存行时无法以压缩的方式存储在内存的，降低了缓存向内存淘汰数据时缓存数据的压缩率。

发明内容

本申请实施例提供了一种内存页的处理方法、设备、装置以及存储介质，能够提高缓存数据的压缩率。该技术方案如下：

第一方面，提供了一种内存页的处理方法，该方法包括如下步骤：在应用程序所访问的虚拟地址（称为第一地址虚拟）引发缺页中断的情况下，为第一虚拟地址分配内存页（称为第一内存页）；之后，在第一内存页填充：缓存压缩模式对应数据序列中的数据（称为第一数据），其中，缓存压缩模式是将缓存中应用程序写入的数据迁移至内存页时所使用的数据压缩模式；然后，再基于第一虚拟地址，对填充后的第一内存页进行访问。

本方法通过在为引发缺页中断的虚拟地址分配完内存页后，预先在分配的内存页中填充第一数据，之后，再对填充第一数据的内存页进行访问，后续如果将内存页中的数据迁移到缓存，内存页中填充的第一数据也会迁移到缓存，如果在缓存中该内存页的数据淘汰到内存，淘汰的缓存数据也可能包括第一数据，由于第一数据为缓存压缩模式对应数据序列中的数据，当包括第一数据的缓存数据淘汰回内存时，能够增加缓存数据匹配缓存压缩模式的概率，进而能够提高缓存数据的压缩率。

在一种可能的实现方式中，上述在第一内存页填充第一数据的过程可以是：在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据，其中，内存预填充功能指示在应用程序访问空内存页之前在空内存页填充第一数据。

基于上述可能的实现方式，在应用程序开启内存预填充功能的情况下，才会为应用程序所分配到的内存页预填充第一数据，无需为运行的各个应用程序分配的内存页都预填充第一数据，减少了填充第一数据的工作量。

在一种可能的实现方式中，在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据之前，该方法还包括如下步骤：基于应用程序的运行情况，为应用程序周期性开启内存预填充功能。

在一种可能的实现方式中，在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据之前，该方法还包括如下步骤：在应用程序的类型是目标类型情况下，为应用程序开启内存预填充功能，目

标类型的应用程序适用缓存压缩模式。

基于上述可能的实现方式，在应用程序适用该缓存压缩模式的情况下，才会为该应用程序开启内存预填充功能，以便在该应用程序运行过程中，通过为该应用程序的内存页预填充第一数据，来提高该应用程序的数据从缓存迁移至内存时的压缩效率，且还能避免对不适用该缓存压缩模式的应用程序的内存页填充第一数据，减少了填充第一数据的工作量。

在一种可能的实现方式中，上述基于第一虚拟地址，对填充后的第一内存页进行访问包括如下步骤：先接收应用程序的第一写请求，第一写请求指示向第一虚拟地址写入第二数据，再将第一内存页中第一虚拟地址对应位置处的第一数据更新为第二数据。

基于上述可能的实现方式，第一数据为应用程序向内存页之前所填充的数据，后续应用程序在向内存页写入数据时，将第一数据更新为待写入数据即可，从而在不影响应用程序正常写入数据的情况下，通过预填充第一数据，来提高缓存压缩率。

在一种可能的实现方式中，多个第二内存页均用于存储应用程序的同一计算任务的任务数据，多个第二内存页均填充第一数据，该方法还包括如下步骤：接收应用程序的第二写请求，第二写请求指示向第二虚拟地址写入任务数据；将多个第二内存页中第二虚拟地址对应位置处的第一数据更新为任务数据。

基于上述可能的实现方式，对应同一计算任务的多个内存页均已填充第一数据，在向任一内存页写入任务数据时，可以将任一内存页中的第一数据更新为任务数据，即使该多个内存页中的任个内存页没有写入任务数据，因为第一数据的存在，也不会向该内存页写入随机数值，从而能够避免随机数值影响该内存页的缓存数据的压缩，从而能够提高缓存压缩率。

在一种可能的实现方式中，第一数据是该数据序列中的各个数据；或者，第一数据是该数据序列中出现频率最高的S个数据，S大于0；或者，第一数据是数据序列中连续的R个数据，R大于0。

基于上述可能的实现方式，在第一内存页填充该数据序列中的第一数据，后续如果将第一内存页的数据迁移到缓存，在从缓存将内存迁移第一内存页的缓存数据时，该缓存数据中仍可能有部分第一数据保留在缓存，以便缓存数据更容易与数据压缩模式匹配，以进一步提高缓存数据的压缩率。

在一种可能的实现方式中，第一数据是多个数据序列中出现频率最高的S个数据，该多个数据序列对应不同的缓存压缩模式，S大于0；或者，第一数据是该多个数据序列共有的数据。

基于上述可能的实现方式，在第一内存页填充多个数据序列中的第一数据，后续如果将第一内存页的数据迁移到缓存，在从缓存将内存迁移第一内存页的缓存数据时，该缓存数据中仍可能有部分第一数据保留在缓存，以便缓存数据更容易与这多个数据压缩模式匹配，增加了该缓存数据与数据压缩模式匹配到的概率，以进一步提高缓存数据的压缩率。

第二方面，提供了一种内存页的处理方法，该方法包括如下步骤：在获取到分配通告消息后，在第一内存页填充第一数据，其中，分配通告消息用于通告针对应用程序访问第一虚拟地址所引发的缺页中断，已为第一虚拟地址分配第一内存页，第一数据是缓存压缩模式对应数据序列中的数据，缓存压缩模式是将缓存中应用程序写入的数据迁移至内存页时所使用的的数据压缩模式。

在一种可能的实现方式中，在第一内存页填充第一数据包括如下步骤：在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据，内存预填充功能指示在应用程序访问空内存页之前在空内存页填充第一数据。

在一种可能的实现方式中，在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据之前，该方法还包括如下步骤：基于应用程序的运行情况，为应用程序周期性开启内存预填充功能。

在一种可能的实现方式中，在应用程序开启内存预填充功能的情况下，在第一内存页填充第一数据之前，该方法还包括如下步骤：在应用程序的类型是目标类型情况下，为应用程序开启内存预填充功能，目标类型的应用程序适用缓存压缩模式。

在一种可能的实现方式中，第一数据是该数据序列中的各个数据；或者，第一数据是该数据序列中出现频率最高的S个数据，S大于0；或者，第一数据是数据序列中连续的R个数据，R大于0。

在一种可能的实现方式中，第一数据是多个数据序列中出现频率最高的S个数据，该多个数据序列对应不同的缓存压缩模式，S大于0；或者，第一数据是该多个数据序列共有的数据。

第三方面，提供了一种内存页的处理设备，该内存页的处理设备包括用于执行上述第一方面或上述第一方面的任一种可选方式提供的内存页的处理方法的功能模块。

第四方面，提供了一种内存填充装置，该内存填充装置包括用于执行上述第二方面或上述第二方面的任一种可选方式提供的内存页的处理方法的功能模块。

第五方面，提供一种电子设备，该电子设备包括处理器，所述处理器用于执行程序代码，使得电子设备执行以实现如上述第一方面或上述第一方面的任一种可选方式提供的内存页的处理方法所执行的操作。

第六方面，提供一种内存填充装置，该内存填充装置包括处理器，所述处理器用于执行程序代码，使得内存填充装置执行以实现如上述第二方面或上述第二方面的任一种可选方式提供的内存页的处理方法所执行的操作。

第七方面，提供一种计算机可读存储介质，该存储介质中存储有至少一条程序代码，该程序代码由处理器读取以使电子设备执行如上述第一方面或上述第一方面的任一种可选方式提供的内存页的处理方法所执行的操作。

第八方面，提供一种计算机可读存储介质，该存储介质中存储有至少一条程序代码，该程序代码由处理器读取以使内存填充装置执行如上述第二方面或上述第二方面的任一种可选方式提供的内存页的处理方法所执行的操作。

第九方面，提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括程序代码，该程序代码存储在计算机可读存储介质中，电子设备的处理器从计算机可读存储介质读取该程序代码，处理器执行该程序代码，使得该电子设备执行上述第一方面或者第一方面的各种可选实现方式中提供的方法。

第十方面，提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括程序代码，该程序代码存储在计算机可读存储介质中，内存填充装置的处理器从计算机可读存储介质读取该程序代码，处理器执行该程序代码，使得该内存填充装置执行上述第二方面或者第二方面的各种可选实现方式中提供的方法。

本申请在上述各方面提供的实现方式的基础上，还可以进行进一步组合以提供更多实现方式。

附图说明

图1是本申请实施例提供的一种应用内存页的处理方法的电子设备架构图；

图2是本申请实施例提供的一种内存页的处理方法流程图；

图3是本申请实施例提供的另一种应用内存页的处理方法的电子设备架构图；

图4是本申请实施例提供的另一种内存页的处理方法流程图；

图5是本申请实施例提供的一种应用程序访问内存的示意图；

图6是本申请实施例提供的一种内存页的处理设备的结构示意图；

图7是本申请实施例提供的一种内存填充装置的结构示意图；

图8是本申请实施例提供的另一种内存填充装置的结构示意图。

具体实施方式

为了便于理解本申请的具体实施方式，该具体实施方式中涉及的部分名词进行如下介绍。

内存页：将内存空间划分成一个个大小相等的分区，每个分区为一个内存页，每个内存页用一个物理页号来指代，不同内存页的物理页号不同，内存页也称为物理块或页框。

虚拟页：将应用程序的虚拟地址空间划分为一个个大小相等的区域，每个区域的大小与内存页的大小相等，每个区域为一个虚拟页，每个虚拟页用一个虚拟页号来指代，不同虚拟页的虚拟页号不同，虚拟页也称为页或者页面。

页表：包括多个页表条目（page table entry，PTE），每个PTE包括一个虚拟页的虚拟页号、一个物理页的物理页号以及有效标识，以指示将该虚拟页映射到该内存页（即是否将虚拟页调入内存页），有效标识的取值用于指示该虚拟页与物理页之间的映射关系是否有效（即指示PTE是否有效），示例性地，有效标识取值为1，指示该映射关系有效，该虚拟页已经调入到该内存页，有效标识取值为0，指示该映射关系无效，该虚拟页没有调入到该内存页。

缺页中断：应用程序在访问某一虚拟页时，内存管理单元（memory management unit，MMU）通过查询页表来确定所要访问的虚拟页是否已经调入到内存页。每当该虚拟页没有调入到内存页时，会产生一次缺页中断，操作系统会从该应用程序申请到的内存中，为该虚拟页分配内存页，将虚拟页调入到分配的内

存页，以解决缺页中断。

虚拟地址（virtual address, VA）：是虚拟地址空间中的地址，虚拟地址空间包括多个虚拟地址，虚拟地址包括一个虚拟页的虚拟页号以及页内偏移量，该页内偏移量指示该虚拟地址在该虚拟页的地址偏移。

物理地址（physical address, PA）：是物理地址空间中的地址，物理地址空间即内存提供的空间，物理地址空间包括多个物理地址，物理地址包括一个物理页的物理页号以及页内偏移量，该页内偏移量指示该物理地址在该内存页的地址偏移。

缓存压缩算法：将缓存（cache）中的数据（称为缓存数据）迁移至内存页时所使用的数据压缩算法，缓存压缩算法定义至少一个缓存压缩模式、每个缓存压缩模式的数据压缩方式以及每个缓存压缩模式所要求的数据序列，不同缓存压缩模式所要求的数据序列不同。缓存压缩模式是将缓存数据迁移至内存页时所使用的数据压缩模式，当待压缩缓存数据与任一缓存压缩模式所要求的数据序列相同（即匹配）时，可以使用于该任一缓存压缩模式的缓存压缩方案，对待压缩缓存数据进行压缩，如果待压缩缓存数据与该至少一个缓存压缩模式所要求的数据序列都不相同，则不能对待压缩缓存数据进行数据压缩。任一缓存压缩模式所要求的数据序列包括 N ($N > 1$) 个数据，该 N 个数据是同一 N 进制中的 N 个数值，因此，该数据序列也可以称为数值序列。以十六进制为例，该 N 个数据中的每个数据为十六进制的数值 0 至数值 F 中的一个数值，再以十进制为例，每个数据为十进制的数值 0 至数值 9 中的一个数值。该 N 个数据可能是同一数值，也可能是不同的数值。 N 个数据中的每个数据以二进制数据形式存储，在存储时，每个数据由 M ($M > 0$) 个比特（bit）的二进制数值（0/1）表示，以 $N=16$, $M=4$ 为例，十六进制的一个数据由 4bit 的二进制数值表示，在存储时，该数据序列的长度为 $M \times N \text{bit}$ 。

FPC：是基于缓存行（cache line）压缩/解压缩的一种缓存压缩算法。每个高速缓存行被划分为 32 位字（word），例如，64 字节的缓存行为 16 个字，每个 32 位的字被编码为 3 位前缀（pre-fix）加上数据。具体地，FPC 定义了零运行（zero run）模式、4 位符号扩展（4-bit sign-extended）模式、1 字节符号扩展（1-byte sign extended）模式、半字节符号扩展（halfword sign-extend）模式、用 0 半字填充的半字（halfword padded with a zero halfword）模式、两个半字且每个字节都有符号扩展（two halfword, each a byte sign-extended）模式以及由重复字节组成的字（word consisting of repeated bytes）模式等 7 个缓存压缩模式，每个缓存压缩模式对应一个 3 位前缀（pre-fix），pre-fix 即对应的数据压缩方式。以零运行模式为例，零运行对应的 pre-fix 为 000，零运行模式要求的数据序列为由数值 0 表示的一个 32 位的字，如果待压缩的缓存数据为一个 32 位的字，且缓存数据为 0x0000000000000000，则该缓存数据与零运行模式匹配，采用零运行模式对该缓存数据进行压缩，得到压缩数据 0x0，即压缩数据是 0 位数据+3 位 pre-fix。

第一数据：在应用程序向空内存页读写数据之前，内存填充装置预先在空内存页填充的数据，空内存页是指没有写入任何数据的内存页。

内存预填充功能：指示在应用程序访问空内存页之前在空内存页填充第一数据。

接下来，结合附图对本申请提供的内存页的处理方法的应用环境进行介绍。

图 1 是本申请实施例提供的一种应用内存页的处理方法的电子设备架构图，图 1 所示的电子设备 100 可以为终端、服务器或者网络设备，在此，本申请实施例对电子设备 100 的设备类型不做限定。

如图 1 所示，电子设备 100 包括处理器核心（core）101、性能监控单元（performance monitoring unit, PMU）102、地址变换高速缓存（translation lookaside buffer, TLB）103、MMU104、内存填充装置 105、内存 106 以及总线 107。其中，处理器核心 101、PMU102、MMU104 以及 TLB103 都集成在中央处理器（central processing unit, CPU），CPU 包括一个或多个处理器核心 101，该 CPU 还可以包括缓存（cache），缓存用于缓存内存 106 中存储的数据，CPU 中的缓存还可以是多级缓存，比如一级缓存至三级缓存，或者一级缓存和二级缓存，在此，本申请实施例对 CPU 中缓存的个数以及缓存的级别不做限定，MMU104 通过总线 107 与内存 106 进行通信，内存 106 也称为主存（main memory）。

处理器核心 101 用于运行应用程序，一个处理器核心 101 可以运行一个或者多个应用程序，在处理器核心 101 应用程序运行过程中，应用程序对虚拟页进行读写访问（即读访问或写访问），处理器核心 101 通过 MMU104，对内存 106 中虚拟页对应的内存页进行读写访问，以完成对虚拟页的读写访问，其中，读访问是指在页面读数据，写访问是指在页面写数据。

PMU102 是 CPU 内嵌的性能监控组件，用于统计 CPU 中发生的特定硬件事件（称为 PMU 事件），比如与处理器核心有关的事件、与缓存有关的事件以及与 TLB 有关的事件等，其中，与处理器核心有关的事

件比如执行的指令个数、捕获异常、时钟周期、分支预测错误等，与缓存有关的事件比如各级缓存的访问次数、缓存未命中（cache miss）等，与 TLB103 有关的事件比如 TLB 未命中（cache miss）。

TLB103 用于缓存页表中的页表条目，页表存储在内存 106。MMU104 是用于管理内存 106 的组件，MMU104 也称为内存管理装置。MMU104 能够将应用程序访问的虚拟地址翻译为内存 106 中的物理地址，对物理地址进行访问。例如，处理器核心 101 在运行应用程序的过程中，向 MMU104 下发应用程序所要访问的虚拟地址，MMU104 在 TLB103 缓存的页表条目中查询该虚拟地址中的虚拟页号，如果查询到包括该虚拟页号的 PTE，则发生 TLB 命中（hit）。如果在 TLB103 中没有查询到包括该虚拟页号的 PTE，则发生 TLB 未命中，MMU104 从内存 106 中的页表查询包括该虚拟页号的 PTE，将查询到的 PTE 中物理页号以及该虚拟地址中的地址偏移量组成物理地址，实现虚拟地址到物理地址的转换，对内存 106 中该物理地址所指示的位置进行读写访问，以实现即对虚拟地址的读写访问。在应用程序运行过程中，MMU104 将内存 106 中应用程序频繁访问的数据迁移至 CPU 中的最后一级缓存（last level cache, LLC），当 LLC 中的空间不足时，MMC 可以将 LLC 中的至少一个缓存行中的缓存数据淘汰回内存 106，在向内存 106 淘汰缓存数据过程中，可以使用缓存压缩算法，将缓存数据迁移到内存 106。

当应用程序访问的虚拟页没有调入到内存页时，触发缺页中断，针对缺页中断，操作系统将该虚拟页调入到内存 106 的某一内存页，调入的内存页是没有填充数据的空内存页，针对这种空内存页，内存填充装置 105 用于应用程序访问调入的虚拟页之前，在该内存页预填充第一数据，这样后续应用程序在向该虚拟页写入数据时，可以将该内存页中待写入位置处的第一数据更新为待写入数据。可选地，内存填充装置 105 在向内存页填充第一数据之前，还可以先判断该内存页所属的应用程序是否开启了内存预填充功能，如果开启了内存预填充功能，则在该内存页填充第一数据，否则，不在该内存页填充第一数据。示例性地，内存填充装置 105 可以根据应用程序运行过程中 PMU102 监控到的 PMU 事件，为应用程序开启内存预填充功能，该过程后续结合方法实施例作详细介绍。

内存填充装置 105 可以通过软件实现、硬件实现或者软硬件结合的方式实现。内存填充装置 105 作为软件功能单元的一种举例，内存填充装置 105 包括程序代码，处理器核心 101 通过运行该程序代码实现内存填充装置 105 的功能。

内存填充装置 105 作为硬件功能单元的一种举例，内存填充装置 105 可能是利用专用集成电路（application-specific integrated circuit, ASIC）实现或可编程逻辑器件（programmable logic device, PLD）实现的设备等。其中，上述 PLD 可以是复杂程序逻辑器件（complex programmable logical device, CPLD）、现场可编程门阵列（field-programmable gate array, FPGA）、通用阵列逻辑（generic array logic, GAL）或其任意组合实现。

在内存填充装置 105 作为硬件功能单元的情况下，内存填充装置 105 集成在 MMU104 中，或者独立于 MMU104 之外，例如，内存填充装置 105 位于 CPU 中 MMU104 之外的位置，或者位于电子设备 100 所在电子设备中除 CPU 之外的位置，在此，本申请实施例对内存填充装置 105 的位置不做限定。

可选地，如图 1 所示，内存填充装置 105 包括判断单元 51 以及填充单元 52，预设判断单元 51 用于判断内存页所属应用程序是否开启内存预填充功能，还用于根据应用程序运行过程中 PMU102 监控到的 PMU 事件，为应用程序开启内存预填充功能。填充单元 52 用于根据判断单元 51 的判断结果，向内存页预填充第一数据，或者，不向内存页预填充第一数据。对于判断单元 51 以及填充单元 52 中的任一子单元，该任一子单元可以通过软件实现、硬件实现或者软硬件结合的方式实现，以判断单元以及填充单元 52 均通过硬件实现为例，如图 1 所示，填充单元 52 集成在 MMU104，判断单元 51 独立于 MMU104 之外，在另一些实施例中，填充单元 52 和判断单元 51 都集成在 MMU104 或者都独立于 MMU104 之外。

接下来，结合图 2，对内存填充装置在内存页预填充第一数据的过程进行详细介绍。

图 2 是本申请实施例提供的一种内存页的处理方法流程图，该方法应用于电子设备，该电子设备如电子设备 100，该方法由电子设备中的操作系统、内存填充装置以及 MMU 通过交互来实现，该方法包括如下步骤。

201、在应用程序所访问的第一虚拟地址引发缺页中断的情况下，操作系统为第一虚拟地址分配第一内存页。

其中，该应用程序为该电子设备的任一 CPU 中的任一处理器核心运行的任一应用程序，该应用程序对应一个虚拟地址空间，第一虚拟地址是该虚拟地址空间中的任一虚拟地址，为便于描述，将第一虚拟地

址中的虚拟页号称为第一虚拟页号，第一虚拟页号所指示的虚拟页称为第一虚拟页，第一虚拟页即应用程序所访问的虚拟页，为第一虚拟地址分配内存页，即为第一虚拟页分配内存页，本步骤 201 也即是在应用程序所访问的第一虚拟页引发缺页中断的情况下，操作系统为第一虚拟页分配内存页，第一内存页是操作系统为第一虚拟页分配的内存页。

在一种可能的实现方式中，电子设备中的处理核心在运行该应用程序的过程中，如果因访问第一虚拟地址引发缺页中断，处理核心通知操作系统发生缺页中断，由操作系统为第一虚拟页分配内存页，将第一虚拟页调入到内存页，以结束缺页中断。接下来，结合图 3 所示的电子设备架构图以及图 4 所示的另一种内存页的处理方法流程图，对缺页中断的触发过程以及处理流程进行如下介绍：

步骤 A1、处理器核心在运行应用程序的过程中，向 MMU 发送第一读写请求，第一读写请求包括第一虚拟地址，以指示对第一虚拟地址进行读写数据。

其中，对第一虚拟地址进行读写数据即对第一虚拟地址进行读写访问，第一读写请求为写请求或者读请求，在第一读写请求为写请求的情况下，第一读写请求还包括待写入的数据（称为第二数据），第一读写请求指示向第一虚拟地址写入第二数据，在第一读写请求为读请求的情况下，第一读写请求指示读取第一虚拟地址处存储的数据。

示例性地，处理器核心在执行应用程序中的任一指令（称为第一指令）时，如果有访问第一虚拟地址的需求，处理器核心向 MMU 发送第一读写请求。

步骤 A2、MMU 接收该第一读写请求，基于第一读写请求，查询第一虚拟地址对应的 PTE（称为第一 PTE），第一 PTE 是页表中包括第一虚拟页号的 PTE。

如图 3 所示，MMU 先请求 TLB 查询第一 PTE，如果 TLB 在本地缓存的 PTE 中查询到第一 PTE，TLB 将第一 PTE 返回给 MMU。如果没有在 TLB 查询到第一 PTE，MMU 请求内存存在页表查询第一 PTE，如果内存存在页表中查询到第一 PTE，内存向 MMU 返回第一 PTE，这样 MMU 可以从 TLB 或者内存中查询到第一 PTE。

如图 3 所示，如果 TLB 没有缓存第一 PTE 且页表包括第一 PTE，随着应用程序的运行，MMU 还可以将页表中的第一 PTE 缓存到 TLB，以便后续应用程序再次访问第一虚拟页时可以在 TLB 查询到第一 PTE。在第一 PTE 缓存到 TLB 后，随着应用程序的运行，TLB 还可能将第一 PTE 淘汰回页表。

在获取到第一 PTE 后，在第一 PTE 中的有效标识不是 0 的情况下，比如第一 PTE 中的有效标识为 1，触发 MMU 执行下述步骤 A3 和步骤 A4，在第一 PTE 中的有效标识是 0 的情况下，MMU 跳转执行下述步骤 A5。

步骤 A3、在第一 PTE 中的有效标识不是 0 的情况下，MMU 基于第一 PTE 以及第一虚拟地址，构造待访问的第一物理地址，对内存中的第一物理地址进行访问。

其中，第一物理地址是第一虚拟地址对应的物理地址，即第一虚拟地址转换的物理地址，通过对第一物理地址进行访问，以实现访问第一虚拟地址。

例如，MMU 将第一 PTE 中的内存页号（称为第一内存页号）与第一虚拟地址中的页内偏移量进行组合，得到第一物理地址，在第一读写请求为读请求的情况下，MMU 读取第一内存页中第一物理地址处存储的数据（称为第四数据），以实现访问第一物理地址，在内存访问请求为写请求的情况下，向第一内存页中第一物理地址处写入第二数据，以实现访问第一物理地址。

步骤 A4、MMU 向处理器核心发送读写完成响应，该读写完成响应指示对第一虚拟地址读写数据完成，此时访问第一虚拟地址的流程结束。

其中，在第一读写请求为读请求的情况下，该读写完成响应即读完成响应，指示已经在第一虚拟地址读取到数据，此时该读写完成响应还包括从第一物理地址读取的第四数据，在第一读写请求为写请求的情况下，该读写完成响应即写完成响应，指示已经在第一虚拟地址写入第二数据。

步骤 A5、在第一 PTE 中的有效标识是 0 的情况下，MMU 向处理器核心发送读写失败响应，该读写失败响应指示因没有对应的内存页导致读写第一虚拟地址失败。

步骤 A6、处理器核心基于该读写失败响应，将第一指令挂起，触发缺页中断，向操作系统发送缺页中断通知消息，该缺页中断通知消息指示第一虚拟页时发生缺页中断。

其中，该缺页中断通知消息包括第一虚拟页号。

步骤 A7、操作系统接收该缺页中断通知消息基于该缺页中断通知消息，为第一虚拟页分配第一内存

页，以进行缺页异常处理。

如图4所示，操作系统在接收到缺页中断通知消息后，从应用程序分配到的内存页中查询是否有未被占用的内存页，其中，应用程序分配到的内存页是应用程序在申请内存时操作系统为应用程序分配的内存空间中的内存页，应用程序分配到的内存页有多个，未被占用的内存页是指没有调入虚拟页的内存页，即没有与页表中与虚拟页建立有效映射关系的内存页。

如果应用程序分配到的内存页中有未被占用的内存页，操作系统将未被占用的任一内存页分配给第一虚拟页。示例性地，如图3所示，操作系统向MMU发送第一内存分配请求，其中，第一内存分配请求包括第一虚拟页号以及该内存页的物理页号，以指示将该内存页分配给第一虚拟页。在接收到第一内存分配请求后，MMU查询页表中是否存在包括该物理页号的有效PTE，由于该内存页还未被占用，则MMU查询不到包括该物理页号的有效PTE，MMU继续查询页表中的第一PTE，如图4所示，MMU将该内存页的物理页号填充至第一PTE，将第一PTE中的有效标识置1，使得第一PTE有效，以指示将该内存页分配给第一虚拟页，即将该第一虚拟页调入到该任一内存页，此时，该内存页即第一内存页。

如果应用程序分配到的内存页中没有未被占用的内存页，此时，应用程序分配到的内存页都已经被占用，操作系统按照内存页的换入换出机制，从应用程序被占用的内存页中确定牺牲页，将牺牲页分配给第一虚拟页，其中，牺牲页即待分配给第一虚拟页的内存页。示例性地，如图3所示，操作系统向MMU发送第二内存分配请求，该第二内存分配请求包括第一虚拟页号以及牺牲页的物理页号，以指示将牺牲页分配给第一虚拟页。在接收到第二内存分配请求后，MMU查询页表中是否存在包括该物理页号的有效PTE，将查询到有效PTE称为第二PTE，查询到第二PTE说明牺牲页已经分配给除第一虚拟页以外的其他虚拟页，为避免该其他虚拟页的数据丢失，如图3和图4所示，MMU查询该牺牲页中是否存储有数据，如果该牺牲页中存储有数据，如将牺牲页中的数据写入到电子设备的磁盘(disk)，使得牺牲页变成空内存页。如果牺牲页中没有存储数据，牺牲页即是空内存页。在牺牲页是空内存页的情况下，MMU将第二PTE中有效标识置0，以无效掉第二PTE，在页表中查询第一PTE，将牺牲页的物理页号填充至第一PTE，将第一PTE中的有效标识置1，使得第一PTE有效，以指示将该牺牲页分配给第一虚拟页，即将该第一虚拟页调入到牺牲页，此时，该牺牲页即第一内存页。

上述是应用程序访问第一虚拟页时页表包括无效的第一PTE为例进行说明的，在另一些实施例，应用程序访问第一虚拟页时，MMU在页表可能查询不到第一PTE，在此情况下，MMU仍然向处理器核心发送读写失败响应，触发缺页中断，进而触发操作系统为第一虚拟页分配第一内存页，在此情况下，当MMU确定出待分配给第一虚拟页的第一内存页（如牺牲页或未被占用的任一内存页）后，MMU基于第一虚拟页号以及第一内存页的物理页号（称为第一物理页号），在页表中创建有效的第一PTE。

如图3所示，在第一PTE有效后，MMU向内存填充装置发送分配通告消息，该分配通告消息用于通告已将第一内存页分配给第一虚拟页，即用于通告针对应用程序访问第一虚拟地址所引发的缺页中断，已为所述第一虚拟地址分配第一内存页。内存分配单元在接收到该分配通告消息后，基于该分配通告消息执行下述步骤202。其中，该分配通告消息包括第一虚拟页号和第一物理页号。

202、内存填充装置在第一内存页填充第一数据，该第一数据是缓存压缩模式对应数据序列中的数据，该缓存压缩模式是将缓存中应用程序写入的数据迁移至内存页时所使用的数据压缩模式。

其中，该缓存压缩模式是MMU所使用的缓存压缩算法中定义的缓存压缩模式，例如，该缓存压缩算法为FPC或其他依赖缓存压缩模式的缓存压缩技术。该缓存压缩算法定义了至少一个缓存压缩模式、每个缓存压缩模式对应的数据序列以及数据压缩方式，每个缓存压缩模式对应的数据序列即每个缓存压缩模式对待压缩数据所要求的数据序列。

第一数据是该至少一个缓存压缩模式中任一缓存压缩模式对应数据序列方式中的数据，示例性地，第一数据是该数据序列中的全部数据（即是该数据序列中的各个数据），或者，第一数据是该数据序列中的部分数据。以第一数据是该数据序列中的部分数据为例，假设该数据序列包括N个数据，第一数据是该数据序列中出现频率最高的S个数据，其中，N大于1，S大于0且小于N。比如S=1，以缓存压缩模式为FPC中的零运行模式为例，十六进制的数值0为零运行模式对应的数据序列中出现频率最高的数据，再以缓存压缩模式为FPC中的1字节符号扩展模式为例，1字节符号扩展模式中出现频率最高的数据为十六进制的数值F。再比如S=2，数据序列为0000FFFF，第一数据为0和F。或者，第一数据是该数据序列中连续的R个数据，比如数据序列中前R个数据为第一数据，或者，中间R个数据为第一数据，或者，后R个数据

为第一数据，其中，R 大于 0 且小于 N，R 与 S 可能相同也可能不同，在此，本申请实施例对 R 和 S 的取值不作限定。

在另一些实施例中，第一数据是该缓存压缩算法定义的多个数据压缩模式所要求的多个数据序列中的数据，比如，第一数据是该多个数据序列中出现频率最高的 S 个数据，或者，第一数据是多个数据序列共有的数据。

如图 3 所示，在接收到 MMU 的分配通告消息后，内存填充装置基于分配通告消息中的第一虚拟页号和第一物理页号，获悉 MMU 已将第一内存页分配给第一虚拟页，此时，第一内存页为空闲内存页，内存填充装置在第一内存页填充第一数据。比如，将第一内存页填充第一数据，或者，将第一内存页的部分内存空间填充上第一数据，剩余部分空间不填充第一数据（即不将第一内存页填充第一数据）。

以在第一内存页填充第一数据为例，假设第一内存页的大小为 64 字节（byte）， $64\text{byte}=16*8*4\text{bit}$ ，如果存储时第一数据的大小为 4bit，如表 1 所示，内存填充装置在第一内存页中每隔 4byte（即 $8*4\text{bit}$ ）填充 8 个第一数据，以将第一内存页填充第一数据。这样后续应用程序再向第一虚拟页写入数据时，MMU 将第一内存页中待写入位置处的第一数据更新为待写入数据即可。

表 1

字节数	4 byte	4 byte	4 byte	4 byte
第一内存页	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据

如图 3 所示，在第一内存页填充第一数据后，内存填充装置向 MMU 发送填充完成响应，该填充完成响应用于指示已经在第一内存页填充第一数据。MMU 接收到填充完成后响应后，向操作系统发送分配完成响应，该分配完成响应指示已经将第一内存页分配给第一虚拟页。操作系统接收到分配完成响应后，向处理器核心发送缺页中断处理完成响应，以指示已经处理完成第一虚拟页引发的缺页中断。处理器核心接收到该中断处理完成响应后，结束缺页中断处理流程，释放被挂起的第一指令，再次执行第一指令，因此，对于应用程序而言，仅能感知发生缺页中断，感知不到第一内存页预填充第一数据的动作。

处理器核心再次执行第一指令时，向 MMU 发送第一读写请求。MMU 再次接收第一读写请求后，基于第一读写请求中的第一虚拟地址，对第一内存页进行访问（即下述步骤 203）。

203、内存管理单元基于该第一虚拟地址，对填充后的该第一内存页进行访问。

示例性地，如果再次接收到的第一读写请求为写请求，为了和后文中的其他写请求区分开，将第一读写请求称为第一写请求，MMU 再次执行步骤 A2，查询第一虚拟地址对应的第一 PTE，此时的第一 PTE 中的有效标识为 1，即第一 PTE 为有效 PTE，则 MMU 基于第一 PTE 以及第一虚拟地址构造第一物理地址，向第一内存页中第一物理地址所指示的位置（称为第一位置）写入第二数据，以实现向第一内存页中第一虚拟地址的对应位置写入数据，此时，第一位置可能已经填充第一数据，也可能没有填充第一数据，如果第一没有填充第一数据，则 MMU 在第一位置写入第二数据，以完成对第一物理地址和第一虚拟地址的访问，如果第一位置已经预填充第一数据，在向第一位置写入第二数据时，将第一位置处的第一数据更新为第二数据，其中，第二数据所占用的存储空间大于或等于第一数据所占用的存储空间。示例性地，MMU 以覆盖写的形式，向第一内存页中第一物理地址所指示的第一位置写入第二数据，使得第一位置处的第一数据更新为第二数据，以完成对第一物理地址和第一虚拟地址的访问。

其中，向第一位置写入数据是指从该第一位置开始写入数据。仍以上述的表 1 为例，假设第一物理地址中的页内偏移量为 0，第一物理地址所指示的第一位置即是第一内存页的起始位置，如果第二数据的大小为 4byte，如表 2 所示，MMU 将第一内存页中第 1 个 4byte 中存储的第一数据更新为第二数据。

表 2

字节数	4 byte	4 byte	4 byte	4 byte
第一内存页	第二数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据
	8 个第一数据	8 个第一数据	8 个第一数据	8 个第一数据

假设第一读写请求为读请求，由于此时第一内存页中预填充的是第一数据，并不是应用程序要读取的数据，应用程序要读取的数据还没有加载至第一内存页，如图3所示，为了让应用程序读取到所需的第四数据，MMU 先将磁盘存储的第四数据写入到第一内存页的第一物理地址，以将第一物理地址处的第一数据更新为第四数据，之后，应用程序再从第一物理地址读取到第四数据，以完成对第一物理地址和第一虚拟地址的访问。

在完成对第一物理地址的访问后，也就实现本次对第一内存页的访问，MMU 向处理器核心返回读写完成响应（即上述步骤 A4）。

在处理器核心接收到该读写完成响应后，处理器核心继续执行第一指令或者执行应用程序的除第一指令之外的其他指令（称为第二指令），在继续执行第一指令或者第二指令的过程中，应用程序会访问第一虚拟页或者除第一虚拟页以外的其他虚拟页，在访问第一虚拟页的情况下，应用程序可能会访问第一虚拟页的除第一虚拟地址以外的其他虚拟地址，或者，再次访问第一虚拟地址，访问第一虚拟页的其他虚拟地址的过程可参考访问第一虚拟地址的过程，在此不再赘述。访问其他虚拟页的过程可参考访问第一虚拟页的过程中，在此不再赘述。

在处理器核心运行应用程序的过程中，应用程序可能多次访问第一虚拟页，针对应用程序对第一虚拟页的任一虚拟地址的写访问，MMU 将该第一内存页中该任一虚拟地址的对应位置处的第一数据更新为待写入数据。如果在运行过程中应用程序频繁访问第一虚拟页的至少一个虚拟地址，MMU 将第一内存页的该至少一个虚拟地址所在目标内存空间中的数据迁移到缓存（如 LLC）中的某个缓存行（称为第一缓存行），其中，该至少一个虚拟地址可能包括第一虚拟地址，也可能不包括第一虚拟地址，目标内存空间的大小与缓存行的大小相同，目标内存空间的大小可能与内存页的大小相同，也可能与内存页的大小不同，在目标内存空间的大小与内存页的大小相同的情况下，该目标内存空间即为第一内存页，第一内存页中存储的所有数据都要迁移至第一缓存行。

为便于描述，将目标内存空间中的数据（即迁移至缓存的数据）称为第一迁移数据，随着应用程序对该目标内存空间的访问，该第一迁移数据包括第一数据和/或应用程序写入的数据。以表2为例，假设缓存行的大小与内存页的大小相同，均为64字节，应用程序频繁访问表2所示的第一内存页，则第一内存页为目标内存空间，第一内存页中存储的第二数据以及各个第一数据组成第一迁移数据，MMU 将第一迁移数据迁移至缓存中的第一缓存行，将页表中的第一PTE缓存到TLB。

在将第一迁移数据迁移至第一缓存行后，应用程序对第一虚拟页进行读访问时，MMU 将第一缓存行中的待读取数据返回为应用程序，以实现对第一内存页和第一虚拟页的读访问，在应用程序对第一虚拟页进行写访问时，MMU 通过在第一缓存行写入数据，以实现对第一内存页和第一虚拟页的写访问，此时第一迁移数据发生更新，将更新后的第一迁移数据称为第二迁移数据。

仍以应用程序访问第一虚拟地址为例，处理器核心向MMU 下发指示第一读写请求，基于第一读写请求中的第一虚拟地址，MMU 可以从TLB 中查询到第一PTE，根据第一PTE 以及第一虚拟地址构造出第一物理地址，对第一缓存行中第一物理地址对应的位置进行读访问或写访问。以写访问为例，将待写入数据称为第五数据，MMU 以覆盖写的形式，向第一缓存行中第一物理地址对应的位置（称为第二位置）写入第五数据，使得第二位置处的数据更新为第五数据，进而使得第一缓存行中的第一迁移数据更新为第二迁移数据，第二位置处的数据可能是第一数据，也可能不是第一数据。仍以第一迁移数据为表2中示出的数据为例，假设第二位置是第一内存页的第9byte 的第一bite，如果第五数据为4byte，如表3所示，向第一缓存行写入第五数据后，第五数据可以覆盖掉第一缓存行中第一内存页的第9byte 至12byte 的8个第一数据。假设第二位置是第一内存页的第13byte 的第一bite，如果第五数据为2字节，如表3所示，向第一缓存行中写入第五数据后，第五数据可以覆盖掉第一缓存行中第一内存页的第9byte 和第10byte 的4个第一数据。

表3

字节数	4 byte	4 byte	4 byte	4 byte
第一缓存行	第二数据	8个第一数据	第五数据	8个第一数据
	第五数据+4个第一数据	8个第一数据	8个第一数据	8个第一数据
	8个第一数据	8个第一数据	8个第一数据	8个第一数据

后续，在第一缓存行中的缓存数据（第一迁移数据或第二迁移数据）满足淘汰条件的情况下，MMU 将

该缓存数据迁移到第一内存页。其中，该淘汰条件为将缓存行中的缓存数据淘汰到内存的条件，假设缓存压缩模式所要求的数据序列长度为 $M \times N$ bit，第一缓存行的存储空间大小为 K bit，其中， K 大于 $M \times N$ ，则第一缓存行中存储有 $K/(M \times N)$ 个待迁移数据。仍以表 3 为例， $K=64$ 字节， $M \times N=4$ 字节，则第一缓存行中存储有 16 个待迁移数据，如第二数据为一个待迁移数据、8 个第一数据为一个待迁移数据、4byte 的第五数据为一个待迁移数据，2byte 的第五数据和 4 个第一数据为一个待迁移数据。

在向第一内存页迁移第一缓存行中的缓存数据时，针对每个待迁移数据，MMU 将该待迁移数据与至少一个缓存压缩模式所要求的数据序列进行匹配，如果该待迁移数据与任一缓存压缩模式所要求的数据序列相同，则该待迁移数据与该数据序列匹配，与该任一缓存压缩模式匹配，否则，该待迁移数据与该任一缓存压缩模式不匹配。如该待迁移数据与该任一缓存压缩模式匹配，MMU 采用该任一缓存压缩模式对应的数据压缩方式，对该待迁移数据进行压缩，得到压缩数据，将该压缩数据写入第一内存页，以实现将待迁移数据迁移到第一内存页。如果该待迁移数据不与各个缓存压缩模式匹配，MMU 将该待迁移数据写入到第一内存页，以实现将待迁移数据迁移到第一内存页。

如表 3 所示，该待迁移数据可能是第一数据，也可能是第五数据与部分第一数据的组合（称为组合数据），也可能是第五数据，如果第一数据为缓存压缩算法中某一个缓存压缩模式中的数据，在待迁移数据是第一数据或者组合数据的情况下，相对于随机数据，该待迁移数据更有可能匹配到该缓存压缩模式，从而能够提高该待迁移数据被压缩的概率，进而能够提高缓存向内存淘汰数据时缓存数据的压缩率。以第一数据为十六进制的数值 0，待迁移数据为 8 个数值 0，缓存压缩模式为 FPC 中的零运行模式为例，相对于随机数据，待迁移数据是一定能与零运行模式匹配。如果第一数据是缓存压缩算法定义的多个缓存压缩模式的多个数据序列中的数据，比如多个数据序列中出现频率最高的 S 个数据或者多个数据序列中共有的数据，在待迁移数据是第一数据或者组合数据的情况下，相对于随机数据，该待迁移数据更有可能匹配到这多个缓存压缩模式中的缓存压缩模式，从而能够进一步增加待迁移数据匹配到缓存压缩模式的概率，进一步提高待迁移数据被压缩的概率，能够进一步提高缓存向内存淘汰数据时缓存数据的压缩率。以第一数据为十六进制的数值 0，FPC 中的零运行模式、4 位符号扩展模式、半字符符号扩展模式以及用 0 半字填充的半字模式等多个数据缓存模式对应的数据序列中均出现了数值 0，如果待迁移数据为 8 个数值 0，待迁移数据可以匹配到零运行模式，如果待迁移数据为是多个数值 0 与其他数值的组合，则待迁移数据可能匹配 4 位符号扩展模式、半字符符号扩展模式或用 0 半字填充的半字模式。

本申请实施例提供的方法，通过在为引发缺页中断的虚拟地址分配完内存页后，预先在分配的内存页中填充第一数据，之后，再对填充第一数据的内存页进行访问，后续如果将内存页中的数据迁移到缓存，内存页中填充的第一数据也会迁移到缓存，如果在缓存中该内存页的数据淘汰到内存，淘汰的缓存数据也可能包括第一数据，由于第一数据为缓存压缩模式对应数据序列中的数据，当包括第一数据的缓存数据淘汰回内存时，能够增加缓存数据匹配缓存压缩模式的概率，进而能够增加缓存数据的压缩率，从而能够减少从缓存向内存传输缓存数据的开销。

图 2 所示的实施例，是以操作系统为第一虚拟页分配完第一内存页后，内存填充装置在第一内存页填充第一数据为例进行说明的，在另一种可能的实现方式中，如图 4 所示，在向第一内存页填充第一数据之前，内存填充装置还会先判断访问第一虚拟页的应用程序是否开启内存预填充功能，在应用程序没有开启内存预填充功能单元的情况下，内存填充装置不执行在第一内存页填充第一数据的步骤，向 MMU 返回填充失败响应，以指示没有在第一内存页填充第一数据。如图 4 中的步骤 A8，在应用程序开启内存预填充功能，内存填充装置在分配该虚拟地址的第一内存页填充第一数据，向 MMU 返回填充完成响应。这样，MMU 在向内存填充装置发送完分配通告消息后，接收到内存填充装置的填充失败响应或填充完成响应，然后，向操作系统返回分配完成响应，以便操作系统向处理器核心发送缺页中断处理完成响应，使得处理器核心再次执行步骤 A1。在另一些实施例中，在没有向第一内存页填充第一数据的情况下，内存填充装置也可以不向 MMU 返回填充失败响应，在向第一内存页填充第一数据的情况下，也可以不向 MMU 返回填充完成响应，针对这种情况，MMU 在向内存填充装置发送完分配通告消息后，经过一定时长后，再向操作系统返回中断处理完成响应，以为内存填充装置预留填充第一数据的时间。

针对图 4 所述的实现方式，在应用程序开启内存预填充功能的情况下，内存填充装置才会为应用程序所分配到的内存页预填充第一数据，无需为电子设备中运行的各个应用程序分配的内存页都预填充第一数据，减少了内存填充装置的工作量。

接下来,为应用程序开启内存预填充功能的方式以判断应用程序是否开启内存预填充功能的过程进行如下介绍。

在一种可能的方式中,内存填充装置通过下述开启方式1或者开启方式2,为应用程序开启内存预填充功能。

开启方式1、在应用程序的类型为目标类型的情况下,内存填充装置为该应用程序开启内存预填充功能。

其中,目标类型的应用程序适用缓存压缩模式,比如编译器、用于模拟离散事件的应用程序、用于处理大数据的应用程序等。非目标类型的应用程序不适用缓存压缩模式,比如用于压缩数据的压缩类应用程序、用于科学计算的应用程序等。

在应用程序运行前或者应用程序运行过程中,内存填充装置从操作系统中应用程序的配置信息,查询应用程序的程序类型,在应用程序的程序类型为目标类型的情况下,则内存填充装置为该应用程序开启内存预填充功能,在应用程序的程序类型不为目标类型的情况下,内存填充装置不为该应用程序开启内存预填充功能。其开启的方式比如在该配置信息中添加预设功能开启标识,预设功能开启标识用于指示应用程序开启了内存预填充功能。

在另一种可能的实现方式中,内存填充装置不负责为应用程序开启内存预填充功能,操作系统按照类似方式,为目标类型的应用程序开启内存预填充功能,或者,内存填充装置和操作系统都不主动为应用程序开启内存预填充功能,只有在用户下达了为该应用程序开启内存预填充功能的指令的情况下,操作系统才会根据用户的指令,在应用程序的该配置信息中添加预设功能开启标识,以为应用程序开启内存预填充功能。

针对上述开启方式1,在向任一内存页填充第一数据之前,内存填充装置通过查询访问该内存页的应用程序的配置信息是否包括预设功能开启标识,来判断该应用程序是否开启了内存预填充功能。以图5所示的应用程序访问内存的示意图为例,假设处理核心运行进程1和进程2,进程1和进程2的进程标识(PID)分别为11111和22222,进程1和进程2分别对应应用程序1和应用程序2,应用程序1是大数据应用,适用于缓存压缩模式,开启了内存预填充功能,应用程序2是压缩应用,不适用缓存压缩模式,没有开启内存预填充功能。处理器核心通过运行进程1和进程2,以实现运行应用程序1和应用程序2,假设进程1所访问的虚拟页1和进程2所访问的虚拟页2都触发了缺页中断,MMU为1虚拟页和2虚拟页创建完有效PTE后,通知内存填充装置中的填充单元已为虚拟页1和2分配内存页,内存填充装置基于MMU的通知,向判断单元发送判断请求,以请求判断进程1和进程2对应的应用程序是否开启内存预填充功能。判断单元接收到该判断请求后,查询应用程序1的配置信息和应用程序1的配置信息,从应用程序2的配置信息中查询到预设功能开启标识,从应用程序1的配置信息中查询到预设功能开启标识与进程1的PID1对应,则向填充单元返回判断响应,以指示进程2对应的应用程序2没有开启内存预填充功能,进程1对应的应用程序1开启了内存预填充功能。内存预设模块接收到判断响应后,根据判断响应的指示,在为虚拟页1分配的内存页中填充第一数据,不在为虚拟页2分配的内存页填充第一数据。

鉴于即使不适用该缓存压缩模式的应用程序分配的内存页预填充第一数据,该应用程序的数据从缓存迁移至内存时也不能很好的与缓存压缩模式匹配,通过上述开启方式1,在应用程序适用该缓存压缩模式的情况下,内存填充装置才会为该应用程序开启内存预填充功能,以便在该应用程序运行过程中通过为该应用程序的内存页预填充第一数据,来提高该应用程序的数据从缓存迁移至内存时的压缩效率,且还能避免对不适用该缓存压缩模式的应用程序的内存页填充第一数据,减少了内存填充装置的工作量。

开启方式2、在应用程序运行过程中,内存填充装置基于应用程序的运行情况,为应用程序周期性开启内存预填充功能。

其中,应用程序的运行情况由应用程序运行过程中触发的第一PMU事件和/或第二PMU事件来反映,第一PMU事件为在运行应用程序时运行了浮点运算指令,浮点运算指令是指用于浮点运算的指令。

第二PMU事件是指与缓存和内存之间的数据交换有关的PMU事件,第二PMU事件能够反映应用程序运行过程中缓存和内存之间的数据交换情况,第二PMU事件有至少一个,比如缓存未命中事件、TLB未命中事件、一级/二级缓存未命中次数所导致的流水线暂停(core_pipeline_stall_due_to_cache_miss)事件。

在一种可能的实现方式中,在应用程序运行过程中,每经过一个时间段,内存填充装置基于该应用程序在该时间段的运行情况,预测应用程序在下一时间段运行时是否适用该缓存压缩模式,如果适用,为应

用程序开启下一时间段的内存预填充功能，否则，不执行为应用程序开启下一时间段的内存预填充功能的步骤。其中，该时间段的时长和下一时间段的时长相同，该时间段的时长即为开启内存预填充功能的单位时长。

以图 5 为例，处理器核心在应用程序运行过程中，PMU 记录并统计应用程序触发的第一 PMU 事件和/或第二 PMU 事件，内存填充装置中的判断单元，每经过一个时间段，从 PMU 获取该时间段中第一 PMU 事件和/或第二 PMU 事件的统计信息，基于该统计信息，预测应用程序下一时间段运行时是否适用该缓存压缩模式。

示例性地，以获取该时间段中第一 PMU 事件的统计信息为例，该时间段中第一 PMU 事件的统计信息即该时间段中运行的浮点运算指令个数，判断单元还从 PMU 获取该时间段中运行的应用程序的总指令数，基于该浮点运算指令个数和总指令数，获取该时间段中运行的应用程序指令中浮点运算指令的占比，该占比为该浮点运算指令个数与该总指令数之间的比值，缓存压缩模式适用于非浮点型数据（比如整型数据），不适用浮点型数据，如果该占比小于或等于比例阈值，说明该应用程序适用该缓存压缩模式，则判断单元为应用程序开启下一时间段的内存预填充功能。如果该占比大于比例阈值，说明该应用程序不适用该缓存压缩模式，则判断单元暂不为应用程序开启内存预填充功能。

以获取第二 PMU 事件的统计信息为例，第二 PMU 事件的统计信息为例，假设缓存未命中事件、TLB 未命中事件和一级/二级缓存未命中次数所导致的流水线暂停事件中的至少一个事件均为第二 PMU 事件，该时间段中缓存未命中事件的统计信息为该时间段中缓存未命中事件的发生次数（称为第一发生次数），如果第一发生次数大于或者等于第一次数阈值，说明在该时间段中缓存和内存之间频繁交换数据，如果第一发生次数小于第一次数阈值，说明在该时间段中缓存和内存之间没有频繁交换数据。该时间段中 TLB 未命中事件的统计信息为该时间段中 TLB 未命中事件的发生次数（称为第二发生次数），如果第二发生次数大于或者等于第二次数阈值，说明在该时间段中缓存和内存之间频繁交换数据，如果第二发生次数小于第一次数阈值，说明在该时间段中缓存和内存之间没有频繁交换数据。该时间段中该流水线暂停事件的统计信息为：该时间段中流水线暂停事件所导致的流水线总暂停时长（pipeline stall），如果 pipeline stall 大于或等于时长阈值，说明在该时间段中缓存和内存之间频繁交换数据，如果 pipeline stall 小于时长阈值，说明在该时间段中缓存和内存之间没有频繁交换数据。如果在该时间段中缓存和内存之间频繁交换数据，说明应用程序在该时间段运行时适用该缓存压缩模式，应用程序在下一时间段运行时也可能适用该缓存压缩模式，如果在该时间段中缓存和内存之间没有频繁交换数据，说明应用程序在该时间段运行时不适用该缓存压缩模式，应用程序在下一时间段运行时也可能不适用该缓存压缩模式。基于此，如果第一发生次数小于第一次数阈值、第二发生次数小于第二次数阈值且 pipeline stall 小于时长阈值，应用程序在下一时间段运行时不适用该缓存压缩模式，判断单元暂不为应用程序开启内存预填充功能，如果第一发生次数、第二发生次数以及 pipeline stall 满足下述至少一项：第一发生次数大于或等于第一次数阈值、第二发生次数大于后等于第二次数阈值、pipeline stall 大于或等于时长阈值，应用程序在下一时间段运行时适用该缓存压缩模式，则判断单元为应用程序开启下一时间段的内存预填充功能。

在另一些实施例中，预设子模块不获取该时间段的第一 PMU 事件的统计信息，获取第二 PMU 事件的统计信息，仅根据第二 PMU 事件的统计信息，来确定是否需要为应用程序开启下一时间段的内存预填充功能。

在另一些实施例中，判断单元也可以将该时间段的第一 PMU 事件的统计信息和/或第一 PMU 事件的统计信息，输入到人工智能（artificial intelligence，AI）模型，由 AI 模型基于该时间段的第一 PMU 事件的统计信息和/或第一 PMU 事件的统计信息，输出判断结果，该判断结果指示是否需要为该应用程序开启下一时间段的内存预填充功能，该阈值判断子模块基于该判断结果的指示，为该应用程序开启下一时间段的内存预填充功能，或者，暂不为该应用程序开启下一时间段的内存预填充功能。其中，该 AI 模型基于多个第一训练数据以及多个第二训练数据统计信息训练得到，第一训练数据为目标类型的应用程序运行过程中触发的第一 PMU 事件和/或第二 PMU 事件的统计信息，第二训练数据是非目标类型的应用程序运行过程中触发的第一 PMU 事件和/或第二 PMU 事件的统计信息，使得 AI 模型在训练过程中能够学习到第一 PMU 事件和/或第二 PMU 事件的统计信息，与是否需要为该应用程序开启下一时间段的内存预填充功能之间的映射关系。该 AI 模型可能是决策树模型或者其他类型的 AI 模型。

以获取该时间段中第一 PMU 事件和第二 PMU 事件的统计信息为例，如果根据第一 PMU 事件的统计

信息,确定出应用程序在下一时间段运行时不适用该缓存压缩模式,判断单元再基于第二 PMU 事件的统计信息,确定是否为应用程序开启下一时间段的内存预填充功能,如果根据第一 PMU 事件的统计信息,确定应用程序在下一时间段运行时适用该缓存压缩模式,则判断单元无需第二 PMU 事件的统计信息做进一步判断。在另一些实施例中,将该判断逻辑用决策树来实现,在使用时,判断单元可以将该时间段中第一 PMU 事件和第二 PMU 事件的统计信息输入到决策树,由决策数根据上述的判断逻辑输出判断结果,该判断结果指示是否为应用程序开启下一时间段的内存预填充功能。

在另一些实施例中,判断单元也可以将该时间段的第一 PMU 事件的统计信息和/或第一 PMU 事件的统计信息,输入到人工智能(artificial intelligence, AI)模型,由 AI 模型基于该时间段的第一 PMU 事件的统计信息和/或第一 PMU 事件的统计信息,输出该判断结果。其中,该 AI 模型基于多个第一训练数据以及多个第二训练数据统计信息训练得到,第一训练数据为目标类型的应用程序运行过程中触发的第一 PMU 事件和/或第二 PMU 事件的统计信息,第二训练数据是非目标类型的应用程序运行过程中触发的第一 PMU 事件和/或第二 PMU 事件的统计信息,通过训练使得 AI 模型能够学习到第一 PMU 事件和/或第二 PMU 事件的统计信息与判断结果之间的映射关系。该 AI 模式例如神经网络模型。

如果判断单元确定出为应用程序开启下一时间段的内存预填充功能,则查询操作系统中该应用程序的配置信息是否包括预设功能开启标识,如果不包括预设功能开启标识,则在该配置信息中添加预设功能开启标识,如果包括预设功能开启标识,则无需添加预设功能开启标识。这样,在下一时间段,判断单元可以通过配置信息中的预设功能开启标识,确定该应用程序下一时间段开启了内存预填充功能。

如果判断单元确定出不为应用程序开启下一时间段的内存预填充功能,则查询操作系统中该应用程序的配置信息中是否包括预设功能开启标识,如果包括预设功能开启标识,则删除该配置信息中的预设功能开启标识。这样在下一时间段,判断单元在该配置信息中查询不到预设功能开启标识,从而能够确定该应用程序下一时间段没有开启内存预填充功能。

图 5 所示的实施例,是以内存填充装置包括判断单元以及填充单元为例进行说明的,在另一些实施例中,内存填充装置不划分子单元,此时,判断单元以及填充单元所执行的步骤都由内存填充装置来实现。

基于上述开启方式 2,在应用程序运行的过程中,可以周期性地为应用程序开启内存预填充功能,从而能够对应应用程序是否开启内存预填充功能进行更细化的控制,在没有开启内存预填充功能的时间段,无需为调入虚拟页的空内存页填充第一数据,进一步减少了内存填充装置的工作量。

在另一些实施例中,内存预填充单元不具有为应用程序开启内存预填充功能的能力,此时,可以由操作系统或者其他单元按照上述开启方式 1 或开启方式 2,为应用程序开启内存预填充功能。

图 2-图 5 所述的实施例都是以应用程序访问一个虚拟页为例进行说明的,在另一些实施例中,应用程序的某个计算任务可能对应多个虚拟页,且该多个虚拟页连续,该多个虚拟页对应的多个内存页用于存储该计算任务的任务数据,该任务数据是该应用程序用在执行该计算任务的过程中会产生数据。

在应用程序写入该任务数据时,可能向该多个虚拟页中的所有虚拟页写入任务数据,也可能向该多个虚拟页的部分虚拟页写入任务数据,在向部分虚拟页写入数据的时,写入任务数据的虚拟页可能相邻,也可能不相邻。示例性地,该计算任务对应虚拟页 1-3,虚拟页 1-3 连续,假设设定当该计算任务满足条件 1 时,应用程序将该计算任务的任务数据写入虚拟页 1,当该计算任务满足条件 2 时,将该计算任务的任务数据写入虚拟页 2,当该计算任务满足条件 3 时,将计算任务的任务数据写入虚拟页 3。应用程序在执行该计算任务时,计算任务可能不满足条件 1-3 中的某个条件时,导致应用程序不会向虚拟页 1-3 中的部分虚拟页填充任务数据,如计算任务执行过程中仅满足条件 1 和 3,不满足条件 2,则应用程序会向虚拟页 1 和 3 写入任务数据,不会向虚拟页 2 写入任务数据,在相关技术中,MMU 在虚拟页 1 和虚拟页 3 对应的内存页 1 和 3 写入任务数据,以实现向虚拟页 1 和 3 写入任务数据,由于应用程序没有向虚拟页 2 写入任务数据,MMU 会在虚拟页 2 对应的内页 2 填充随机数值,一旦内存页 1-3 中的数据被迁移至缓存,内存页 2 中填充的随机数值也会被迁移至缓存,当缓存中内存页 1-3 中的缓存数据被淘汰回内存页时,内存页 2 的缓存数据中的随机数值导致该缓存数据不容易匹配到缓存压缩模式,导致缓存压缩率低。

本申请通过内存填充单元在应用程序的各个空内存页填充第一数据,如果应用程序所执行的计算任务的任务数据没有填充到对应的内存页,由于内存页中已经填充第一数据,MMU 则不会在内存页中继续填充随机数值,从而可以避免因填充随机数值导致的缓存压缩率低的问题。

示例性地,以应用程序的某个计算任务对应多个虚拟页(称为第二虚拟页)为例,将第二虚拟页对应

的内存页称为第二内存页，多个第二内存页均用于存储应用程序的该计算任务的任务数据。按照在第一内存页填充第一数据的处理流程（图2或者图4），在各个第二内存页调入对应第二虚拟页时，内存填充单元都会在该第二内存页填充第一数据，从而实现应用程序在向各个第二内存页写入任务数据之前，预先为各个第二内存页填充第一数据。

在处理器核心执行该应用程序的该计算任务的过程中，在应用程序向多个第二虚拟页中的任一虚拟地址（称为第二虚拟地址）第二虚拟地址写入任务数据时，触发处理器核心向内存管理装置发送应用程序的第二写请求，该第二写请求指示向第二虚拟地址写入任务数据。内存管理装置接收到该第二写请求后，将第二写请求中的第二虚拟地址转换成物理地址（称为第二物理地址），其中，转换地址的过程可参考将第一虚拟地址转换成第一物理地址的过程。内存管理装置将该多个第二内存页中该第二物理地址处的第一数据更新待写入任务数据，以实现将多个第二内存页中第二虚拟地址对应位置处的第一数据更新为任务数据。其中，将第二物理地址所在的第二内存页称为第三内存页，在向第三内存页中第二物理地址处写入任务数据时，如果第三内存页中的数据还未迁移到缓存中的缓存行（称为第二缓存行），则该内存管理装置将该任务数据写入到第三内存页中的第二物理地址处，如果第三内存页中的数据已经迁移到第二缓存行，则该内存管理装置将该任务数据写入到第二缓存行中第二物理地址对应的位置，以将该位置处的第一数据覆盖掉。后续如果将第二缓存行中第三内存页的缓存数据淘汰回第三内存页，由于缓存数据包括第一数据，使得该缓存数据更容易匹配缓存压缩模式，从而能够提高缓存压缩率。

在将多个第二内存页中的数据迁移至缓存后，在多个第二内存页填充的第一数据和/或应用程序写入的任务数据都会一起迁移到缓存。对于该多个第二内存页中除第三内存页以外的任一内存页（称为第四内存页），在向第四内存页写入任务数据之前，第四内存页中的数据均是第一数据，缓存中第四内存页的缓存数据均是第二数据，如果在计算任务执行过程中应用程序一直没有向第四内存页写入任务数据，缓存中第四内存页的缓存数据一直都是第一数据，而不会是随机数值，后续将第四内存页的缓存数据淘汰回第四内存页时，缓存数据更容易匹配缓存压缩模式，从而能够提高缓存压缩率。如果后续应用程序向第四内存页写入了任务数据，该任务数据会被写入到缓存中第四内存页的缓存数据，以覆盖掉该缓存数据中的部分或全部第一数据，后续在将第四内存页的缓存数据淘汰回第四内存页时，如果第四内存页的缓存数据仍有第一数据，使得该缓存数据更容易匹配缓存压缩模式，从而能够提高缓存压缩率。

以上介绍了本申请实施例的方法，以下介绍本申请实施例的装置和设备。应理解，以下介绍的装置和设备具有和上述方法实施例的同一发明构思，能够实现上述方法实施例所能达到的有益效果，下面将结合图6至图8描述本申请实施例的装置和设备。应理解，方法实施例所描述的技术特征同样适用于以下装置实施例和设备实施例。

图6是本申请实施例提供的一种内存页的处理设备的结构示意图，图6所示的设备600可以为电子设备，如终端、服务器或者网络设备等，在此对设备600的设备类型不做限定。如图6所示，设备600包括：

运行装置601，用于在应用程序所访问的第一虚拟地址引发缺页中断的情况下，通过运行操作系统，为所述第一虚拟地址分配第一内存页，其中，运行装置601为设备600中运行操作系统的模块。

内存填充装置602，用于在所述第一内存页填充第一数据，所述第一数据是缓存压缩模式对应数据序列中的数据，所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式；

内存管理装置603，用于基于所述第一虚拟地址，对填充后的所述第一内存页进行访问。

在一种可能的实现方式中，所述内存填充装置602还用于：

在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据，所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。

在一种可能的实现方式中，所述内存填充装置602还用于：

所述内存填充装置基于所述应用程序的运行情况，为所述应用程序周期性开启所述内存预填充功能，否则，不执行所述为所述应用程序开启所述内存预填充功能的步骤。

在一种可能的实现方式中，所述内存填充装置602还用于：

在所述应用程序的类型是目标类型情况下，为所述应用程序开启所述内存预填充功能，所述目标类型的应用程序适用所述缓存压缩模式。

在一种可能的实现方式中，所述内存管理装置603用于：

接收所述应用程序的第一写请求,所述第一写请求指示向所述第一虚拟地址写入第二数据;

将所述第一内存页中所述第一虚拟地址对应位置处的所述第一数据更新为所述第二数据。

在一种可能的实现方式中,多个第二内存页均用于存储所述应用程序的同一计算任务的任务数据,所述多个第二内存页均填充所述第一数据,所述内存管理装置 603 还用于:

接收所述应用程序的第二写请求,所述第二写请求指示向第二虚拟地址写入所述任务数据;

将所述多个第二内存页中所述第二虚拟地址对应位置处的所述第一数据更新为所述任务数据。

在一种可能的实现方式中,所述第一数据是所述数据序列中的各个数据;或者,所述第一数据是所述数据序列中出现频率最高的 S 个数据, S 大于 0;或者,所述第一数据是所述数据序列中连续的 R 个数据, R 大于 0。

在一种可能的实现方式中,所述第一数据是多个数据序列中出现频率最高的 S 个数据,所述多个数据序列对应不同的所述缓存压缩模式, S 大于 0;或者,所述第一数据是所述多个数据序列共有的数据。

设备 600 在处理内存页时,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将设备 600 的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的设备 600 与上述方法实施例属于同一构思,其具体实现过程详见上述方法实施例,这里不再赘述。

图 7 是本申请实施例提供的一种内存填充装置的结构示意图,如图 7 所示,装置 700 包括:

获取单元 701,用于获取分配通告消息,所述分配通告消息用于通告针对应用程序访问第一虚拟地址所引发的缺页中断,已为所述第一虚拟地址分配第一内存页;

填充单元 702,用于在所述第一内存页填充第一数据,所述第一数据是缓存压缩模式对应数据序列中的数据,所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式。

在一种可能的实现方式中,所述填充单元 702 还用于:

在所述应用程序开启内存预填充功能的情况下,在所述第一内存页填充所述第一数据,所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。

在一种可能的实现方式中,所述装置 700 还包括:

开启单元,用于基于所述应用程序的运行情况,为所述应用程序周期性开启所述内存预填充功能。

在一种可能的实现方式中,所述装置 700 还包括:

开启单元,用于在所述应用程序的类型是目标类型情况下,为所述应用程序开启所述内存预填充功能,所述目标类型的应用程序适用所述缓存压缩模式。

上述两种实现方式涉及的开启单元可以为上述实施例介绍的内存填充装置中的判断单元。

在一种可能的实现方式中,所述第一数据是所述数据序列中的各个数据;或者,所述第一数据是所述数据序列中出现频率最高的 S 个数据, S 大于 0;或者,所述第一数据是所述数据序列中连续的 R 个数据, R 大于 0。

在一种可能的实现方式中,所述第一数据是多个数据序列中出现频率最高的 S 个数据,所述多个数据序列对应不同的所述缓存压缩模式, S 大于 0;或者,所述第一数据是所述多个数据序列共有的数据。

应理解,装置 700 在处理内存页时,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将装置 700 的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的装置 700 与上述方法实施例属于同一构思,其具体实现过程详见上述方法实施例,这里不再赘述。

图 8 是本申请实施例提供的另一种内存填充装置的结构示意图,如图 8 所示,内存填充装置 800 包括:总线 802、处理器 804、存储器 806 和通信接口 808。处理器 804、存储器 806 和通信接口 808 之间通过总线 802 通信。内存填充装置 800 可以是服务端或客户端。应理解,本申请不限定内存填充装置 800 中的处理器、存储器的个数。总线 802 可以是外设部件互连标准(peripheral component interconnect, PCI)总线或扩展工业标准结构(extended industry standard architecture, EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示,图 8 中仅用一条线表示,但并不表示仅有一根总线或一种类型的总线。总线 802 可包括在内存填充装置 800 各个部件(例如,存储器 806、处理器 804、通信接口 808)之间传递信息的通路。处理器 804 可以包括中央处理器(central processing unit, CPU)、图形处理器(graphics processing unit, GPU)、微处理器(micro processor, MP)或者数字信号处理器(digital signal processor, DSP)等处

理器中的任意一种或多种。存储器 806 可以包括易失性存储器 (volatile memory)，例如随机存取存储器 (random access memory, RAM)。存储器 806 还可以包括非易失性存储器 (non-volatile memory)，例如只读存储器 (read-only memory, ROM)、快闪存储器、机械硬盘 (hard disk drive, HDD) 或固态硬盘 (solid state drive, SSD)。存储器 806 中存储有可执行的程序代码，处理器 804 执行该可执行的程序代码，使得内存填充装置 800 实现内存页的处理方法。在一种可能的实现方式中，内存填充装置 800 是芯片。

在示例性实施例中，还提供了一种电子设备，该电子设备包括处理器，处理器用于执行程序代码，使得电子设备执行以实现应用于电子设备的内存页的处理方法所执行的操作，示例性地，该电子设备的硬件结构可参考内存填充装置 800，与内存填充装置 800 的区别在于，电子设备中的存储器所存储的程序代码用于实现应用于电子设备的内存页的处理方法。

在示例性实施例中，还提供了一种计算机可读存储介质，例如包括程序代码的存储器，上述程序代码可由内存填充装置 800 的处理器执行以完成上述实施例中的内存页的处理方法。或者，上述程序代码由电子设备的处理器执行，以完成上述实施例中应用于电子设备的内存页的处理方法。例如，该计算机可读存储介质是非临时计算机可读存储介质，如只读存储器 (read-only memory, ROM)、随机存取存储器 (random access memory, RAM)、只读光盘 (compact disc read-only memory, CD-ROM)、磁带、软盘和光数据存储设备等。

本申请实施例还提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括程序代码，该计算机指令存储在计算机可读存储介质中，内存填充装置的处理器从计算机可读存储介质读取该程序代码，处理器执行该程序代码，使得内存填充装置执行上述实施例中的内存页的处理方法。或者，电子设备的处理器从计算机可读存储介质读取该程序代码，处理器执行该程序代码，使得电子设备执行上述实施例中的内存页的处理方法。

其中，本实施例提供的装置、设备、计算机可读存储介质、计算机程序产品或芯片均用于执行上文所提供的对应的方法，因此，其所能达到的有益效果可参考上文所提供的对应的方法中的有益效果，此处不再赘述。

在本申请所提供的几个实施例中，应该理解到，所揭露的装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述模块或单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个装置，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或单元的间接耦合或通信连接，可以是电性，机械或其它的形式。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是一个物理单元或多个物理单元，即可以位于一个地方，或者也可以分布到多个不同地方。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个可读取存储介质中。基于这样的理解，本申请实施例的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来，该软件产品存储在一个存储介质中，包括若干指令用以使得一个设备（可以是单片机，芯片等）或处理器 (processor) 执行本申请各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：U 盘、移动硬盘、ROM、RAM、磁碟或者光盘等各种可以存储程序代码的介质。

在本申请的描述中，除非另有说明，“/”表示“或”的意思，例如，A/B 可以表示 A 或 B。本文中的“和/或”仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A 和/或 B，可以表示：单独存在 A，同时存在 A 和 B，单独存在 B 这三种情况。此外，“至少一个”是指一个或多个，“多个”是指两个或两个以上。“第一”、“第二”等字样并不对数量和执行次序进行限定，并且“第一”、“第二”等字样也并不限定一定不同。

本申请中，“示例性的”或者“例如”等词用于表示作例子、例证或说明。本申请中被描述为“示例性的”或者“例如”的任何实施例或设计方案不应被解释为比其他实施例或设计方案更优选或更具优势。确切而言，

使用“示例性的”或者“例如”等词旨在以具体方式呈现相关概念。

需要说明的是，本申请所涉及的信息（包括但不限于用户设备信息、用户个人信息等）、数据（包括但不限于用于分析的数据、存储的数据、展示的数据等）以及信号，均为经用户授权或者经过各方充分授权的，且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。例如，本申请中涉及到的向内存页写入的数据都是在充分授权的情况下获取的。

上述所有可选技术方案，可以采用任意结合形成本公开的可选实施例，在此不再一一赘述。

以上所述仅为本申请的可选实施例，并不用以限制本申请，凡在本申请的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。

权利要求书

- 1.一种内存页的处理方法，其特征在于，所述方法包括：
在应用程序所访问的第一虚拟地址引发缺页中断的情况下，为所述第一虚拟地址分配第一内存页；
在所述第一内存页填充第一数据，所述第一数据是缓存压缩模式对应数据序列中的数据，所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式；
基于所述第一虚拟地址，对填充后的所述第一内存页进行访问。
- 2.根据权利要求1所述的方法，其特征在于，所述在所述第一内存页填充第一数据包括：
在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据，所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。
- 3.根据权利要求2所述的方法，其特征在于，所述在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据之前，所述方法还包括：
基于所述应用程序的运行情况，为所述应用程序周期性开启所述内存预填充功能。
- 4.根据权利要求2所述的方法，其特征在于，所述在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据之前，所述方法还包括：
在所述应用程序的类型是目标类型情况下，为所述应用程序开启所述内存预填充功能，所述目标类型的应用程序适用所述缓存压缩模式。
- 5.根据权利要求1-4中任一项所述的方法，其特征在于，所述基于所述第一虚拟地址，对填充后的所述第一内存页进行访问包括：
接收所述应用程序的第一写请求，所述第一写请求指示向所述第一虚拟地址写入第二数据；
将所述第一内存页中所述第一虚拟地址对应位置处的所述第一数据更新为所述第二数据。
- 6.根据权利要求1-5所述的方法，其特征在于，多个第二内存页均用于存储所述应用程序的同一计算任务的任务数据，所述多个第二内存页均填充所述第一数据，所述方法还包括：
接收所述应用程序的第二写请求，所述第二写请求指示向第二虚拟地址写入所述任务数据；
将所述多个第二内存页中所述第二虚拟地址对应位置处的所述第一数据更新为所述任务数据。
- 7.根据权利要求1-6中任一项所述的方法，其特征在于，
所述第一数据是所述数据序列中的各个数据；
或者，所述第一数据是所述数据序列中出现频率最高的S个数据，S大于0；
或者，所述第一数据是所述数据序列中连续的R个数据，R大于0。
- 8.根据权利要求1-6中任一项所述的方法，其特征在于，
所述第一数据是多个数据序列中出现频率最高的S个数据，所述多个数据序列对应不同的所述缓存压缩模式，S大于0；
或者，所述第一数据是所述多个数据序列共有的数据。
- 9.一种内存页的处理方法，其特征在于，所述方法包括：
获取分配通告消息，所述分配通告消息用于通告针对应用程序访问第一虚拟地址所引发的缺页中断，已为所述第一虚拟地址分配第一内存页；
在所述第一内存页填充第一数据，所述第一数据是缓存压缩模式对应数据序列中的数据，所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式。

10.根据权利要求 9 所述的方法，其特征在于，所述在所述第一内存页填充第一数据包括：

在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据，所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。

11.根据权利要求 10 所述的方法，其特征在于，所述在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据之前，所述方法还包括：

基于所述应用程序的运行情况，为所述应用程序周期性开启所述内存预填充功能。

12.根据权利要求 10 所述的方法，其特征在于，所述在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据之前，所述方法还包括：

在所述应用程序的类型是目标类型情况下，为所述应用程序开启所述内存预填充功能，所述目标类型的应用程序适用所述缓存压缩模式。

13.一种内存页的处理设备，其特征在于，所述设备包括：

运行装置，用于在应用程序所访问的第一虚拟地址引发缺页中断的情况下，通过运行操作系统，为所述第一虚拟地址分配第一内存页；

内存填充装置，用于在所述第一内存页填充第一数据，所述第一数据是缓存压缩模式对应数据序列中的数据，所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式；

内存管理装置，用于基于所述第一虚拟地址，对填充后的所述第一内存页进行访问。

14.根据权利要求 13 所述的设备，其特征在于，所述内存填充装置还用于：

在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据，所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。

15.根据权利要求 14 所述的设备，其特征在于，所述内存填充装置还用于：

基于所述应用程序的运行情况，为所述应用程序周期性开启所述内存预填充功能。

16.根据权利要求 14 所述的设备，其特征在于，所述内存填充装置还用于：

在所述应用程序的类型是目标类型情况下，为所述应用程序开启所述内存预填充功能，所述目标类型的应用程序适用所述缓存压缩模式。

17.根据权利要求 13-16 中任一项所述的设备，其特征在于，所述内存管理装置用于：

接收所述应用程序的第一写请求，所述第一写请求指示向所述第一虚拟地址写入第二数据；

将所述第一内存页中所述第一虚拟地址对应位置处的所述第一数据更新为所述第二数据。

18.根据权利要求 13-17 所述的设备，其特征在于，多个第二内存页均用于存储所述应用程序的同一计算任务的任务数据，所述多个第二内存页均填充所述第一数据，所述内存管理装置还用于：

接收所述应用程序的第二写请求，所述第二写请求指示向第二虚拟地址写入所述任务数据；

将所述多个第二内存页中所述第二虚拟地址对应位置处的所述第一数据更新为所述任务数据。

19.根据权利要求 13-18 中任一项所述的设备，其特征在于，

所述第一数据是所述数据序列中的各个数据；

或者，所述第一数据是所述数据序列中出现频率最高的 S 个数据，S 大于 0；

或者，所述第一数据是所述数据序列中连续的 R 个数据，R 大于 0 所述第一数据为十六进制的数值 0 或者数值 F。

20.根据权利要求 13-18 中任一项所述的设备，其特征在于，
所述第一数据是多个数据序列中出现频率最高的 S 个数据，所述多个数据序列对应不同的所述缓存压缩模式，S 大于 0；

或者，所述第一数据是所述多个数据序列共有的数据。

21.一种内存填充装置，其特征在于，所述装置包括：

获取单元，用于获取分配通告消息，所述分配通告消息用于通告针对应用程序访问第一虚拟地址所引发的缺页中断，已为所述第一虚拟地址分配第一内存页；

填充单元，用于在所述第一内存页填充第一数据，所述第一数据是缓存压缩模式对应数据序列中的数据，所述缓存压缩模式是将缓存中所述应用程序写入的数据迁移至内存页时所使用的数据压缩模式。

22.根据权利要求 21 所述的装置，其特征在于，所述填充单元还用于：

在所述应用程序开启内存预填充功能的情况下，在所述第一内存页填充所述第一数据，所述内存预填充功能指示在所述应用程序访问空内存页之前在所述空内存页填充所述第一数据。

23.一种电子设备，其特征在于，所述电子设备包括处理器，所述处理器用于执行程序代码，使得所述电子设备执行如权利要求 1 至权利要求 8 中任一项所述的方法。

24.一种内存填充装置，其特征在于，所述内存填充装置包括处理器，所述处理器用于执行程序代码，使得所述内存填充装置执行如权利要求 9 至权利要求 12 中任一项所述的方法。

25.一种计算机可读存储介质，其特征在于，所述存储介质中存储有至少一条程序代码，所述至少一条程序代码由处理器读取以使执行如权利要求 1 至权利要求 12 中任一项所述的方法。