(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114860163 B
(45)授权公告日 2023.08.22

(21)申请号 202210439155.X
(22)申请日 2020.07.01
(65)同一申请的已公布的文献号
申请公布号 CN 114860163 A
(43)申请公布日 2022.08.05
(66)本国优先权数据
202010348770.0 2020.04.28 CN
(62)分案原申请数据
202010625111.7 2020.07.01
(73)专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
(72)发明人 崔文林 黄克骥 张鹏 罗四维
(74)专利代理机构 广州三环专利商标代理有限公司 44202
专利代理师 胡丽平
(51)Int.Cl.
G06F 3/06(2006.01)
(56)对比文件
CN 110333825 A,2019.10.15
CN 103440208 A,2013.12.11
CN 104303167 A,2015.01.21
CN 108804350 A,2018.11.13
US 2016/0342363 A1,2016.11.24
US 2016/0283136 A1,2016.09.29
US 2015/0378913 A1,2015.12.31
审查员 石爽

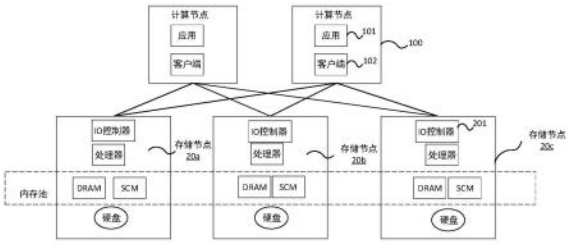
权利要求书1页 说明书17页 附图7页

(54)发明名称

一种存储系统、内存管理方法和管理节点

(57)摘要

一种存储系统，内存管理方法和管理节点。该存储系统包括管理节点、一个或多个第一存储器和一个或多个第二存储器。所述管理节点用于创建内存池以提供存储数据的服务。所述第一存储器的性能高于所述第二存储器，其中，所述一个或多个第一存储器中至少有一个第一存储器位于第一存储节点，所述一个或多个第二存储器中至少有一个第二存储器位于第二存储节点。所述管理节点还用于控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移。所述内存池是基于不同性能的存储器创建的，并且这些不同性能的存储器位于不同的存储节点上，从而实现了跨节点的，融合了不同性能的存储器的内存池。



CN 114860163 B

1.一种存储系统,其特征在于,包括多个服务器,每个服务器包括处理器、具有内存接口的第一存储器以及具有内存接口的第二存储器,所述第一存储器的性能高于所述第二存储器;每个服务器包括的所述第一存储器和所述第二存储器组成内存池;

在所述内存池中,每个服务器提供的一个或多个所述第一存储器属于第一层级,每个服务器提供的一个或多个第二存储器属于第二层级;

在每个服务器中,所述处理器用于通过所述第一存储器的内存接口访问所述第一存储器,以及通过所述第二存储器的内存接口访问所述第二存储器;

每个服务器还包括硬盘,所述硬盘的性能低于所述第一存储器和所述第二存储器;每个服务器包括的硬盘组成存储池;

所述处理器还用于根据数据的访问频率确定将所述数据存储于所述内存池中的所述第一层级或所述第二层级,或者存储于所述存储池。

2.根据权利要求1所述的存储系统,其特征在于,所述处理器还用于:提供存储空间给客户端,所述存储空间来源于所述第一层级以及第二层级,所述客户端中可运行应用程序,所述客户端不感知所述存储空间来自的服务器。

3.一种内存管理方法,其特征在于,应用于存储系统,所述存储系统包括多个服务器,所述方法包括:

为每个服务器提供具有内存接口的第一存储器以及具有内存接口的第二存储器,所述第一存储器的性能高于所述第二存储器,每个服务器包括的所述第一存储器和所述第二存储器组成内存池,在所述内存池中,每个服务器提供的一个或多个所述第一存储器属于第一层级,每个服务器提供的一个或多个第二存储器属于第二层级所述;

为每个服务器提供处理器,所述处理器通过所述第一存储器的内存接口访问所述第一存储器,以及通过所述第二存储器的内存接口访问所述第二存储器;

为每个服务器提供硬盘,所述硬盘的性能低于所述第一存储器和所述第二存储器;每个服务器包括的硬盘组成存储池;

所述处理器根据数据的访问频率确定将所述数据存储于所述内存池中的所述第一层级或所述第二层级,或者存储于所述存储池。

4.根据权利要求3所述的方法,其特征在于,所述方法还包括:

提供存储空间给客户端,所述存储空间来源于所述第一层级以及第二层级,所述客户端中可运行应用程序,所述客户端不感知所述存储空间来自的服务器。

5.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质中存储有程序指令,所述程序指令用于执行权利要求3或4所述的方法。

一种存储系统、内存管理方法和管理节点

[0001] 本申请是分案申请，原申请的申请号是202010625111.7，原申请日是2020年7月1日，原申请的全部内容通过引用结合在本申请中。

技术领域

[0002] 本申请涉及存储技术领域，特别是一种存储系统、内存管理方法和管理节点。

背景技术

[0003] 伴随着存储级存储器(Storage Class Memory，SCM)产品的成熟，内存种类变得丰富，然而现在内存的使用通常仍局限于一个存储节点内部，并没有充分利用内存的性能优势。

发明内容

[0004] 本申请第一方面提供了一种存储系统，该存储系统包括管理节点、一个或多个第一存储器和一个或多个第二存储器。所述管理节点用于创建内存池以提供存储数据的服务。所述第一存储器的性能高于所述第二存储器，其中，所述一个或多个第一存储器中至少有一个第一存储器位于第一存储节点，所述一个或多个第二存储器中至少有一个第二存储器位于第二存储节点。所述管理节点还用于控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移。

[0005] 第一方面提供的内存池至少包含了以下几种情况：1)第一存储节点包含第一存储器和第二存储器，第二存储节点也包含第一存储器和第二存储器，所有的第一存储器和所有的第二存储器都是所述内存池的一部分；2)第一存储节点仅包含第一存储器，第二存储节点仅包含第二存储器，第一存储器和第二存储器都是所述内存池的一部分；3)第一存储节点包含第一存储器和第二存储器，第二存储节点仅包含第一存储器或第二存储器中的其中一种，这些第一存储器和第二存储器均是所述内存池的一部分。除此之外，所述存储系统还可以包含其他存储节点，所述其他存储节点所包含的各种类型的存储器也可以为所述内存池提供存储空间。

[0006] 第一存储器和第二存储器因为类型不同而导致性能有所差异，总体而言，第一存储器的性能高于第二存储器的性能，这里的性能是指存储器的运算速度和/或访问时延。举例来说，第一存储器为动态随机存取存储器，第二存储器为储存级存储器。另外，第一方面的第一存储器和第二存储器是根据类型来划分的，例如，对于动态随机存取存储器来说，无论它位于第一存储节点还是第二存储节点，都被称为第一存储器。而对于储存级存储器，无论它位于第一存储节点还是第二存储节点，都被称为第二存储器。同理，所述内存池还可以包含第三存储器、第四存储器等等。

[0007] 在第一方面的存储系统中，基于多种性能的存储器创建内存池，并且这些多种性能的存储器位于不同的存储节点上，从而实现了跨节点的，融合了不同性能的存储器的内存池，使得各种类型的存储器(无论是内存还是硬盘)都能够作为存储资源为上层提供存储

服务，从而更好地发挥其性能优势。由于所述内存池中包含了不同性能的存储器，所以可以控制数据基于其访问频率在不同性能的存储器之间迁移，既能够在数据的访问频率较高时迁移至高性能的存储器以提高读数据的效率，也可以在数据的访问频率较低时迁移至低性能的存储器以节省高性能的存储器的存储空间。另外本申请中的内存池为计算节点或LUN提供存储空间，它改变内存资源以处理器为中心的架构。

[0008] 在一种实现中，所述管理节点还用于获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。内存池的容量取决于各个存储节点提供给它的存储器的容量，内存池所包含的存储器的类型取决于各个存储节点提供给它的存储器的类型。所以管理节点在创建所述内存池之前需要从各个存储节点采集存储器的状态信息。

[0009] 除此之外，所述内存池可以在可用空间不足或者在存储系统发现新的存储节点时进行扩容，扩容时管理节点也需要从新的存储节点采集存储器的状态信息。

[0010] 在一种实现中，所述内存池的存储空间包括若干个页，一个页在内存池中的位置被称为全局地址，所述全局地址映射到所述页的物理地址，所述页的物理地址用于指示为所述页分配的物理空间在存储节点内的存储器内的位置。举例来说，所述页的尺寸为4KB，或者8KB，或者16KB等等。页的尺寸可以是固定大小，也可以根据需求配置成不同大小。具有不同大小的页的内存池，在使用时会更加灵活。

[0011] 在一种实现中，所述第一存储节点和所述第二存储节点都保存有索引表，所述索引表用于记录所述页的全局地址与所述页的物理地址的映射关系。

[0012] 在一种实现中，可以预先为某些页的全局地址分配好物理地址，将全局地址和物理地址的映射关系记录在所述索引表中，那么当存储节点接收到针对这些页的写数据请求时，就可以直接根据所述索引表查询到物理地址，从而将待写入数据写入所述物理地址指示的物理空间中。采用预先分配的方式，在执行写数据请求的时候可以直接将数据写入预先分配的物理空间中，由此提高写数据的效率。

[0013] 在一种实现中，可以不预先为页分配物理地址，而是在存储节点接收到针对这些页的写数据请求时，再从存储器中分配物理空间，将待写入数据写入所述物理空间，并且将所述页的全局地址与物理地址的映射关系记录在所述索引表中。这种按需分配的方式，可以更加灵活地分配物理空间，需要多少就分多少，达到节省空间的目的。

[0014] 任意一个存储节点对索引表有更新时，都可以将更新的内容发送给其他节点以及管理节点。使得每个存储节点都拥有完整的索引表，从而避免了信息不透明的问题。因为每个存储节点都可能接收到计算节点的读数据请求，那么就可以通过所述索引表查询到待读取数据的物理地址，获得正确的数据。

[0015] 在第一方面的存储系统中，关于写数据请求的执行，本申请提供了至少三种实现。

[0016] 在一种实现中，第一存储节点包括IO控制器，所述IO控制器存储有所述索引表，所述IO控制器与计算节点通信，所述IO控制器存储有所述索引表，所述IO控制器与计算节点通信。所述IO控制器用于接收所述计算节点发送的第一数据和所述第一数据的第一逻辑地址，根据所述第一逻辑地址确定所述第一数据在所述内存池中的第一全局地址，根据所述索引表确定是否已经为所述第一全局地址分配有物理空间，当确定已经为所述第一全局地址分配有物理空间时，将所述第一数据写入所述第一物理地址指示的物理空间中。在这种

实现中，第一存储节点和计算节点之间采用LUN语义进行通信，所述第一逻辑地址是指LUN ID，LBA和长度。

[0017] 相应的，在上述实现中，如果所述IO控制器用于接收所述计算节点发送的读数据请求以读取所述第一数据，所述读数据请求携带所述第一逻辑地址，那么所述IO控制器根据所述第一逻辑地址确定所述第一数据在所述内存池中的第一全局地址，并根据所述索引表确定所述第一全局地址对应的第一物理地址，从所述第一物理地址指示的物理空间读取所述第一数据。

[0018] 在一种实现中，第一存储节点和计算节点之间采用内存语义进行通信，此时，第一逻辑地址是指虚拟空间的ID，起始地址和长度。除此之外，写数据请求的执行过程和上述实现流程一致。读数据请求的执行过程也同样如此。

[0019] 在一种实现中，存储系统中的各个存储节点将自己本地的那部分内存池映射给计算节点，由此计算节点可以“看到”所述内存池，并获得所述内存池的全局地址。那么计算节点发送给第一存储节点的写数据请求就可以直接携带全局地址，第一存储节点中的IO控制器根据所述索引表确定所述全局地址对应的物理地址，将数据写入所述物理地址指示的物理空间中。相应的，在这种实现中，如果计算节点给第一存储节点发送读数据请求，所述读数据请求也携带的是待读取数据的全局地址，IO控制器可以直接根据索引表确定全局地址对应的物理地址。按照这种实现方式，写数据和读数据的效率都可以提高，因为减少了逻辑地址转换为全局地址的步骤。

[0020] 在一种实现中，管理节点在控制数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移时具体用于，指示各个存储节点监控其本地的数据的访问频率，指示该存储节点在访问频率高于一定阈值时将该数据迁移至性能更高的存储器中；或者在访问频率低于一定阈值时，指示该存储节点将该数据迁移至性能更低的存储器中。注意，这里的迁移可以是在存储节点内迁移，也可以是跨节点的迁移。

[0021] 在上种实现中，可以以页为粒度进行访问频率监控，并且以页为粒度进行数据迁移。也可以以数据项为粒度进行访问频率监控，并且以数据项为粒度进行数据迁移。数据项是比页粒度更小的单元，以数据项为粒度进行监控可以更加准确地对热点数据(或者非热点)进行识别，并且将真正的热点数据迁移至性能更高的存储器中提高读数据的效率。

[0022] 在一种实现中，第一方面提供的内存还支持数据预取。具体的，在上述任意一种读数据请求的执行过程中，或者在读数据请求完成之后，将与待读取数据相关联的其他数据预取至较高性能的存储器中，相关联的数据是指逻辑地址与待读取数据的逻辑地址连续的数据。数据预取可以提高高性能的存储器的命中率，提高读数据的效率。

[0023] 在一种实现中，第一方面的存储系统适用于计算与存储分离的场景。换言之，计算节点独立于系统中的任意一个存储节点。计算节点和存储节点的通信是通过外部网络来实现的，这种方式的好处是扩容比较方便。例如，当计算资源不足时，可以增加计算节点的数量，存储节点的数量不变。当存储资源不足时，可以增加存储节点的数量，计算节点的数量不变。

[0024] 在一种实现中，第一方面的存储系统适用于计算与存储融合的场景。换言之，一个计算节点和一个存储节点是集成在同一个物理设备中的，我们可以把这个集成了计算和存储的物理设备叫做存储服务器，或者也称作存储节点。在这种情况下，计算节点和存储节点

的通信是通过内部网络来实现的，那么在执行读数据请求或写数据请求时，访问时延比较低。

[0025] 第二方面提供了一种内存管理方法，该方法应用于存储系统中，所述存储系统中的管理节点或者存储节点执行其中的方法步骤，以实现第一方面的功能。

[0026] 第三方面提供了一种管理节点，所述管理节点位于第一方面提供的存储系统中，所述存储系统包括一个或多个第一存储器和一个或多个第二存储器。所述管理节点包括创建模块控制模块。所述创建模块用于创建内存池以提供存储数据的服务，所述内存池包括所述一个或多个第一存储器以及所述一个或多个第二存储器，所述第一存储器的性能高于所述第二存储器，其中，所述一个或多个第一存储器中至少有一个第一存储器位于第一存储节点，所述一个或多个第二存储器中至少有一个第二存储器位于第二存储节点。所述控制模块用于控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移。

[0027] 第三方面提供的管理节点基于多种性能的存储器创建内存池，并且这些多种性能的存储器位于不同的存储节点上，从而实现了跨节点的，融合了不同性能的存储器的内存池，使得各种类型的存储器(无论是内存还是硬盘)都能够作为存储资源为上层提供存储服务，从而更好地发挥其性能优势。由于所述内存池中包含了不同性能的存储器，所以可以控制数据基于其访问频率在不同性能的存储器之间迁移，既能够在数据的访问频率较高时迁移至高性能的存储器以提高读数据的效率，也可以在数据的访问频率较低时迁移至低性能的存储器以节省高性能的存储器的存储空间。另外本申请中的内存池为计算节点或LUN提供存储空间，它改变内存资源以处理器为中心的架构。

[0028] 在一种实现中，所述创建模块还用于获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。所述创建模块具体用于根据所述状态信息创建所述内存池。

[0029] 在一种实现中，所述内存池的存储空间包括若干个页，一个页在内存池中的全局地址映射到所述页的物理地址；其中，所述页的全局地址用于指示所述页在所述内存池中的位置，所述页的物理地址用于指示为所述页分配的物理空间在存储节点内的存储器内的位置。

[0030] 在一种实现中，所述数据存储在所述第一存储器，所述控制模块具体用于指示所述第一存储节点获取所述数据的访问频率；以及指示所述第一存储节点在所述访问频率低于设定的频率阈值时将所述数据迁移至所述内存池中的第二存储器。

[0031] 第四方面提供了一种管理节点，所述管理节点位于第一方面提供的存储系统中，所述存储系统包括一个或多个第一存储器和一个或多个第二存储器。所述管理节点包括接口和处理器。所述处理器用于创建内存池以提供存储数据的服务，所述内存池包括所述一个或多个第一存储器以及所述一个或多个第二存储器，所述第一存储器的性能高于所述第二存储器，其中，所述一个或多个第一存储器中至少有一个第一存储器位于第一存储节点，所述一个或多个第二存储器中至少有一个第二存储器位于第二存储节点。所述处理器还用于控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移。所述接口用于与所述第一存储节点以及所述第二存储节点通信。

[0032] 第四方面提供的管理节点基于多种性能的存储器创建内存池，并且这些多种性能的存储器位于不同的存储节点上，从而实现了跨节点的，融合了不同性能的存储器的内存

池，使得各种类型的存储器(无论是内存还是硬盘)都能够作为存储资源为上层提供存储服务，从而更好地发挥其性能优势。由于所述内存池中包含了不同性能的存储器，所以可以控制数据基于其访问频率在不同性能的存储器之间迁移，既能够在数据的访问频率较高时迁移至高性能的存储器以提高读数据的效率，也可以在数据的访问频率较低时迁移至低性能的存储器以节省高性能的存储器的存储空间。另外本申请中的内存池为计算节点或LUN提供存储空间，它改变内存资源以处理器为中心的架构。

[0033] 在一种实现中，所述处理器还用于通过所述接口获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。所述处理器具体用于根据所述状态信息创建所述内存池。

[0034] 在一种实现中，所述内存池的存储空间包括若干个页，一个页在内存池中的全局地址映射到所述页的物理地址；其中，所述页的全局地址用于指示所述页在所述内存池中的位置，所述页的物理地址用于指示为所述页分配的物理空间在存储节点内的存储器内的位置。

[0035] 在一种实现中，所述数据存储在所述第一存储器，所述处理器具体用于指示所述第一存储节点获取所述数据的访问频率；以及指示所述第一存储节点在所述访问频率低于设定的频率阈值时将所述数据迁移至所述内存池中的第二存储器。

[0036] 第五方面提供了一种计算机可读存储介质，所述存储介质中存储有程序指令，所述程序指令用于执行以下方法：创建内存池以提供存储数据的服务，所述内存池包括一个或多个第一存储器以及一个或多个第二存储器，所述第一存储器的性能高于所述第二存储器，其中，所述一个或多个第一存储器中至少有一个第一存储器位于第一存储节点，所述一个或多个第二存储器中至少有一个第二存储器位于第二存储节点；以及控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移。

[0037] 在一种实现中，还包括获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。所述创建内存池具体包括：基于所述状态信息创建所述内存池。

[0038] 在一种实现中，所述内存池的存储空间包括若干个页，一个页在内存池中的全局地址映射到所述页的物理地址；其中，所述页的全局地址用于指示所述页在所述内存池中的位置，所述页的物理地址用于指示为所述页分配的物理空间在存储节点内的存储器内的位置。

[0039] 在一种实现中，所述控制所述数据在所述内存池中的所述第一存储器与所述第二存储器之间迁移具体包括：指示所述第一存储节点获取所述数据的访问频率；指示所述第一存储节点在所述访问频率低于设定的频率阈值时将所述数据迁移至所述第二存储器。

[0040] 第六方面提供了一种计算程序产品，所述计算机程序产品包括：计算机程序代码，当所述计算机程序代码被运行时，使得上述各方面中由管理节点或者计算节点执行的方法被执行。

附图说明

[0041] 图1为本实施例提供的存储系统的系统架构示意图；

[0042] 图2是本实施例提供的存储节点的结构示意图；

[0043] 图3是本实施例提供的IO控制器的结构示意图；

[0044] 图4是本实施例提供的内存池的架构示意图；

[0045] 图5是本实施例提供的内存池所包含各层级的存储器的示意图；

[0046] 图6是本实施例提供的另一种内存池的架构示意图；

[0047] 图7是本实施例提供的另一种内存池的架构示意图；

[0048] 图8是本实施例提供的存储池的架构示意图；

[0049] 图9是本实施例提供的写数据方法的流程示意图；

[0050] 图10是本实施例提供的另一种写数据方法的流程示意图；

[0051] 图11是本实施例提供的读数据方法的流程示意图；

[0052] 图12是本实施例提供的管理节点的一种结构示意图；

[0053] 图13是本实施例提供的管理节点的另一种结构示意图。

具体实施方式

[0054] 下面将结合附图，对本发明实施例中的技术方案进行描述。

[0055] 本发明实施例描述的网络架构以及业务场景是为了更加清楚的说明本发明实施例的技术方案，并不构成对于本发明实施例提供的技术方案的限定，本领域普通技术人员可知，随着网络架构的演变和新业务场景的出现，本发明实施例提供的技术方案对于类似的技术问题，同样适用。

[0056] 本实施例提供的存储系统包括计算节点集群和存储节点集群。其中，计算节点集群包括一个或多个计算节点100(图1中示出了两个计算节点100，但不限于两个计算节点100)。计算节点100是用户侧的一种计算设备，如服务器、台式计算机等。在硬件层面，计算节点100中设置有处理器和内存(图1中未示出)。在软件层面，计算节点100上运行有应用程序(application)101(简称应用)和客户端程序102(简称客户端)。应用101是对用户呈现的各种应用程序的统称。客户端102用于接收由应用101触发的数据访问请求，并且与存储节点20交互，向存储节点20发送所述数据访问请求。客户端102还用于接收来自存储节点的数据，并向应用101转发所述数据。可以理解的是，当客户端102是软件程序时，客户端102的功能由计算节点100所包含的处理器运行内存中的程序来实现。客户端102也可以由位于计算节点100内部的硬件组件来实现。计算节点集群中的任意一个客户端102可以访问存储节点集群中的任意一个存储节点20。

[0057] 存储节点集群包括一个或多个存储节点20(图1中示出了三个存储节点20，但不限于三个存储节点20)，各个存储节点20之间可以互联。存储节点如服务器、台式计算机或者存储阵列的控制器、硬盘框等。在功能上，存储节点20主要用于对数据进行计算或处理等。另外，所述存储节点集群还包括管理节点(图1未示出)。管理节点用于创建并管理内存池。各个存储节点20从存储节点中选举出一个节点让它承担管理节点的职能。管理节点可以与任意一个存储节点20通信。

[0058] 在硬件上，如图1所示，存储节点20至少包括处理器、存储器和IO控制器201。其中，处理器202是中央处理器(central processing unit，CPU)，用于处理来自存储节点20外部的数据，或者存储节点20内部生成的数据。存储器，是指用于存储数据的装置，它可以是内存，也可以是硬盘。内存是指与处理器直接交换数据的内部存储器，它可以随时读写数据，

而且速度很快，作为操作系统或其他正在运行中的程序的临时数据存储器。内存包括至少两种存储器，例如内存既可以是随机存取存储器，也可以是只读存储器(Read Only Memory,ROM)。举例来说，随机存取存储器可以是动态随机存取存储器(Dynamic Random Access Memory,DRAM)，也可以是存储级存储器(Storage Class Memory,SCM)。DRAM是一种半导体存储器，与大部分随机存取存储器(Random Access Memory,RAM)一样，属于一种易失性存储器(volatile memory)设备。SCM是一种同时结合传统储存装置与存储器特性的复合型储存技术，存储级存储器能够提供比硬盘更快速的读写速度，但运算速度上比DRAM慢，在成本上也比DRAM更为便宜。

[0059] 然而，DRAM和SCM在本实施例中只是示例性的说明，内存还可以包括其他随机存取存储器，例如静态随机存取存储器(Static Random Access Memory,SRAM)等。而对于只读存储器，举例来说，可以是可编程只读存储器(Programmable Read Only Memory,PROM)、可抹除可编程只读存储器(Erasable Programmable Read Only Memory,EPROM)等。另外，内存还可以是双列直插式存储器模块或双线存储器模块(Dual In-line Memory Module，简称DIMM)，即由动态随机存取存储器(DRAM)组成的模块。在图2以及后面的描述中，均以DRAM和SCM为例进行说明，但不代表存储节点20不包含其他类型的存储器。

[0060] 本实施例中的存储器还可以是硬盘。与内存203不同的是，硬盘读写数据的速度比内存慢，通常用于持久性地存储数据。以存储节点20a为例，其内部可以设置一个或多个硬盘；或者，在存储节点20a的外部还可以挂载一个硬盘框(如图2所示)，在硬盘框中设置多个硬盘。无论哪一种部署方式，这些硬盘都可以视作存储节点20a所包含的硬盘。硬盘类型为固态硬盘、机械硬盘，或者其他类型的硬盘。类似的，存储节点集群中的其他存储节点，如存储节点20b、存储节点20c也可以包含各种类型的硬盘。一个存储节点20中可以包含一个或多个同一种类型的存储器。

[0061] 本实施例中的内存池中所包括的硬盘，也可以具有内存接口，处理器可以直接对其进行访问。

[0062] 请参考图2，图2是存储节点20的内部结构示意图。在实际应用中，存储节点20可以是服务器也可以是存储阵列。如图2所示，除了处理器和存储器之外，存储节点20还包括IO控制器。由于内存的访问时延很低，操作系统调度以及软件本身的开销就可能成为数据处理的瓶颈。为了减少软件开销，本实施例引入硬件组件IO控制器，用于将IO的访问进行硬件化，降低CPU调度和软件栈的影响。

[0063] 首先，存储节点20有自己的IO控制器22，用于与计算节点100通信，还用于与其他存储节点通信。具体的，存储节点20可通过IO控制器22接收来自计算节点100的请求，或者通过IO控制器22向计算节点100发送请求，存储节点20也可以通过IO控制器22向存储节点30发送请求，或者通过IO控制器22接收来自存储节点30的请求。其次，存储节点20内的各个内存之间可以通过IO控制器22通信，也可以通过IO控制器22与计算节点100通信。最后，如果存储节点20所包含的各个硬盘位于存储节点20内部，那么这些硬盘之间可以通过IO控制器22通信，也可以通过IO控制器22与计算节点100通信。如果所述各个硬盘位于存储节点20外接的硬盘框中，那么硬盘框中设置有IO控制器24，IO控制器24用于与IO控制器22通信，硬盘可以通过IO控制器24向IO控制器22发送数据或指令，也可以通过IO控制器24接收IO控制器22发送的数据或指令。另外，存储节点20还可以包括总线(图2中未示出)用于存储节点20

内部各组件之间的通信。

[0064] 请参考图3，图3是IO控制器的结构示意图，以IO控制器22为例，它包括通信单元220和计算单元2211，其中，通信单元提供高效的网络传输能力，用于外部或内部通信，这里以网络接口控制器(network interface controller,NIC)为例。计算单元是一个可编程的电子部件，用于对数据进行计算处理等，本实施例以数据处理单元(data processing unit,DPU)为例予以说明。DPU具有CPU的通用性和可编程性，但更具有专用性，可以在网络数据包，存储请求或分析请求上高效运行。DPU通过较大程度的并行性(需要处理大量请求)与CPU区别开来。可选的，这里的DPU也可以替换成图形处理单元(graphics processing unit,GPU)、嵌入式神经网络处理器(neural-network processing units,NPU)等处理芯片。DPU用于提供针对内存池的数据卸载服务，例如地址索引或地址查询功能，分区功能以及对数据进行过滤、扫描等操作。当IO请求通过NIC进行存储节点20之后，直接在计算单元221上进行处理，绕过了存储节点20内部的CPU和操作系统，压薄了软件栈，减少了CPU调度的影响。以执行读IO请求为例，当NIC接收计算节点100发送的所述读IO请求后，DPU可直接在索引表中查询所述读IO请求所述对应的信息。另外IO控制器22还包括DRAM 222，DRAM 222在物理上和图2所描述的DRAM一致，只是这里的DRAM 222是属于IO控制器22自己的内存，用于临时存储经过IO控制器22的数据或指令，它不构成内存池的一部分。另外，IO控制器22还可以将DRAM 222映射至计算节点100，使得DRAM 222的空间对计算节点100可见，由此将IO访问转化为基于内存语义的访问。IO控制器24的结构与功能与IO控制器22类似，这里不再一一展开描述。

[0065] 下面介绍本实施例提供的内存池，图4是内存池的架构示意图。所述内存池中包含有多种不同类型的存储器，每一种类型的存储器可以看作一个层级。每个层级的存储器的性能与其他层级的存储器的性能不同。本申请中的存储器的性能主要从运算速度和/或访问时延等方面进行考量。图5是本实施例提供的内存池所包含各层级的存储器的示意图。由图5所示，内存池由各个存储节点20中的存储器组成，其中，各存储节点中的DRAM位于所述内存池的第一层级，因为DRAM在各种类型的存储器性能最高。SCM的性能低于DRAM，因此各个存储节点中的SCM位于所述内存池的第二层级。进一步地，硬盘的性能低于SCM，所以各个存储节点中的硬盘位于所述内存池的第三层级。虽然图3中只示出了三种类型的存储器，但是按照前面的描述，在产品实践中存储节点20内部可部署多种不同类型的存储器，即各种类型的内存或硬盘均可以成为内存池的一部分，并且位于不同存储节点上的相同类型的存储器在所述内存池中属于同一层级。本申请不对内存池所包含的存储器的类型，以及层级的数量做任何限定。内存池的层级只是内部的划分，对上层应用来说是不感知的。需要注意的是，虽然各个存储节点上的同一种存储器处于同一层级，但是对某一个存储节点来说，使用其本地的DRAM的性能高于使用其他存储节点的DRAM的性能，同理，使用本地的SCM的性能高于使用其他存储节点的SCM的性能，等等。因此，对该存储节点来说，当需要分配某一层级的内存空间时，会优先分配其本地属于该层级的空间给用户，当本地空间不足时再从其他存储节点的同一层级中分配。

[0066] 在图4或图5所示的内存池包含了存储节点中的所有类型的存储器，然而在其他实施方式中，如图6所示，内存池可以只包含部分类型的存储器，例如只包含较高性能的存储器，例如DRAM和SCM，而排除硬盘等性能相对较低的存储器。

[0067] 本实施例提供的另一种内存池的网络架构可参考图7，如图7所示，在这种网络架构中，存储节点和计算节点集成在同一个物理设备中，本实施例中将所述集成的设备统一称为存储节点。应用部署在存储节点20内部，所以应用可直接通过存储节点20中的客户端触发写数据请求或读数据请求，由该存储节点20处理，或者发送给其他存储节点20处理。此时，客户端向本地存储节点20发送的读写数据请求具体是指客户端向处理器发送数据访问请求。除此之外，存储节点20所包含的组件及其功能与图6中的存储节点20类似，这里不再赘述。与图4至图6任一所示的内存池类似，这种网络架构下的内存池可以包含了存储节点中的所有类型的存储器，也可以只包含部分类型的存储器，例如只包含较高性能的存储器，例如DRAM和SCM，而排除硬盘等性能相对较低的存储器(如图7所示)。

[0068] 另外，在图4至图7所示的内存池中，并非存储节点集群中的每一个存储节点都必须为所述内存池贡献存储空间，所述内存池可以仅覆盖集群中的部分存储节点。在某些应用场景中，存储节点集群中还可以创建两个或两个以上的内存池，每个内存池覆盖多个存储节点，由这些存储节点为该内存池提供存储空间。不同内存池所述占用的存储节点可以重复也可以不重复。总而言之，本实施例中的内存池是在至少两个存储节点中建立的，其包含的存储空间来源于至少两种不同类型的存储器。

[0069] 当内存池仅包含存储集群中的较高性能的存储器(例如DRAM和SCM)时，管理节点还可以将存储集群中的较低性能的存储器(例如硬盘)构建成一个存储池。图8以图6所示的网络架构为例对存储池进行了描述，与内存池类似，图8所示的存储池也跨越了至少两个存储节点，其存储空间由所述至少两个存储节点中的一种或多种类型的硬盘构成。当存储集群中既包含内存池又包含存储池时，存储池用于持久化地存储数据，特别是访问频率较低的数据，而内存池则用于临时存储数据，特别是访问频率较高的数据。具体的，当内存池中存储的数据的数据量到达设定的阈值时，所述内存池中的部分数据将会被写入存储池中存储。可以理解的是，存储池也可以建立在图7所示的网络架构中，其实现原理与上面的描述类似。然而，存储池并非本申请讨论的重点，接下来继续讨论内存池。

[0070] 关于内存池的创建。各个存储节点20通过心跳通道定期向管理节点上报存储器的状态信息。管理节点可以有一个，也可以部署多个。它可以作为独立节点部署在存储节点集群中，也可以和存储节点20联合部署。换言之，由某一个或多个存储节点20承担管理节点的职能。存储器的状态信息包括但不限于：该存储节点所包含的各种存储器的类型、健康状态、每种存储器的总容量以及可用容量等等。管理节点根据采集的信息创建内存池，创建意味着将各个存储节点20所提供的存储空间集中起来作为内存池统一管理，因此内存池的物理空间来源于各个存储节点所包含的各种存储器。然而，在某些场景下，存储节点20可以根据自身情况，例如存储器的健康状态，选择性地提供存储器给内存池。换言之，有可能某些存储节点中的某些存储器并非内存池的一部分。

[0071] 采集完信息之后，管理节点需要对被纳入所述内存池的存储空间进行统一编址。经过统一编址，所述内存池的每段空间都有一个唯一的全局地址。所谓全局地址，它所指示的空间在内存池中是唯一的，并且每个存储节点20都知道该地址的含义。在给内存池的一段空间分配了物理空间之后，该空间的全局地址就拥有了其对应的物理地址，所述物理地址指示了该全局地址所代表的空间实际位于哪个存储节点的哪个存储器上，以及在该存储器中的偏移量，即物理空间的位置。这里的每段空间是指“页”，后面会详细介绍。在实际应

用中,为了保证数据的可靠性,往往采用纠删码(erasure coding,EC)校验机制或者多副本机制来实现数据冗余。EC校验机制是指将数据划分为至少两个数据分片,按照一定的校验算法计算所述至少两个数据分片的校验分片,当其中一个数据分片丢失时,可以利用另一个数据分片以及校验分片恢复数据。那么,对于所述数据而言,其全局地址是多个细粒度的全局地址的集合,每个细粒度的全局地址对应了一个数据分片/校验分片的物理地址。多副本机制是指存储至少两份相同的数据副本,并且,这至少两份数据副本存储在两个不同的物理地址中。当其中一份数据副本丢失时,可以使用其他数据副本恢复。因此,对于所述数据而言,其全局地址也是多个更细粒度的全局地址的集合,每个细粒度的全局地址对应了一个数据副本的物理地址。

[0072] 管理节点可以在创建内存池之后为各个全局地址分配物理空间,也可以在接收写数据请求时为所述写数据请求所对应的全局地址分配物理空间。各个全局地址与其物理地址之间的对应关系被记录在索引表中,管理节点将所述索引表同步给各个存储节点20。各个存储节点20存储所述索引表,以便后续读写数据时根据索引表查询全局地址对应的物理地址。

[0073] 在某些应用场景中,内存池并非直接将其存储空间暴露给计算节点100,而是将存储空间虚拟化为逻辑单元(logical unit,LU)提供给计算节点100使用。每个逻辑单元具有唯一的逻辑单元号(logical unit number,LUN)。由于计算节点100能直接感知到逻辑单元号,本领域技术人员通常直接用LUN代指逻辑单元。每个LUN具有LUN ID,用于标识所述LUN。此时,内存池以页为粒度为LUN提供存储空间,换句话说,当存储节点20向内存池申请空间时,内存池以一个页或者页的整数倍为其分配空间。一个页的大小可以是4KB,也可以是8KB等等,本申请不对页的大小进行限定。数据位于一个LUN内的具体位置可以由起始地址和该数据的长度(length)确定。对于起始地址,本领域技术人员通常称作逻辑块地址(logical block address,LBA)。可以理解的是,LUN ID、LBA和length这三个因素标识了一个确定的地址段,一个地址段可以索引到一个全局地址。为了保证数据均匀存储在各个存储节点20中,计算节点100通常采用分布式哈希表(Distributed Hash Table,DHT)方式进行路由,按照分布式哈希表方式,将哈希环均匀地划分为若干部分,每个部分称为一个分区,一个分区对应一个上面描述的地址段。计算节点100向存储节点20发送的数据访问请求,都会被定位到一个地址段上,例如从所述地址段上读取数据,或者往所述地址段上写入数据。

[0074] 在上面描述的应用场景中,计算节点100与存储节点20之间利用LUN语义进行通信。在另一种应用场景中,计算节点100与存储节点20之间利用内存语义进行通信。此时,IO控制器22将其DRAM的空间映射给计算节点100,使得计算节点100可以感知到所述DRAM的空间(本实施例中将其称为虚拟空间),对所述虚拟空间进行访问。在这种场景中,计算节点100发送给存储节点20的读/写数据请求不再携带LUN ID、LBA和length,而是其他逻辑地址,例如虚拟空间ID、虚拟空间的起始地址以及长度。在另一种应用场景中,IO控制器22可以将它管理的内存池中的空间映射给计算节点100,使得计算节点100可以感知到这部分空间,并且获得这部分空间所对应的全局地址。例如,存储节点20a中的IO控制器22用于管理内存池中由存储节点20a提供的存储空间,存储节点20b中的IO控制器22用于管理内存池中由存储节点20b提供的存储空间,存储节点20c中的IO控制器22用于管理内存池中由存储节点20c提供的存储空间,等等。因此整个内存池对计算节点100来说是可见的,那么计算节点

100在向存储节点发送待写入数据时，可以直接指定该数据的全局地址。

[0075] 下面以应用向内存池申请存储空间为例说明空间分配流程。一种情况下，应用是指存储节点的内部服务，例如，存储节点20a内部生成一个内存申请指令，所述内存申请指令包括申请的空间大小以及存储器的类型。为了便于理解，这里假设申请的空间为16KB，存储器的类型为SCM。简而言之，申请的空间大小是由存储的数据的大小决定的，而申请的存储器的类型是由该数据的冷热信息决定的。存储节点20a从存储的索引表中获取一段空闲的全局地址，例如地址区间为[000001-000004]，其中，地址为000001的空间为一个页。所谓空闲的全局地址，是指该全局地址尚未被任何数据占用。然后，存储节点20a查询本地的SCM是否拥有16KB的空闲空间，若有，则从本地分配空间给所述全局地址，若否，则继续查询其他存储节点20的SCM是否拥有16KB的空闲空间，该步骤可以通过向其他存储节点20发送查询指令来实现。由于其他存储节点20与存储节点20a在距离上有远近之分，为了降低时延，存储节点20a在本地不能支撑分配16KB的空闲空间的情况下，可以优先向距离近的存储节点20查询。待获得物理地址之后，存储节点20a将所述全局地址与所述物理地址的对应关系记录在索引表中，并将所述对应关系同步到其他存储节点。在确定物理地址之后，存储节点20a就可以使用所述物理地址对应的空间存储数据了。另一种情况下，应用是指计算节点100中的应用101，这种情况下，内存申请指令则是计算节点100生成之后发送给存储节点20a的。那么，用户可以通过计算节点100指定申请的空间大小以及存储器的类型。

[0076] 上述索引表的作用主要在于记录全局地址与分区ID的对应关系、以及全局地址与物理地址的对应关系，除此之外，还可以用于记录数据的属性信息。例如，全局地址为000001的数据所具有的冷热信息或数据常驻策略等等。后续可以根据这些属性信息实现数据在各种存储器之间的迁移，或者进行属性设置等。应理解，数据的属性信息只是索引表的一个可选项，并非必须记录。

[0077] 有新的存储节点加入存储节点集群时，管理节点收集节点更新信息，将新的存储节点纳入内存池，对该存储节点所包含的存储空间进行编址，从而生成新的全局地址，再刷新分区与全局地址之间的对应关系(因为无论是扩容还是缩容，分区的总数是不变的)。扩容也同样适用于某些存储节点增加了内存或硬盘的情况，管理节点定期收集各个存储节点所包含的存储器的状态信息，如果的新的存储器加入，将其纳入内存池，并对新的存储空间进行编址，从而生成新的全局地址，再刷新分区与全局地址之间的对应关系。同理，本实施例提供的内存池也支持缩容，只要更新全局地址与分区的对应关系即可。

[0078] 本实施例提供的内存池中的各个存储器均提供内存接口给处理器，使得处理器看到的是一段连续的空间，可以直接对内存池中的存储器进行读写操作。

[0079] 在本实施例的存储系统中，基于多种性能的存储器创建内存池，并且这些多种性能的存储器位于不同的存储节点上，从而实现了跨节点的，融合了不同性能的存储器的内存池，使得各种类型的存储器(无论是内存还是硬盘)都能够作为存储资源为上层提供存储服务，从而更好地发挥其性能优势。由于所述内存池中包含了不同性能的存储器，所以可以控制数据基于其访问频率在不同性能的存储器之间迁移，既能够在数据的访问频率较高时迁移至高性能的存储器以提高读数据的效率，也可以在数据的访问频率较低时迁移至低性能的存储器以节省高性能的存储器的存储空间。另外本申请中的内存池为计算节点或LUN提供存储空间，它改变内存资源以处理器为中心的架构。

[0080] 下面介绍执行一种写数据方法的过程。请参考图9，图9是本实施例提供的执行该方法的流程示意图。如图9所示，该方法包括如下步骤。

[0081] S101，计算节点100发送给存储节点一个写数据请求，所述写数据请求中携带待写入数据以及所述待写入数据的逻辑地址。在LUN语义的应用场景中，所述逻辑地址包括LUN ID、LBA和length，在内存语义的应用场景中，所述逻辑地址包括虚拟空间ID、虚拟空间的起始地址和length。存储节点的通信单元220接收所述写数据请求后，将所述写数据请求存储在DRAM 222中。

[0082] S102，计算单元221从DRAM 222中获取所述写数据请求，以所述逻辑地址为输入，按照一定算法输出key，所述key可以唯一定位到一个分区ID。

[0083] S103，计算单元221在索引表中查询所述分区ID对应的全局地址。

[0084] S104，计算单元221判断是否为所述全局地址分配物理地址，若否执行S105：为所述全局地址分配物理空间，并创建所述全局地址与物理地址之间的对应关系。具体的分配方式可参考前面的空间分配流程。若判断的结果为已经为所述全局地址分配了物理地址，则执行S106。

[0085] 如果采用多副本机制保证数据可靠性，那么意味着所述待写入数据需要存储多份副本在存储集群中，每份副本存储在不同的物理地址。每份副本的写入过程类似，所以这里以写入一份副本为例予以说明。

[0086] S106，计算单元221将所述待写入数据写入所述物理地址指示的物理空间的位置中。所述物理地址指示了所述物理空间位于的存储节点，所述存储节点内的存储器，以及所述存储器内的偏移量，所以IO控制器22可以直接根据该地址进行存储。举例来说，如果所述物理地址指示的物理空间位于存储节点内的SCM中，由IO控制器22执行数据写入的动作。如果所述物理地址指示的物理空间位于存储节点内的硬盘中，那么计算单元221通知通信单元220，由通信单元220将所述写数据请求发送给IO控制器24，由IO控制器24执行数据写入的动作。如果所述物理地址所指示的物理空间位于其他存储节点上，则计算单元221通知通信单元220，由通信单元220将所述待写入数据发送给其他存储节点，并指示该节点将所述待写入数据写入所述物理地址指示的物理空间的位置中。

[0087] 如果采用EC校验机制，那么在上述流程中，计算单元221从DRAM 222获取所述写数据请求中的所述待写入数据，将所述待写入数据划分成多个数据分片，计算生成所述多个数据分片的校验分片。每个数据分片或者校验分片都具有自己的逻辑地址，所述逻辑地址是所述写数据请求中携带的逻辑地址的子集。对于每个数据分片/校验分片，计算单元221以它的逻辑地址为输入，按照一定算法输出key，所述key可以唯一定位到一个分区ID。计算单元221在索引表中查询所述分区ID对应的全局地址，并进一步获取全局地址对应的物理地址，再将每个数据分片或校验分片存储在物理地址指示的空间的位置中。

[0088] 本实施例提供的另一种写数据方法，在该方法中，各个存储节点20中的IO控制器22将其管理的内存池的全局地址提供给计算节点100，使得计算节点100可以感知到内存池的空间，并且通过全局地址来访问存储节点20。此时，计算节点100发送给存储节点20的写数据请求中携带的是全局地址，而非逻辑地址了。请参考图10，图10是本实施例提供的执行该方法的流程示意图。如图10所示，该方法包括如下步骤。

[0089] S301，计算节点100发送给存储节点20写数据请求，所述写数据请求中携带待写入

数据以及所述待写入数据的全局地址。在计算节点100内部保存有一张关于内存池的全局地址的位图，所述位图记录了内存池中若干个页所对应的全局地址，以及该页的使用情况。例如某个页的全局地址对应的记录是“1”，那么代表该页已经存储了数据，如果某个页的全局地址对应的记录是“0”，那么代表该页尚未存储数据，是一个空闲页。因此，计算节点可以根据所述位图获知哪些全局地址所指示的存储空间已经存储有数据了，以及哪些全局地址所指示的存储空间是空闲的，在发送写数据请求时可以选择空闲的页的全局地址，将其携带在所述写数据请求中。具体的，存储节点20在执行完毕一个写数据请求之后，会给计算节点100发送一个响应消息，计算节点100可以根据所述响应消息，在所述位图中对该请求所对应的页的全局地址进行标记(设置为“1”)。存储节点20的通信单元220所述写数据请求后，将所述写数据请求存储在DRAM 222中。

[0090] 另外，由图1所示，存储节点集群中包含多个存储节点20，那么计算节点100在发送写数据请求时需要根据全局地址选择一个特定的存储节点20，由前面的描述可知，全局地址对应有物理地址，物理地址指明了所述全局地址指示的空间位于哪个存储节点的哪个存储器上。因此，对某一个特定的存储节点20来说，它只能管理它自己所拥有的存储器所对应的全局地址，对所述全局地址执行写入数据或读取数据的操作。如果该存储节点20接收到的待写入其他存储节点的数据，可以将其转发给其他存储节点，但这样处理时延会比较大。

[0091] 为了降低访问时延，管理节点对内存池进行编址时，可以在全局地址中嵌入一个或多个字节，该字节用于指示全局地址所指示的空间位于哪个存储节点。或者根据一定算法进行编址，使得每个全局地址对应一个唯一的存储节点。由此，计算节点100可以识别出全局地址所对应的存储节点，直接将写请求发送给该存储节点处理。

[0092] S302，计算单元221从DRAM 222中获取所述写数据请求，判断是否为所述全局地址分配物理地址，若否执行S303：为所述全局地址分配物理空间，并创建所述全局地址与物理地址之间的对应关系。具体的分配方式可参考前面的空间分配流程。若判断的结果为已经为所述全局地址分配了物理地址，则执行S304。

[0093] S304，计算单元221将所述待写入数据写入所述物理地址指示的物理空间中。该步骤可参考图9中S106的描述，这里不再赘述。

[0094] 另外，与图9描述的过程类似，在本实施例中同样可以采用多副本机制或EC校验机制来保证数据可靠性，该部分也可以参考图9的描述。当数据初次写入存储节点集群时，通常都存储在内存池的DRAM层。随着数据的访问频率逐渐变低，或者DRAM层的空间容量逐渐变小，存储节点集群会内部触发数据迁移，该过程对计算节点集群来说不感知的。数据迁移策略保存在管理节点中，由管理节点根据数据迁移策略控制数据迁移操作。数据迁移策略包括但不限于：执行数据迁移操作的触发条件，例如周期性地执行，或者当一定条件满足时执行。这里的一定条件可以是数据的访问频率高于与低于设定阈值，也可以是数据所位于的存储器的可用容量高于或低于设定阈值等等。所谓“控制”是指：管理节点指示各个存储节点20监控其存储的数据的访问频率，以及指示存储节点20对其存储的数据在各个存储器之间进行迁移。

[0095] 除了定期触发数据迁移操作之外，计算节点100在向存储节点发送写数据请求，可以在所述写数据请求中携带该数据的冷热信息(用于指示其访问频率)，存储节点在执行所述写数据请求时，一种执行方式是：首先将该数据写入DRAM，然后立即根据所述冷热信息执

行数据迁移操作，以将该数据从DRAM迁入与其冷热信息匹配的存储器。或者，存储节点也可以根据该数据的元数据结构或者逻辑地址等获得该数据的冷热信息，再根据所述冷热信息执行数据迁移操作。另一种执行方式是，存储节点直接根据所述冷热信息确定与其冷热信息匹配的存储器，通过IO控制器直接将所述数据写入所述存储器中。

[0096] 另外，计算节点100也可以在写数据请求中指定该数据的常驻策略，常驻策略是指某一种类型的数据需要长期存储在某种类型的存储器中。对于这类数据，一旦存储在指定的存储器中了，无论其访问频率是升高还是降低，都不会对其执行数据迁移操作。

[0097] 以位于DRAM层的目标数据为例，假设所述目标数据位于存储节点20a，存储节点20a将定期统计所述目标数据的访问频率，当所述访问频率低于DRAM层的访问阈值时，将所述目标数据迁移至SCM层或其他层。一种可选方案是，内存池的每层存储器具有一个访问阈值区间，当数据的访问频率高于该区间的最高值或数据的访问频率低于该区间的最低值时，则意味着该数据需要被迁移至与其访问频率匹配的层。另一种可选方案是，不设置每层存储器的访问阈值区间，仅将所述访问频率与设定的访问阈值进行比较，当低于所述访问阈值时则意味需要将性能更低的层。继续以所述目标数据为例，如果所述目标数据当前的访问频率跌落至硬盘层的访问频率区间内，则首先判断存储节点20a本地的硬盘是否具有空闲空间，若有，则将其迁移至存储节点20a本地的硬盘；否则将所述目标数据发送给其他存储节点，例如存储节点20b，并指示存储节点20b将所述目标数据写入其硬盘中。迁移前后，所述目标数据的全局地址不会发生变化，因为上层应用是不感知的，发生变化的是所述目标数据的物理地址。迁移完成后，各个存储节点20均在自己的索引表中更新所述目标数据的全局地址与物理地址的对应关系。

[0098] 除了按照数据的访问频率(又称为冷热程度)实现数据在各层之间的迁移之外，另一种迁移策略为根据各层的可用容量来实现数据迁移。我们知道越高层级的存储器的性能越好，成本也越高，其存储空间比低层的存储器更为珍贵。例如，当DRAM层的可用容量低于设定的容量阈值时，则DRAM层需要将其存储的一部分数据迁移至SCM层或其他层，以便腾挪出更多的空间容纳新写入的数据。至于选择哪部分数据往低层级存储器迁移，可以参考现有的内存淘汰算法，这里不再展开描述。同理，SCM层或者其他层也具有自己的容量阈值，当该层的可用容量低于所述容量阈值，则将存储的一部分数据迁移至其他层。

[0099] 前面提到，内存池以页为粒度对外提供存储空间。因此，在统计数据的访问频率时也可以以页为单位来统计，那么相应的，也是以页为单位实现各层级之间的数据迁移。然而，在产品实践中，应用往往需要在页之上再分配更小粒度的对象，比如一个数据项(item)。如果页的大小是4KB，那么一个数据项的大小是1KB、2KB或3KB(只要比页的尺寸更小即可)这样，以页粒度进行访问频率统计必然存在较大的不准确性，可能一个页中某些数据项是经常访问的，但是此页中其它的数据项却是几乎没有怎么访问过，如果按照页粒度的页面统计，将会使得此页会常驻到DRAM或者SCM介质上，导致大量空间的浪费。因此本实施例还提供了数据项粒度的访问频率统计，以数据项为粒度进行数据迁移，然后聚合冷热页面，这样可以实现更为高效的换入和换出性能。

[0100] 接下来介绍执行读数据请求方法的过程。请参考图11，图11是本实施例提供的执行读数据请求方法的流程示意图。如图11所示，该方法包括如下步骤。

[0101] S201，计算节点100发送给存储节点一个读数据请求，所述读数据请求中携带待读

取数据的逻辑地址，存储节点的IO控制器22接收所述读数据请求。在LUN语义的应用场景中，所述逻辑地址包括LUN ID、LBA和length，在内存语义的应用场景中，所述逻辑地址包括虚拟空间ID、虚拟空间的起始地址和length。存储节点的通信单元220接收所述读数据请求后，将所述写数据请求存储在DRAM 222中。

[0102] S202，计算单元221从DRAM 222中获取所述读数据请求，以所述逻辑地址为输入，按照一定算法输出key，所述key可以唯一定位到一个分区ID。

[0103] S203，计算单元221在索引表中查询所述分区ID对应的全局地址。

[0104] S204，计算单元221在索引表中查询所述全局地址对应的物理地址。

[0105] S205，计算单元221从所述物理地址指示的物理空间中读取所述待读取数据，由通信单元220将所述待读取数据返回给计算节点100。所述物理地址指示了所述物理空间位于的存储节点，所述存储节点内的存储器，以及所述存储器内的偏移量，所以计算单元221可以直接根据该地址进行读取。如果所述物理地址所指示的物理空间位于其他存储节点上，则将所述读数据请求发送给其他存储节点，并指示该节点从所述物理地址指示的物理空间中读取所述数据。

[0106] 如果采用多副本机制存储数据，那么存储节点可以按照上面的流程读取任意一个数据副本，将该数据副本发送给计算节点100。如果采用EC校验机制，那么存储节点需要按照上面的流程读取每个数据分片和校验分片，合并获得所述待读取数据，并对所述待读取数据进行验证，验证无误后再返回给计算节点100。可以理解的是，与图11所示的读数据方法是与图9所示的写数据方法对应的，所以该方法中的读数据请求携带的是待读取数据的逻辑地址，本实施例还提供了另一种读数据方法，该方法与图10所示的写数据方法对应，在该方法中，所述读数据请求携带的是待读取数据的全局地址，计算单元221可以直接根据所述全局地址，查询到物理地址，从而获取所述待读取数据。

[0107] 此外，图1至图8所示的内存池还支持数据预取机制。本领域技术人员可以理解，从性能较高的存储器读取数据的速度高于从性能较低的存储器读取数据的速度。因此，如果待读取数据在性能较高的存储器中命中，那么就不必再从性能较低的存储器中读取了，因此读取数据的效率较高。为了提高缓存的数据命中率，通常的做法是从性能较低的存储器中预先读取一段数据，写入性能较高的存储器中。那么，当计算节点100发送读数据请求要求读取这段数据时，由于该数据已经被提前读取到性能较高的存储器中了，因此IO控制器可以直接从所述性能较高的存储器中读取该数据。对于一段逻辑地址连续的数据来说，它们被一起读取的可能性较大。因此，在实践中通常根据逻辑地址来预取数据。数据预取的方式包括同步预取和异步预取。同步预取是指，当执行读数据请求而待读取数据在高层级的存储器中未命中时，根据本次待读取数据的逻辑地址，将与所述待读取数据的逻辑地址连续的数据从低层级的存储器中读取出来，写入高层级的存储器中。异步预取是指，当执行读数据请求而待读取数据在高层级的存储器中被命中时，根据本次待读取数据的逻辑地址，将与所述待读取数据的逻辑地址连续的数据从从低层级的存储器中读取出来，写入高层级的存储器中。

[0108] 结合图11，所述执行读数据请求的方法中还可以包含：

[0109] S206，计算单元221将与所述待读取数据的逻辑地址连续的其他数据迁移到高层级的存储器中。在S205中，计算单元221从物理地址指示的物理空间中读取了所述待读取数

据，所述待读取数据可能存储在高层级的存储器(例如DRAM)中，也可能存储在低层级的存储器(例如SCM)中。如果所述待读取数据存储在DRAM中，那么计算单元221在DRAM中命中。如果所述待读取数据存储在SCM中，那么计算单元221在DRAM中未命中。无论是哪种情况，计算单元221都可以将与所述待读取数据的逻辑地址连续的其他数据预取到DRAM中。

[0110] 具体的，计算单元221首先获取与所述待读取数据的逻辑地址连续的逻辑地址。为了描述方便，姑且将所述待读取数据的逻辑地址称为逻辑地址1，与逻辑地址1连续的逻辑地址称为逻辑地址2。计算单元221以逻辑地址2为输入，按照一定算法输出key，所述key可以唯一定位到一个分区ID。然后，计算单元221在索引表中查询所述分区ID对应的全局地址，以及所述全局地址对应的物理地址。最后，计算单元221再从所述物理地址指示的物理空间中读取所述其他数据。所述其他数据可能位于计算单元221本地的存储节点中，也可能位于其他存储节点中。如果所述物理地址所指示的物理空间位于其他存储节点上，则该节点从所述物理地址指示的物理空间中读取所述数据。

[0111] 同理，如果计算节点100发送给存储节点20的读数据携带的是全局地址，那么预取数据时则根据所述全局地址，将与所述全局地址连续的地址中存储的数据提前读取至高层级的存储器中。

[0112] 图12是本实施例提供的管理节点的一种结构示意图，管理节点包括了处理器401和存储器402。所述存储器402中存储有程序403。处理器401、存储器402和接口404之间通过系统总线405连接并完成相互间的通信。处理器401是单核或多核中央处理单元，或者为特定集成电路，或者为被配置成实施本发明实施例的一个或多个集成电路。存储器402可以为随机存取存储器(Random Access Memory，RAM)，也可以为非易失性存储器(non-volatile memory)，例如至少一个硬盘存储器。存储器402用于存储计算机执行指令。具体的，计算机执行指令中可以包括程序403。当管理节点运行时，处理器401运行所述程序403以执行下述方法。

[0113] 例如，创建内存池以提供存储数据的服务，所述内存池包括所述第一存储器和所述至少两个第二存储器；所述控制模块用于控制所述数据从所述第一存储器迁移至所述第二存储器，或者从所述第二存储器迁移至所述第一存储器。

[0114] 可选的，所述方法还包括获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。由此，管理节点在创建内存池时具体用于根据所述状态信息创建所述内存池。

[0115] 可选的，所述管理节点在控制所述数据从第一存储器迁移至所述第二存储器时具体包括：指示所述第一存储节点获取所述数据的访问频率；以及指示所述第一存储节点在所述访问频率低于设定的频率阈值时将所述数据迁移至所述第二存储器。

[0116] 图13是本实施例提供的管理节点的另一种结构示意图，所述管理节点包括创建模块501和控制模块502。其中，创建模块501用于创建内存池以提供存储数据的服务，所述内存池包括所述第一存储器和所述至少两个第二存储器。控制模块502用于控制所述数据从所述第一存储器迁移至所述第二存储器，或者从所述第二存储器迁移至所述第一存储器。

[0117] 可选的，创建模块501还用于获取存储器的状态信息，所述状态信息包括所述第一存储器的类型和容量，以及所述第二存储器的类型和容量。创建模块501在创建内存池时具体用于根据所述状态信息创建所述内存池。

[0118] 可选的，控制模块502在控制所述数据从第一存储器迁移至所述第二存储器时具体用于：指示所述第一存储节点获取所述数据的访问频率；以及指示所述第一存储节点在所述访问频率低于设定的频率阈值时将所述数据迁移至所述第二存储器。

[0119] 在实践中，创建模块501和控制模块502的功能均可由图12所示的处理器401执行程序403来实现，也可由处理器401独立实现。

[0120] 在上述实施例中，可以全部或部分地通过软件、硬件、固件或者其任意结合来实现。当使用软件实现时，可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机指令时，全部或部分地产生按照本发明实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中，或者从一个计算机可读存储介质向另一个计算机可读存储介质传输，例如，所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如：同轴电缆、光纤、数据用户线(Digital Subscriber Line，DSL))或无线(例如：红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质(例如：软盘、硬盘、磁带)、光介质(例如：数字通用光盘(Digital Versatile Disc，DVD))、或者半导体介质(例如：固态硬盘(Solid State Disk，SSD))等。

[0121] 本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成，也可以通过程序来指令相关的硬件完成，所述的程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，磁盘或光盘等。

[0122] 在本申请的各个实施例中，如果没有特殊说明以及逻辑冲突，不同的实施例之间的术语和/或描述具有一致性、且可以相互引用，不同的实施例中的技术特征根据其内在的逻辑关系可以组合形成新的实施例。

[0123] 本申请中，“至少一个”是指一个或者多个，“多个”是指两个或两个以上。“和/或”，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的情况，其中A，B可以是单数或者复数。在本申请的文字描述中，字符“/”，一般表示前后关联对象是一种“或”的关系；在本申请的公式中，字符“/”，表示前后关联对象是一种“相除”的关系。

[0124] 可以理解的是，在本申请的实施例中涉及的各种数字编号仅为描述方便进行的区分，并不用来限制本申请的实施例的范围。上述各过程的序号的大小并不意味着执行顺序的先后，各过程的执行顺序应以其功能和内在逻辑确定。

[0125] 以上所述为本申请提供的实施例，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。

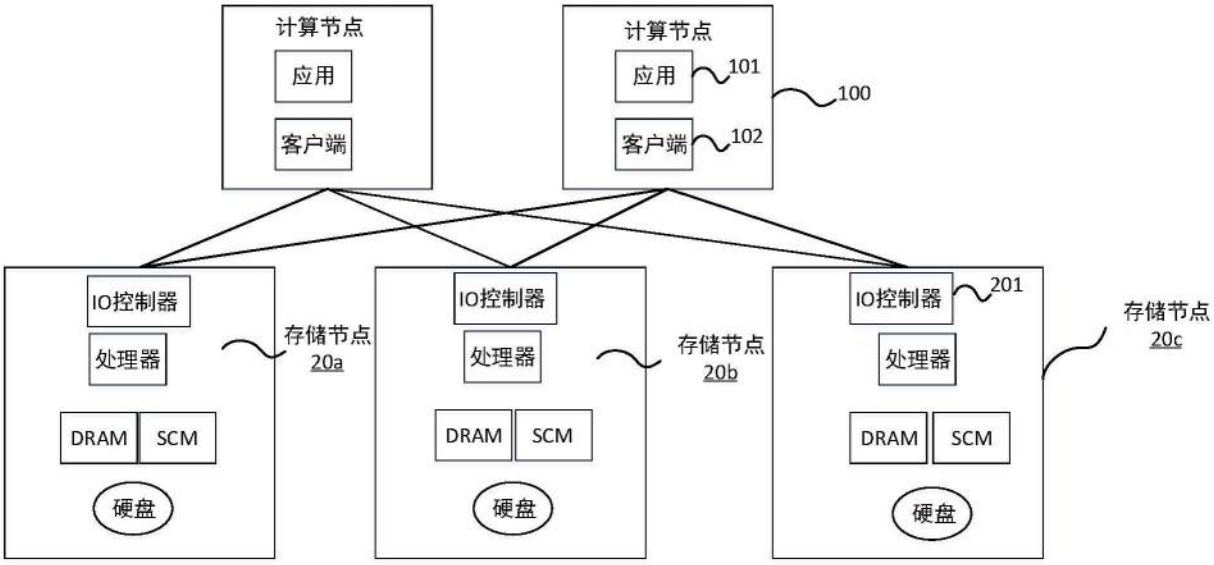

计算节点
应用
客户端
计算节点
应用
101
客户端
102
100
IO控制器
处理器
DRAM
SCM
硬盘
存储节点
20a
IO控制器
处理器
DRAM
SCM
硬盘
存储节点
20b
IO控制器
201
处理器
DRAM
SCM
硬盘
存储节点
20c

图1

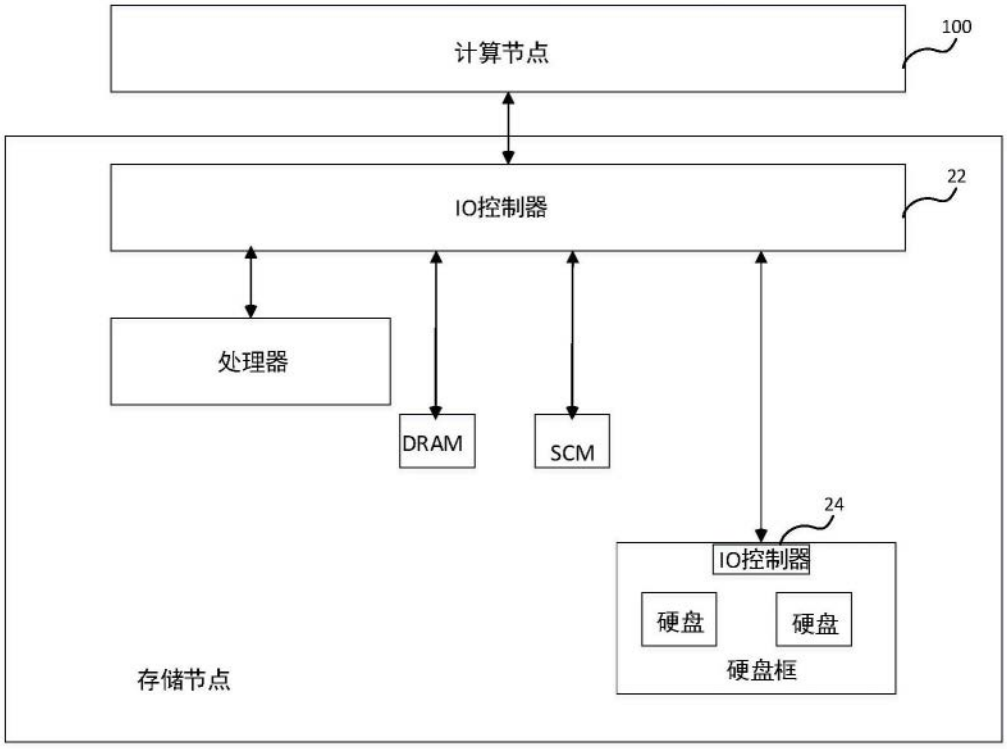

计算节点
100
IO控制器
22
处理器
DRAM
SCM
24
IO控制器
硬盘
硬盘
硬盘框
存储节点

图2

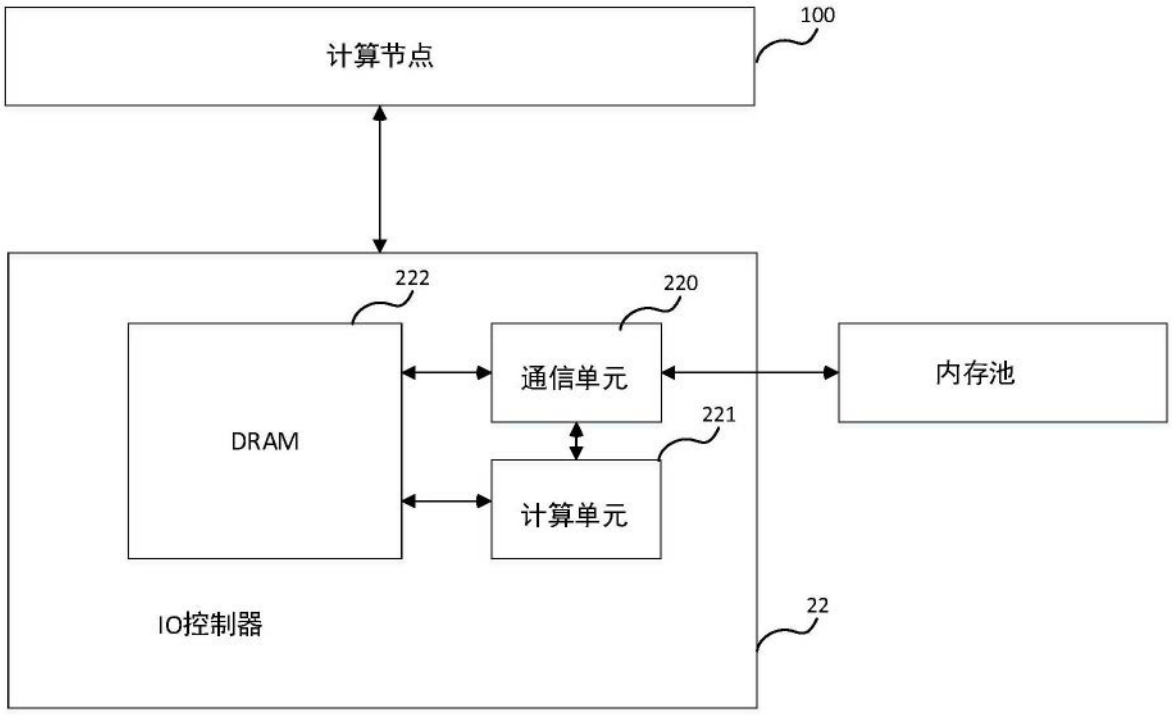

计算节点
100
222
220
通信单元
内存池
DRAM
221
计算单元
IO控制器
22

图3

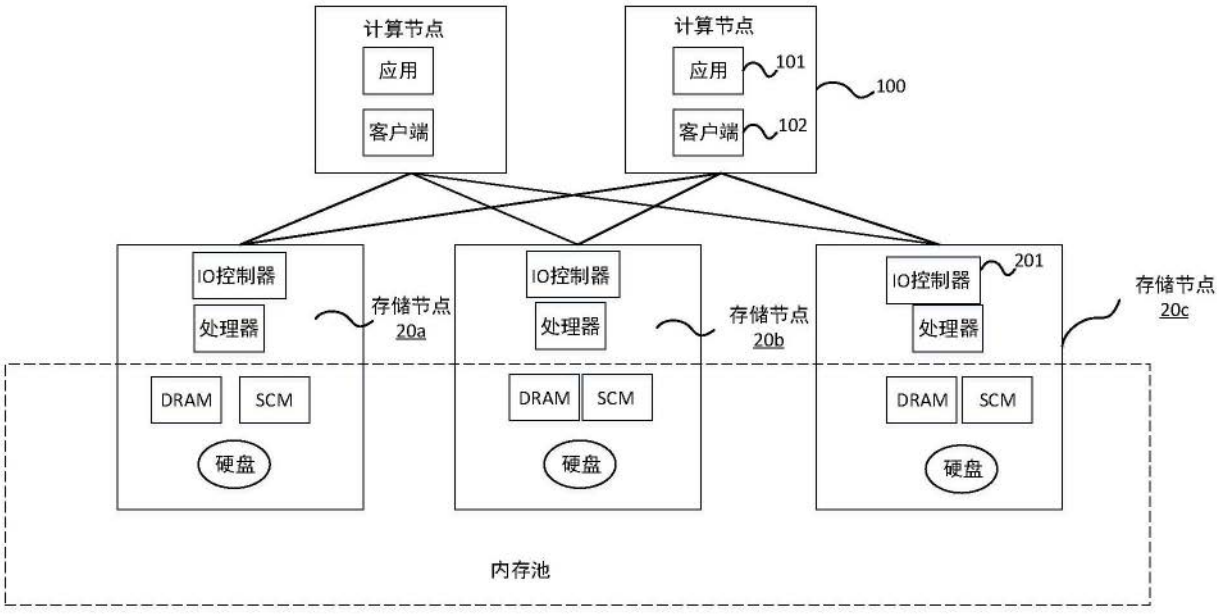

计算节点
应用
客户端
计算节点
应用
101
100
客户端
102
IO控制器
处理器
存储节点
20a
DRAM
SCM
硬盘
IO控制器
处理器
存储节点
20b
DRAM
SCM
硬盘
IO控制器
201
处理器
存储节点
20c
DRAM
SCM
硬盘
内存池

图4

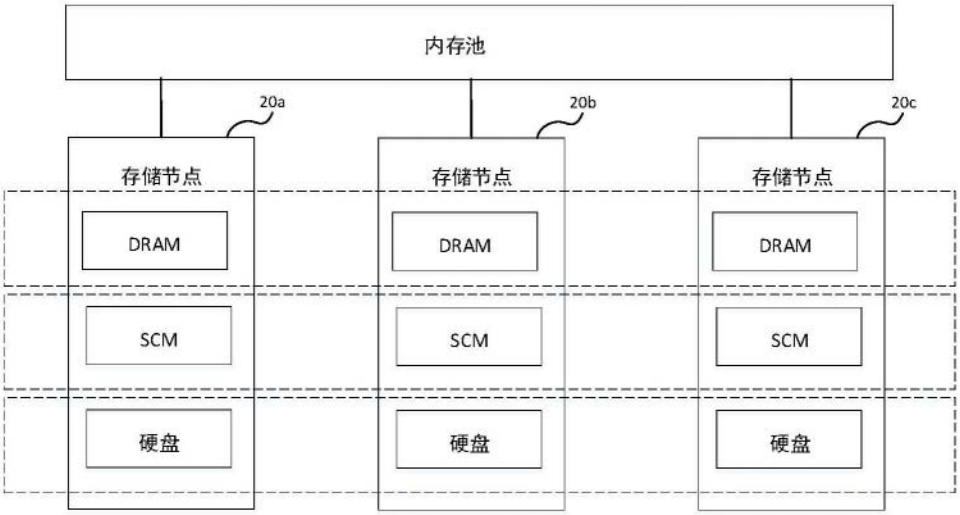

内存池
20a
20b
20c
存储节点
存储节点
存储节点
DRAM
DRAM
DRAM
SCM
SCM
SCM
硬盘
硬盘
硬盘

图5

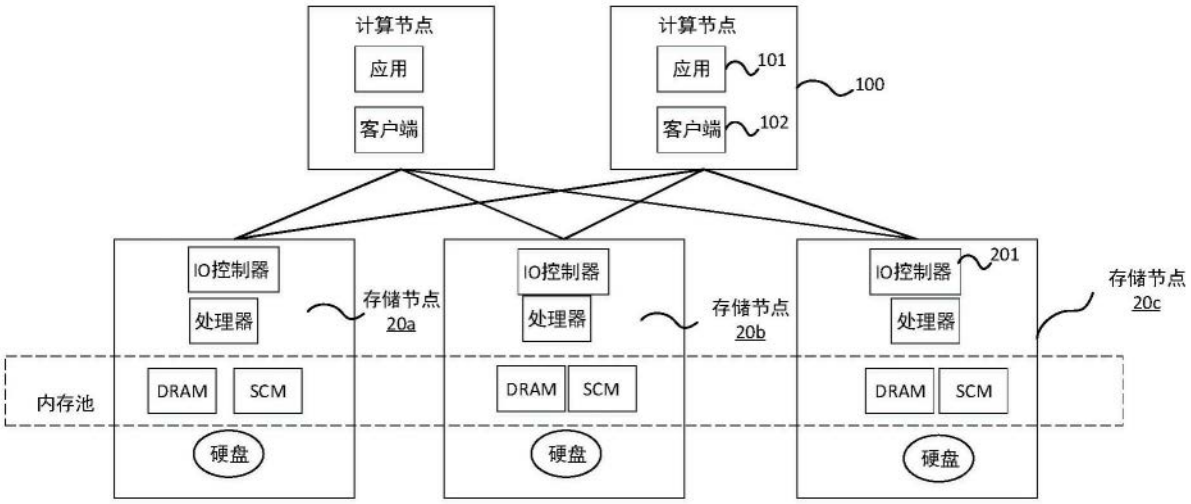

计算节点
应用
客户端
计算节点
应用
101
100
客户端
102
IO控制器
处理器
存储节点
20a
IO控制器
处理器
存储节点
20b
IO控制器
201
处理器
存储节点
20c
内存池
DRAM
SCM
DRAM
SCM
DRAM
SCM
硬盘
硬盘
硬盘

图6

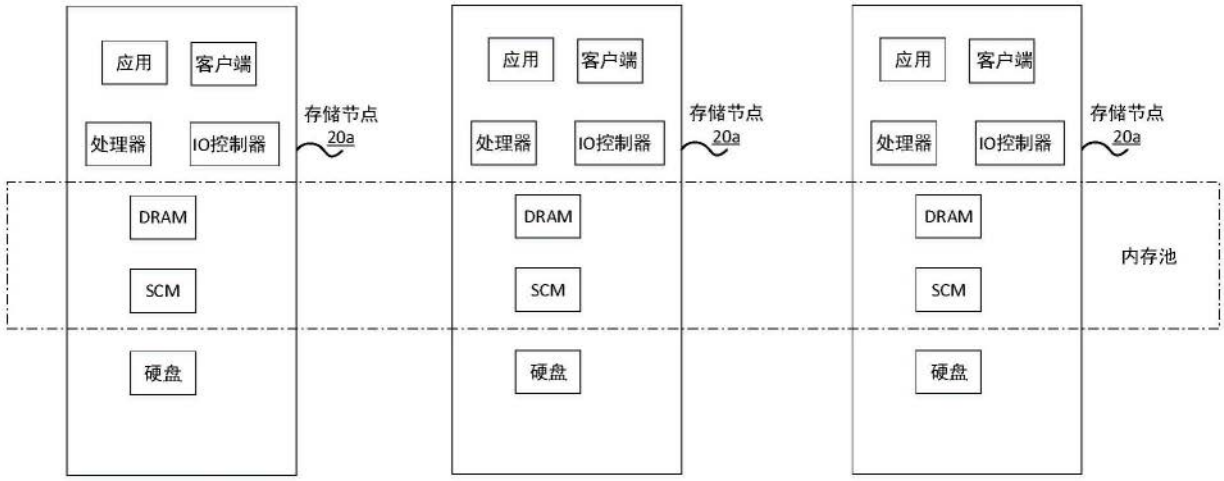

应用
客户端
处理器
IO控制器
存储节点
20a
DRAM
SCM
硬盘
内存池

图7

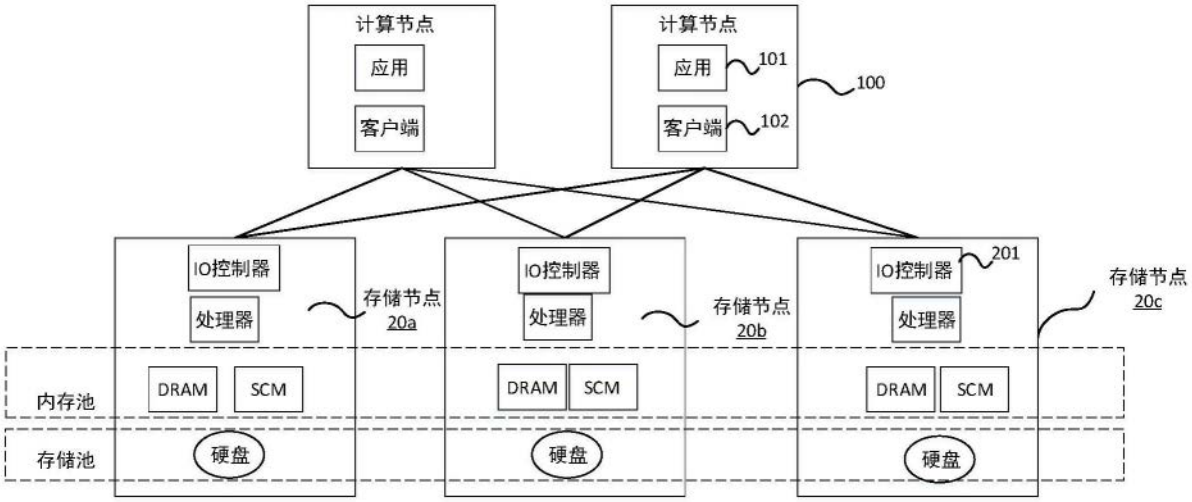

计算节点
应用
客户端
101
100
102
IO控制器
处理器
存储节点
20a
存储节点
20b
201
存储节点
20c
内存池
DRAM
SCM
存储池
硬盘

图8

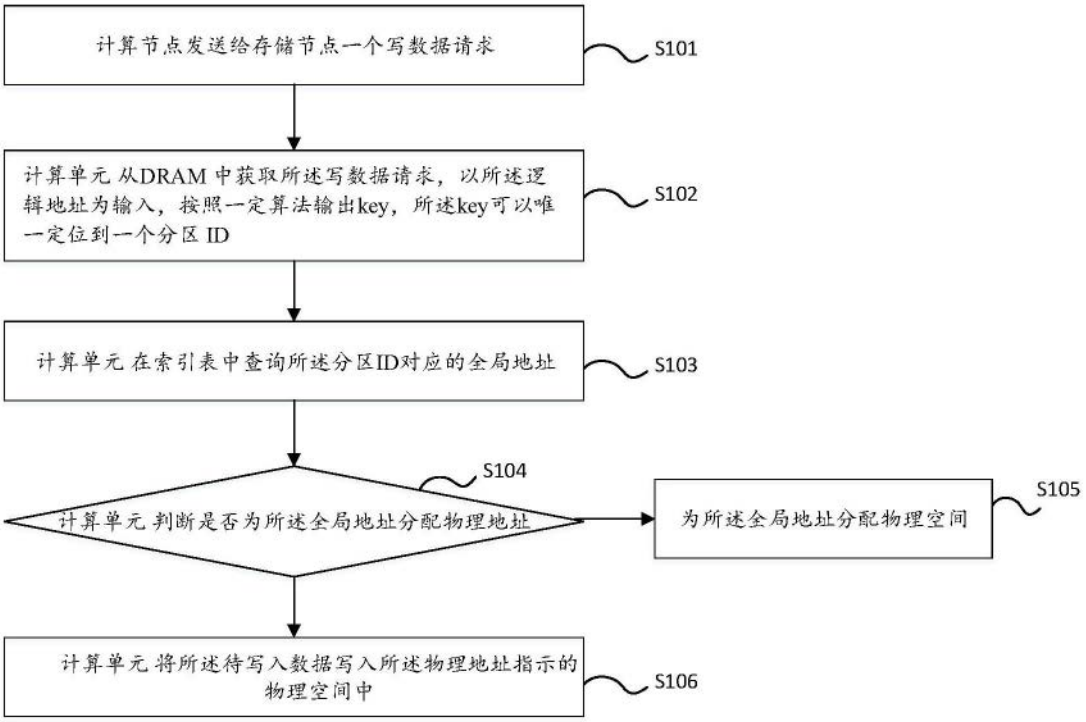

计算节点发送给存储节点一个写数据请求
S101
计算单元 从DRAM 中获取所述写数据请求，以所述逻辑地址为输入，按照一定算法输出key，所述key可以唯一定位到一个分区 ID
S102
计算单元 在索引表中查询所述分区ID对应的全局地址
S103
S104
计算单元 判断是否为所述全局地址分配物理地址
为所述全局地址分配物理空间
S105
计算单元 将所述待写入数据写入所述物理地址指示的物理空间中
S106

图9

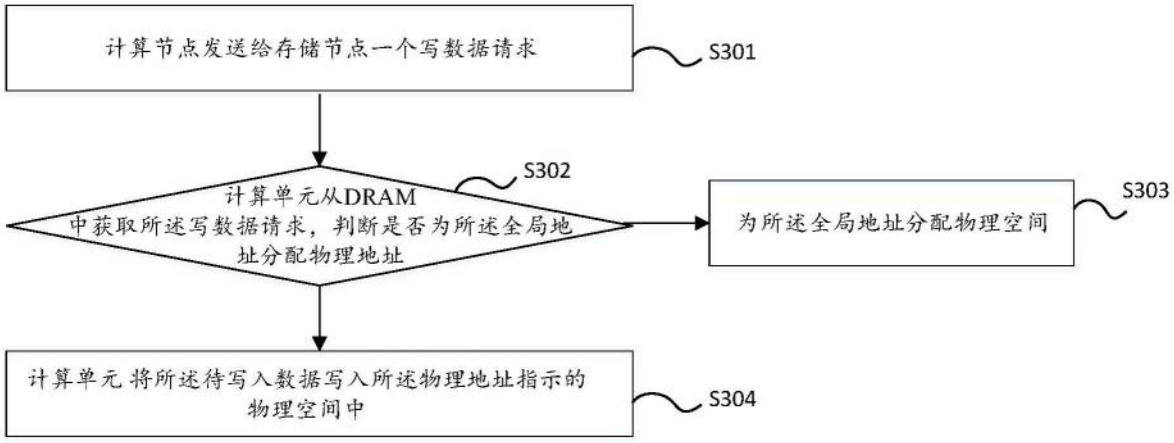

计算节点发送给存储节点一个写数据请求
S301
S302
计算单元从DRAM 中获取所述写数据请求，判断是否为所述全局地址分配物理地址
为所述全局地址分配物理空间
S303
计算单元 将所述待写入数据写入所述物理地址指示的物理空间中
S304

图10

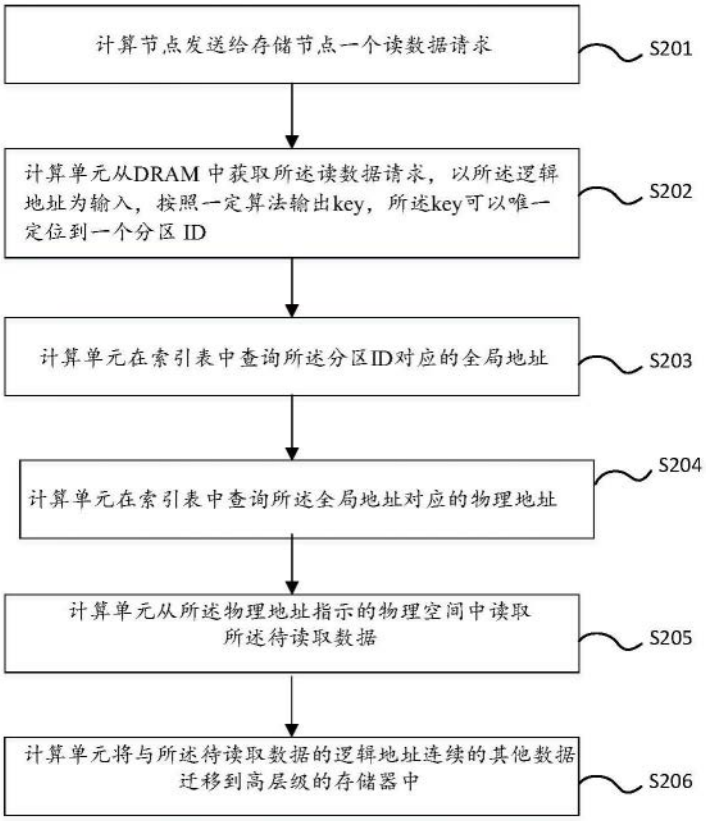

计算节点发送给存储节点一个读数据请求
S201
计算单元从DRAM中获取所述读数据请求，以所述逻辑地址为输入，按照一定算法输出key，所述key可以唯一定位到一个分区ID
S202
计算单元在索引表中查询所述分区ID对应的全局地址
S203
计算单元在索引表中查询所述全局地址对应的物理地址
S204
计算单元从所述物理地址指示的物理空间中读取所述待读取数据
S205
计算单元将与所述待读取数据的逻辑地址连续的其他数据迁移到高层级的存储器中
S206

图11

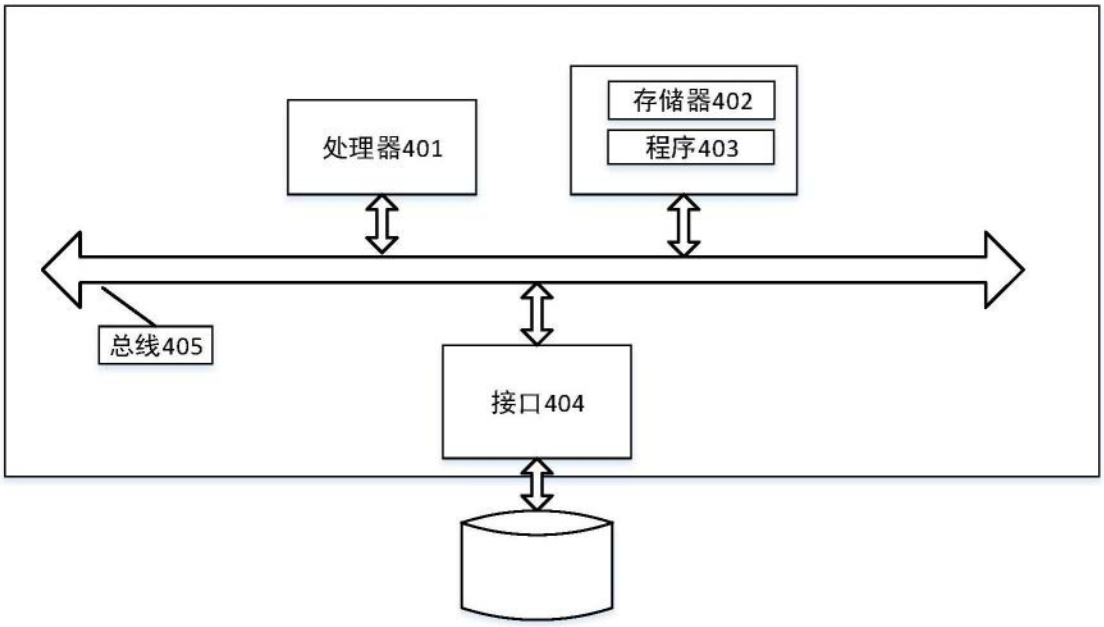

处理器401
存储器402
程序403
总线405
接口404

图12

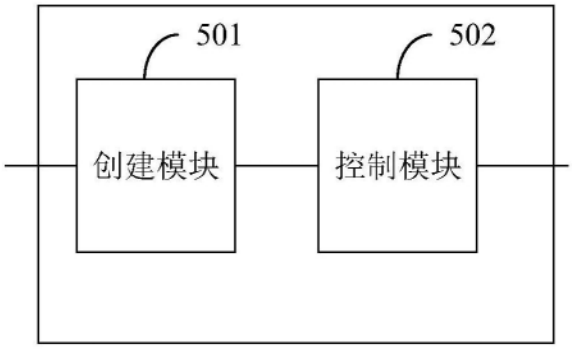


图13