(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116032930 A
(43)申请公布日 2023.04.28

(21)申请号 202111258240.8

(22)申请日 2021.10.27

(71)申请人 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 覃国

(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 熊永强 李稷芳

(51)Int.Cl.
H04L 67/1004(2022.01)
H04L 69/08(2022.01)

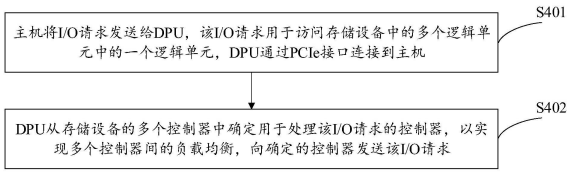
权利要求书2页 说明书15页 附图8页

(54)发明名称

网络存储方法、存储系统、数据处理单元及计算机系统

(57)摘要

本申请公开了一种网络存储方法、存储系统、数据处理单元及计算机系统。该网络存储方法用于存储系统，存储系统包括主机、数据处理单元和存储设备，数据处理单元通过PCIe接口连接到主机，存储设备包括多个控制器和多个逻辑单元；该方法包括：主机将输入/输出请求发送给数据处理单元，该输入/输出请求用于访问多个逻辑单元中的其中一个逻辑单元；数据处理单元从多个控制器中确定用于处理该输入/输出请求的控制器，以实现多个控制器之间的负载均衡，向确定的控制器发送该输入/输出请求。主机无需安装多路径软件，由数据处理单元负责将主机下发的I/O请求均衡分发到不同控制器，能够减轻主机中CPU的压力。



CN 116032930 A

1.一种网络存储方法,其特征在于,用于存储系统,所述存储系统包括主机、数据处理单元和存储设备,所述数据处理单元通过快捷外围部件互连标准PCIe接口连接到所述主机,所述存储设备包括多个控制器和多个逻辑单元;所述方法包括:

所述主机将输入/输出请求发送给所述数据处理单元,所述输入/输出请求用于访问所述多个逻辑单元中的一个逻辑单元;

所述数据处理单元从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡,向确定的所述控制器发送所述输入/输出请求。

2.根据权利要求1所述的方法,其特征在于,所述方法还包括:

所述数据处理单元与所述主机之间通过NVMe协议进行通信,所述数据处理单元与所述存储设备中的控制器之间通过NVMe-oF协议进行通信。

3.根据权利要求1或2所述的方法,其特征在于,所述数据处理单元从所述多个控制器中确定用于处理所述输入/输出请求的控制器,包括:

所述数据处理单元根据哈希算法,从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡。

4.一种存储系统,其特征在于,所述存储系统包括主机、数据处理单元和存储设备,所述数据处理单元通过PCIe接口连接到所述主机,所述存储设备包括多个控制器和多个逻辑单元;

所述主机用于,将输入/输出请求发送给所述数据处理单元,所述输入/输出请求用于访问所述多个逻辑单元中的一个逻辑单元;

所述数据处理单元用于,从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡,向确定的所述控制器发送所述输入/输出请求。

5.根据权利要求4所述的系统,其特征在于,所述数据处理单元还用于:

通过NVMe协议与所述主机进行通信;

通过NVMe-oF协议与所述存储设备中的控制器进行通信。

6.根据权利要求4或5所述的系统,其特征在于,所述从所述多个控制器中确定用于处理所述输入/输出请求的控制器,包括:

根据哈希算法,从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡。

7.根据权利要求4或5所述的系统,其特征在于,所述数据处理单元还用于:

向所述存储设备发送上报命令,所述上报命令指示所述存储设备将所述主机对应的逻辑单元的信息发送给所述数据处理单元;

接收所述存储设备中的控制器发送的所述主机对应的逻辑单元的信息;

根据所述主机对应的逻辑单元的信息,为所述主机对应的逻辑单元生成对应的设备,将生成的所述设备的信息发送给所述主机。

8.根据权利要求7所述的系统,其特征在于,所述主机还用于:

根据所述数据处理单元发送的所述设备的信息,生成所述设备对应的虚拟存储设备,所述虚拟存储设备用于提供给所述主机中的应用进行访问。

9.一种数据处理单元,其特征在于,所述数据处理单元用于:

接收主机发送的输入/输出请求,其中,所述数据处理单元通过PCIe接口连接到所述主机,所述输入/输出请求用于访问存储设备中的一个逻辑单元,所述存储设备包括多个逻辑单元和多个控制器;

从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡,向确定的所述控制器发送所述输入/输出请求。

10.根据权利要求9所述的数据处理单元,其特征在于,所述数据处理单元还用于:

通过NVMe协议与所述主机进行通信;

通过NVMe-oF协议与所述存储设备中的控制器进行通信。

11.根据权利要求9或10所述的数据处理单元,其特征在于,所述数据处理单元还用于:

根据哈希算法,从所述多个控制器中确定用于处理所述输入/输出请求的控制器,以实现所述多个控制器之间的负载均衡。

12.根据权利要求9或10所述的数据处理单元,其特征在于,所述数据处理单元还用于:

向所述存储设备发送上报命令,所述上报命令指示所述存储设备将所述主机对应的逻辑单元的信息发送给所述数据处理单元;

接收所述存储设备中的控制器发送的所述主机对应的逻辑单元的信息;

根据所述主机对应的逻辑单元的信息,为所述主机对应的逻辑单元生成对应的设备,将生成的所述设备的信息发送给所述主机。

13.一种计算机系统,其特征在于,包括主机和权利要求9至12中任一项所述的数据处理单元,所述数据处理单元通过PCIe接口连接到所述主机。

网络存储方法、存储系统、数据处理单元及计算机系统

技术领域

[0001] 本申请涉及存储技术领域，尤其涉及一种网络存储方法、存储系统、数据处理单元及计算机系统。

背景技术

[0002] 如图1所示，在传统的网络存储方案中，需要在主机中安装多路径软件，主机通过多路径软件和主机总线适配器(host bus adapter，HBA)卡，将输入/输出(input/output，I/O)请求分发给存储阵列中的不同控制器。

[0003] 如果主机采用开源的多路径软件或者第三方的多路径软件，这些多路径软件与存储阵列中的控制器的配合效果往往不够好，可能会导致控制器间的负载不均衡或者控制器间的大量I/O转发。如果让主机采用存储阵列提供商所定制的多路径软件，由定制多路径软件与存储阵列中的控制器进行配合以均衡分发，就需要为不同的主机操作系统开发相应的定制多路径软件，然而，目前操作系统的类型以及版本众多，会带来软件开发上的巨大困难。

发明内容

[0004] 为了解决主机依赖多路径软件实现I/O分发所存在的上述问题，本申请实施例提供一种网络存储方法、存储系统、数据处理单元及计算机系统，主机无需安装多路径软件，能够实现I/O请求在控制器间的均衡分发，避免I/O请求在控制器间的大量转发。

[0005] 第一方面，本申请提供了一种网络存储方法，用于存储系统，存储系统包括主机、数据处理单元和存储设备，数据处理单元通过PCIe接口连接到主机，存储设备包括多个控制器和多个逻辑单元；该网络存储方法包括：主机将输入/输出请求发送给数据处理单元，该输入/输出请求用于访问多个逻辑单元中的其中一个逻辑单元；数据处理单元从多个控制器中确定用于处理输入/输出请求的控制器，以实现多个控制器之间的负载均衡，向确定的这个控制器发送该输入/输出请求。

[0006] 可以看出，数据处理单元在这里可以作为主机的外部设备(PCIe设备)，通过主机上的PCIe接口连接到主机上。于是，主机可以直接把I/O请求发送给数据处理单元，完全不需要关注I/O请求在存储设备中的多个控制器之间的分发(分配及发送)工作，减轻了主机中的处理器(比如中央处理器(central processing unit，CPU))的负担，节省了CPU资源。数据处理单元收到主机发送的I/O请求后，从多个控制器中确定一个用于处理该I/O请求的控制器，以使得多个控制器之间能够负载均衡，可以尽量避免I/O请求在控制器间的转发，然后数据处理单元把该I/O请求发送给确定的这个控制器。

[0007] 基于第一方面，在可能的实施例中，数据处理单元与主机之间通过NVMe协议进行通信，数据处理单元与存储设备中的控制器之间通过NVMe-oF协议进行通信。

[0008] 可以看出，数据处理单元与主机、存储设备(中的控制器)之间，分别采用的是不同的通信协议，所以，数据处理单元在主机和存储设备之间还要负责完成协议转换的工作。主

机无需关注和存储设备之间的交互，原本由主机中的处理器来完成的网络协议处理、I/O分发等任务，现在都卸载到了数据处理单元来执行，能够减轻主机中处理器的压力。

[0009] 由于主机与数据处理单元之间采用NVMe协议进行通信，所以主机可以把数据处理单元当成本地的NVMe存储设备，与真正的存储设备的交互工作全都由数据处理单元来负责。数据处理单元通过NVMe-oF这样的高性能存储协议与(远端或者云上)的存储设备进行通信，有助于提升主机的网络存储效率，不会对主机操作系统的性能造成影响，还便于拓展主机的外部存储容量。

[0010] 基于第一方面，在可能的实施例中，数据处理单元从多个控制器中确定用于处理输入/输出请求的控制器，包括：数据处理单元根据哈希算法，从多个控制器中确定用于该处理输入/输出请求的控制器，以实现多个控制器之间的负载均衡。

[0011] 也就是说，数据处理单元可以使用哈希算法做控制器间的负载均衡。比如，可以采用一致性哈希算法与控制器进行配合，以逻辑单元中划分的分片作为负载均衡的粒度，根据I/O请求所要访问的分片，将主机下发的每个I/O请求分发给存储设备中的各个控制器，避免I/O请求在控制器间的I/O转发。

[0012] 基于第一方面，在可能的实施例中，所述方法还包括：数据处理单元向存储设备发送上报命令，上报命令指示存储设备将主机对应的逻辑单元的信息发送给数据处理单元；数据处理单元接收存储设备中的控制器发送的主机对应的逻辑单元的信息，根据主机对应的逻辑单元的信息，为主机对应的逻辑单元生成对应的设备，将生成的设备的信息发送给主机。

[0013] 需要说明的是，主机对应的逻辑单元，指的是存储设备为主机分配的那部分逻辑单元。比如，存储设备可以根据主机的存储资源需求量或者其他因素，将存储设备中的一个或者多个逻辑单元分配给主机。

[0014] 可以看出，数据处理单元可以为主机执行扫描逻辑单元的操作，以便发现存储设备为该主机分配的逻辑单元，然后上报给主机。由于存储设备中的控制器与逻辑单元之间没有归属关系，所以存储设备可以分别通过各个控制器，将主机对应的逻辑单元的信息发送给数据处理单元。数据处理单元接收到每个控制器所发送的信息后，需要进行聚合，然后为主机对应的每个逻辑单元分别生成一个对应的设备，最后将生成的设备的信息通过NVMe协议发送给主机。

[0015] 基于第一方面，在可能的实施例中，所述方法还包括：主机根据数据处理单元发送的设备的信息，生成该设备对应的虚拟存储设备，虚拟存储设备用于提供给主机中的应用进行访问。

[0016] 应理解，主机收到数据单元发送的设备的信息后，可以把该设备抽象为对应的虚拟存储设备，不会直接把该设备暴露给主机中的应用。

[0017] 第二方面，本申请提供了一种存储系统，该存储系统包括主机、数据处理单元和存储设备，数据处理单元通过PCIe接口连接到主机，存储设备包括多个控制器和多个逻辑单元；主机用于，将输入/输出请求发送给数据处理单元，该输入/输出请求用于访问多个逻辑单元中的一个逻辑单元；数据处理单元用于，从多个控制器中确定用于处理该输入/输出请求的控制器，以实现多个控制器之间的负载均衡，向确定的该控制器发送该输入/输出请求。

[0018] 基于第二方面,在可能的实施例中,数据处理单元还用于:通过NVMe协议与主机进行通信;通过NVMe-oF协议与存储设备中的控制器进行通信。

[0019] 基于第二方面,在可能的实施例中,从多个控制器中确定用于处理输入/输出请求的控制器,包括:根据哈希算法,从多个控制器中确定用于处理输入/输出请求的控制器,以实现多个控制器之间的负载均衡。

[0020] 基于第二方面,在可能的实施例中,数据处理单元还用于:向存储设备发送上报命令,上报命令指示存储设备将主机对应的逻辑单元的信息发送给数据处理单元;接收存储设备中的控制器发送的主机对应的逻辑单元的信息;根据主机对应的逻辑单元的信息,为主机对应的逻辑单元生成对应的设备,将生成的设备的信息发送给主机。

[0021] 基于第二方面,在可能的实施例中,主机还用于:根据数据处理单元发送的设备的信息,生成该设备对应的虚拟存储设备,该虚拟存储设备用于提供给主机中的应用进行访问。

[0022] 第三方面,本申请提供了一种数据处理单元,数据处理单元用于:接收主机发送的输入/输出请求,其中,该数据处理单元通过PCIe接口连接到主机,该输入/输出请求用于访问存储设备中的一个逻辑单元,存储设备包括多个逻辑单元和多个控制器;从多个控制器中确定用于处理输入/输出请求的控制器,以实现多个控制器之间的负载均衡,向确定的该控制器发送该输入/输出请求。

[0023] 需要说明的是,数据处理单元是一种专用处理器,这里可以理解为是数据处理单元芯片。该芯片作为主机外部的PCIe设备,通过PCIe接口连接到主机上,进而能够为主机提供I/O请求的均衡分发、网络协议处理、扫描逻辑单元等功能,减轻了主机中的处理器的压力。

[0024] 基于第三方面,在可能的实施例中,数据处理单元还用于:通过NVMe协议与主机进行通信;通过NVMe-oF协议与存储设备中的控制器进行通信。

[0025] 基于第三方面,在可能的实施例中,数据处理单元还用于:根据哈希算法,从多个控制器中确定用于处理该输入/输出请求的控制器,以实现多个控制器之间的负载均衡。

[0026] 基于第三方面,在可能的实施例中,数据处理单元还用于:向存储设备发送上报命令,上报命令指示存储设备将主机对应的逻辑单元的信息发送给数据处理单元;接收存储设备中的控制器发送的该主机对应的逻辑单元的信息;根据主机对应的逻辑单元的信息,为主机对应的逻辑单元生成对应的设备,将生成的设备的信息发送给主机。

[0027] 第四方面,本申请提供了一种计算机系统,包括主机和上述第三方面中任一实施例的数据处理单元,数据处理单元通过PCIe接口连接至主机。

[0028] 综上所述,本申请实施例为主机配置数据处理单元(数据处理单元作为主机外部的PCIe设备,通过PCIe接口连接到主机),将I/O请求的均衡分发、网络协议处理、扫描逻辑单元等任务均卸载到了数据处理单元,能够减轻主机中处理器的压力。由于主机不需要安装多路径软件,只需要配备该数据处理单元,能够避免为众多类型的操作系统开发相应的定制多路径软件的困难。而且,目前几乎所有类型的操作系统均支持NVMe协议,使用主机操作系统中的NVMe驱动器即可实现对数据处理单元的驱动。所以,该数据处理单元不会受到主机操作系统类型、版本的限制,适配性/通用性好,可以广泛使用。

[0029] 数据处理单元通过NVMe协议与主机通信,对主机呈现为本地的NVMe设备,所以主

机无需关注与存储设备之间的交互。数据处理单元通过NVMe-oF协议与存储设备中的控制器进行通信,实现高性能的数据存取,实际是把主机的存储空间拓展到位于远端或者云上的存储设备,能够简化主机的设计,尽可能减少主机本地所需的存储容量,节省成本。数据处理单元接收到主机发送来的I/O请求后,从存储设备的多个控制器中确定用于处理该I/O请求的控制器,以实现控制器之间的负载均衡。具体可以采用哈希算法与控制器进行配置,以分片作为负载均衡的粒度在控制器间进行I/O分发,避免I/O请求在控制器之间的转发。

[0030] 数据处理单元还用于实现扫描逻辑单元的功能,向存储设备发送上报命令,以指示存储设备将主机对应的逻辑单元的信息发送给数据处理单元。数据处理单元接收到每个控制器发来的主机对应的逻辑单元的信息后,为主机对应的每一个逻辑单元都生成一个对应的设备,然后将生成的设备的信息发送给主机,以便主机感知到存储设备为其分配的存储空间(即主机对应的逻辑单元),进而可以往分配的存储空间中存取数据。

附图说明

[0031] 为了更清楚地说明本申请实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍。

[0032] 图1是本申请实施例提供的一种主机依赖多路径软件实现I/O请求分发的示意图;

[0033] 图2是本申请实施例提供的以分片为单位将I/O请求打散分发到不同控制器的示意图;

[0034] 图3是本申请实施例提供的一种存储系统的架构图;

[0035] 图4是本申请实施例提供的一种网络存储方法的流程示意图;

[0036] 图5是本申请实施例提供的DPU将主机下发的I/O请求分发给各控制器的示意图;

[0037] 图6是本申请实施例提供的多路径模块将主机下发的I/O请求分配给不同控制器的示意图;

[0038] 图7是本申请实施例提供的一种存储设备向主机上报LUN的流程示意图;

[0039] 图8是本申请实施例提供的向主机上报LUN1和LUN2的示意图;

[0040] 图9是本申请实施例提供的一种计算机系统的结构示意图。

具体实施方式

[0041] 为了便于理解本申请实施例中的技术方案,下面先对本申请实施例中涉及的部分术语及概念进行解释说明。

[0042] 1、数据处理单元(data processing unit,DPU):

[0043] DPU是专用处理器的一个大类,是继中央处理器(central processing unit,CPU)、图形处理器(graphics processing unit,GPU)之后,数据中心场景中的第三颗重要的算力芯片,为高带宽、低延迟、数据密集的计算场景提供计算引擎。DPU具有的CPU通用性和可编程性,但更具有专用性,DPU通过较大程度的并行性与CPU区别开来。

[0044] 2、逻辑单元(logic unit,LU):

[0045] 在存储技术中,各种存储器(比如硬盘)一般不直接暴露给主机,而是将多个存储器抽象化为存储池(或者内存池),然后将存储池的存储空间划分为逻辑单元提供给主机使用。每个逻辑单元都会分配相应的标识,即逻辑单元号(logical unit number,LUN)。需要

说明的是,由于主机一般能直接感知到LUN,本领域技术人员通常直接用LUN代指逻辑单元,若无特殊说明,下文均采用LUN表示逻辑单元。

[0046] LUN可能具有两种不同功能的标识,分别是LUN ID和通用唯一识别码(universally unique identifier,UUID)。LUN ID是LUN在设备内的标识,用于区别设备内的不同LUN。然而,LUN ID只是在同一个设备中是唯一的,有可能会和其他设备中的某个LUN的LUN ID重合。UUID虽然也是LUN的标识,但UUID是全球唯一的编号。需要说明的是,由于LUN挂载到主机上之后,从主机上看到的LUN是主机LUN,主机LUN的标识称为主机LUN ID。由于主机LUN并不是实际存在的物理空间,只是LUN在主机上的映射。因此,主机LUN ID也可以认为是LUN的标识。

[0047] 3、LUN无归属:

[0048] LUN无归属,指的是各个LUN与(存储)控制器之间没有归属关系,每个控制器都可以对任意一个LUN进行访问。

[0049] 举例来说,如图2所示,两个磁盘框中的磁盘共同构成存储池,存储系统中的每个控制器都可以访问这个存储池的存储空间,所以这个存储池是全局性的。而且,控制器间是共享访问缓存的(或者说控制器间的缓存互为备份),即缓存也是全局性的。

[0050] 存储池的存储空间被划分为LUN供主机使用,每个LUN都有对应的标识。对于每个LUN而言,它不归属于任何一个控制器,每个控制器都可以对其进行访问。比如,假设将LUN1切分为12个分片,每个分片占用LUN1中设定大小的逻辑空间。当主机有访问LUN1中的各分片的一些I/O请求时,这些I/O请求通过存储区域网络(storage area network,SAN)下发,并且被打散分发给各个控制器(图中仅是一种I/O打散分发的示例)。其中,控制器A被分配到访问LUN1中的分片1、分片3或分片7的一些I/O请求,然后控制器A根据分配到的I/O请求访问LUN1并执行相应的I/O操作;控制器B被分配到访问LUN1中的分片2,分片4或分片11的一些I/O请求,所以控制器B根据分配到的I/O请求也会去访问LUN1……总之,LUN1不归属于任意一个控制器,4个控制器均可以访问LUN1。

[0051] 下面介绍本申请实施例中涉及的存储系统300。

[0052] 请参见图3,图3是本申请实施例中提供的一种存储系统300的架构图,存储系统300包括一个或多个主机301(图中仅以一个主机301为例)、DPU 302和存储设备303。其中,每个主机301均有对应的DPU 302,DPU 302通过快捷外围部件互连标准(peripheral component interconnect express,PCIe)接口与主机301连接(或者说通过PCIe总线与主机中的处理器连接)。需要说明的是,图中的DPU 302位于主机301的外部,但DPU 302也可以直接集成到主机301的内部或者以插接的形式连接到主机301。

[0053] 下面分别对主机301、DPU 302和存储设备303进行具体介绍。

[0054] 1、主机301可以是服务器、笔记本电脑、台式计算机等设备,本申请不做具体限定。如图3所示,在主机301的操作系统(operating system,OS)中,包括非易失性存储器标准(non-volatile memory express,NVMe)驱动器,NVMe驱动器用于驱动NVMe设备。

[0055] 在可能的实施例中,主机301中可以安装有各种应用程序,用户可以通过应用程序触发I/O请求,以存取数据。

[0056] 2、DPU 302为主机301提供一种或多种类型的网口,以便主机301能连接到相应类型的网络中,进而能够和存储设备303进行交互。所以,主机301无需单独的网络接口卡

(network interface card,NIC)。

[0057] 比如,在存储前端网络为聚合以太网上的远程直接内存访问(remote direct memory access over converged ethernet,RoCE)网络时,DPU 302为主机301提供RoCE网口,以便主机301能连接到RoCE网络,进而能够在存储设备303中存/取数据。

[0058] 需要说明的是,DPU 302除了可以直接通过网络与存储设备303通信,还可以通过交换机访问存储设备303以存取数据。图3中只绘制了一个网口,但不代表只有一个或一种类型的网口。

[0059] 在可能的实施例中,DPU 302可以包括NVMe控制器、多路径模块和NVMe over Fabrics(简称NVMe-oF或者NoF)启动器:

[0060] 1)NVMe控制器(即NVMe controller),用于实现NVMe协议,使DPU 302对主机301呈现为一个NVMe设备。换句话说,NVMe控制器可以将DPU 302模拟成一个NVMe设备,从主机301来看,DPU 302相当于本地的NVMe设备,所以主机301可以使用自身的NVMe驱动器来驱动DPU 302,主机301和DPU 302之间采用NVMe协议进行通信。

[0061] 2)多路径模块,用于对存储设备303通过多条路径(或者说通过多个控制器305)上报到DPU 302的LUN进行聚合,并将聚合后的LUN交由NVMe控制器上报给主机301。多路径模块还用于,将主机301下发的I/O请求打散分配给存储设备303中的各个控制器305,以实现各个控制器305之间的负载均衡。

[0062] 3)NoF启动器(即NoF initiator,也可简称NoF INI),用于实现和NoF目标器(即NoFtarget)之间的NoF连接,实现建立通信链路、扫描LUN、I/O下发等功能。

[0063] 需要说明的是,DPU 302中还可以包括更多或者更少的模块,图中的三个模块NVMe控制器、多路径模块、NoF启动器仅是一种示例性的划分,各模块也可以进一步拆分为多个功能模块,或者将多个模块组合为一个模块,本申请不做限定。

[0064] 3、存储设备303可以是存储阵列、硬盘框、服务器(集群)、台式计算机等,本申请不做具体限定。存储设备303包括存储介质304以及多个控制器305。

[0065] 1)存储介质304,用于提供存储空间。存储介质304的存储空间被虚拟化为存储池,然后划分为多个LUN,LUN与控制器305之间无归属关系,每个控制器305都可以控制访问任意一个LUN。

[0066] 例如,如图3所示,假设存储介质304的存储空间被划分为了n个逻辑单元(n为大于1的正整数),n个逻辑单元分别用LUN1、LUN2至LUNn来表示。存储设备303中的每一个控制器305都可以访问这n个逻辑单元中的任意一个,所以控制器305和LUN之间没有归属关系。

[0067] 在本申请实施例中,存储介质304可以是同一种类型的存储器,比如一个或多个固态硬盘;也可以是多种类型的存储器的组合,比如固态硬盘和机械硬盘的组合;还可以是一个或多个硬盘框,在硬盘框中可以放置一块或多块硬盘,等等。需要说明的是,本申请对存储介质304不做具体限定。

[0068] 2)控制器305,用于对存储介质304进行访问控制,根据接收到的I/O请求执行相应的I/O操作。每个控制器305中都有NoF目标器(即NoF target),NoF目标器是一个软件模块,用于与NoF启动器实现NoF通信,一个NoF target就表示接收NoF命令的一个节点。

[0069] 在可能的实施例中,每个控制器305都有一个对应的接口卡,以便该控制器305能够连接到网络中。比如,在存储前端网络为RoCE网络时,每个控制器305均有一个RoCE接口

卡,以便该控制器305能够连接到RoCE网络。除了RoCE网络,还可以是其他类型的支持NoF协议的网络,本申请不做限定,不需要改变原来的组网结构。需要说明的是,图3中的接口卡处于控制器305的外部,但接口卡也可以直接集成到控制器305的内部或者以插卡的方式连接到控制器305,本申请不作限定。

[0070] 在可能的实施例中,存储设备303中的每个控制器305内部都设有缓存,各个控制器305间可以共享缓存,互为备份。

[0071] 以存储设备303是包含两个控制器305(分别是控制器A和控制器B)的存储阵列为例,控制器A和控制器B间具有镜像通道,当控制器A将一份数据写入其缓存后,可以通过镜像通道将该数据的副本发送给控制器B,控制器B将该副本存储在自身的缓存中。于是,控制器A和控制器B互为备份,当控制器A发生故障的时候,控制器B可以接管控制器A的业务,当控制器B发生故障时,控制器A可以接管控制器B的业务,从而避免硬件故障导致整个存储阵列的不可用。类似的,如果存储阵列中部署有4个控制器305,任意两个控制器305之间都具有镜像通道,因此任意两个控制器305互为备份。

[0072] 下面基于存储系统300,介绍本申请提供的网络存储方法的实施例。

[0073] 请参见图4,图4是本申请实施例提供的一种网络存储方法的流程示意图,该方法用于上述存储系统300,包括以下步骤:

[0074] S401、主机301将I/O请求发送给DPU 302,所述I/O请求用于访问存储设备303中的多个逻辑单元中的一个逻辑单元,DPU通过PCIe接口连接到主机301。

[0075] 应理解,主机301一次可以将一个或者多个I/O请求发送给DPU 302,每个I/O请求可以访问这多个逻辑单元中的任意一个或者某个指定的逻辑单元。步骤S401中是以一个I/O请求为例进行表述,但不代表只能有一个I/O请求。

[0076] 在可能的实施例中,每个I/O请求中都包含地址信息,地址信息指示该I/O请求所针对的地址段,以便在该地址段上存/取数据。地址信息可以直接指示I/O请求所要访问的地址段,也可以间接指示I/O请求所要访问的地址段,本申请不做限定。

[0077] 举例来说,如果主机301发送的是读请求(即O请求),那么读请求中的地址信息指示该读请求所要访问的地址段,以便从该地址段上读取数据;如果主机301发送的是写请求(即I请求),写请求中还包括要写入的数据,那么写请求中的地址信息指示该写请求所要访问的地址段,以便往该地址段上存储数据。

[0078] 由前述内容可知,存储设备303中的存储介质304用于提供存储空间,但存储介质304的存储空间并不直接暴露给主机301,而是虚拟化为存储池,然后可以划分为LUN提供给主机301使用。因此,在一些可能的实施例中,地址信息中包括一种或多种LUN的标识。这里的LUN的标识,可以是LUN的UUID,也可以是存储介质304中的LUN在主机301中的映射的标识,比如主机LUN ID,或者LUN在主机301中对应的块设备(虚拟存储设备)的标识等。

[0079] 可以理解的是,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)来确定。对于起始地址,本领域技术人员通常称为逻辑地址块(logical block address,LBA)。所以,在可能的实施例中,地址信息中包括LUN的标识、LBA和length,LUN的标识、LBA和length这三个因素可以标识一个确定的地址段。所以,主机301发送的I/O请求都会被定位到一个地址段上,以便从所述地址段上读取数据,或者往所述地址段上写入数据。

[0080] 需要说明的是，I/O请求中除了可以携带LUN ID、LBA和length作为地址信息，还可以采用其他逻辑地址构建地址信息，例如虚拟空间ID、虚拟空间的起始地址以及长度。本申请对地址信息的表现方式不做具体限定。

[0081] 在一种可能的实施例中，主机301中可以安装各种应用程序，用户可以通过主机301中运行的应用程序触发主机301生成I/O请求以存取数据，然后主机301会把生成的I/O请求通过NVMe协议发送给DPU 302。

[0082] S402、DPU 302从存储设备303中的多个控制器305中确定用于处理该I/O请求的控制器305，以实现多个控制器305间的负载均衡，向确定的控制器305发送该I/O请求。

[0083] 在一具体实施例中，当DPU 302接收到主机301下发的一个或多个I/O请求时，DPU 302会从存储设备303中的多个控制器305当中，为这一个或多个I/O请求分别确定一个用于处理该I/O请求的控制器305，然后将这一个或多个I/O请求分别发送给确定的控制器305，以实现多个控制器305间的负载均衡。或者说，存储设备303中的每个控制器305均有一条到DPU 302的路径，当DPU 302接收到主机301下发的一个或者多个I/O请求时，DPU 302为这一个或者多个I/O请求分别确定一条到存储设备303的路径，也就是打散到多条路径，相当于间接为每个I/O请求确定一个控制器305，DPU 302再将每个I/O请求分别通过确定的路径发送到该路径相应的控制器305，使得多个控制器305间能够负载均衡。

[0084] 举例来说，如图5所示，存储设备303包含两个控制器305，分别是控制器A和控制器B。控制器A和控制器B均有一条路径到DPU 302，为了便于描述，将控制器A和DPU 302之间的这条路径称为路径1，将控制器B和DPU 302之间的这条路径称为路径2，路径1和路径2均可以归结为存储设备303和DPU 302之间的路径。将存储介质304中的逻辑单元LUN1划分为6个分片，分别用分片1至分片6来表示，每个分片都占用LUN1中设定大小的逻辑空间。

[0085] 假设主机301生成了一些I/O请求，这些I/O请求分别要访问LUN1中的不同分片，然后主机301将这些I/O请求发送给DPU 302。相应的，当DPU 302接收到主机301下发的这些I/O请求后，分别为每个I/O请求确定一个用于处理该I/O请求的控制器305，以实现控制器A和控制器B之间的负载均衡。假设访问LUN1中的分片1、分片3或分片6的这部分I/O请求被确定由控制器A来处理(或者说是确定由路径1发送给存储设备303)，访问LUN1中的分片2、分片4或分片5的这部分I/O请求被确定由控制器B来处理(或者说是确定由路径2发送给存储设备303)。于是，DPU 302通过相应的路径，将这些I/O请求分别发送给了存储设备303中的不同控制器305，其中，访问LUN1中的分片2、分片4或分片5的这部分I/O请求通过路径1发送给控制器A，访问LUN1中的分片2、分片4或分片5的这部分I/O请求则通过路径2发送给控制器B。

[0086] 在可能的实施例中，DPU 302包括NVMe控制器、多路径模块和NoF启动器；NVMe控制器负责接收主机301通过NVMe协议发送的一个或多个I/O请求，然后交给DPU 302中的多路径模块；接着，多路径模块为这一个或多个I/O请求分别确定一个用于处理该I/O请求的控制器305，使得负载在存储设备303中的多个控制器305间均衡分配；最后，NoF启动器根据多路径模块的分配情况，将这一个或多个I/O请求分别发送给确定的控制器305。

[0087] 例如，如图6所示，假设存储设备303包含两个控制器305，分别是控制器A和控制器B，控制器A和控制器B均有一条路径到DPU 302，将控制器A和DPU 302之间的这条路径称为路径1，将控制器B和DPU 302之间的这条路径称为路径2，路径1和路径2均可以称为存储设备303和DPU 302之间的路径。LUN1是存储介质304中划分出来的其中一个逻辑单元，将LUN1

的逻辑空间切分为六个分片，分别用分片1至分片6来表示，每个分片都占用LUN1中设定大小的逻辑空间。可以理解的是，访问LUN1中的任意一个分片的I/O请求，都可以归结为访问LUN1的I/O请求。

[0088] 假设主机301生成了一些I/O请求，这些I/O请求分别要访问的是LUN1中的不同分片，然后主机301通过NVMe协议将这些I/O请求发送给了DPU 302。当DPU 302中的NVMe控制器接收到这些I/O请求后，将这些I/O请求发送给多路径模块。多路径模块为这些I/O请求分别确定一条到存储设备303的路径，以实现控制器A和控制器B之间的负载均衡。其中，访问LUN1中的分片1、分片3或分片6的I/O请求被确定通过路径1发送给存储设备303，相当于把访问分片1、分片3或分片6的I/O请求分配给了控制器A；访问LUN1中的分片2、分片4或分片6的I/O请求被确定通过路径2发送给存储设备303，相当于分配给了控制器B。最后，NoF启动器根据多路径模块的分配情况，将访问分片1、分片3或分片6的I/O请求通过路径1发送给控制器A，将访问分片2、分片4或分片5的I/O请求通过路径2发送给控制器B。控制器A和控制器B会分别根据自身接收到的I/O请求去访问存储介质304中的相应位置，执行相应的I/O操作。

[0089] 在可能的实施例中，多路径模块可以根据哈希(Hash)算法进行负载均衡，从多个控制器305中确定用于处理该I/O请求的控制器305，以实现多个控制器305间的负载均衡，然后向确定的这个控制器305发送该I/O请求。实际采用的哈希算法可以与控制器305间进行配合，以分片为单位在控制器305间进行I/O分发，尽量避免I/O请求在控制器305间的转发。

[0090] 比如，假设每个逻辑单元都划分为6个分片，每个分片均占用该逻辑单元中设定大小的逻辑空间，多路径模块可以根据一致性哈希算法(consistent hashing)，将访问同一逻辑单元中的分片1、分片3或分片5的I/O请求分配给控制器A，将访问同一逻辑单元中的分片2、分片4或分片6的I/O请求分配给控制器B，也就是说，访问同一逻辑单元的不同分片的I/O请求，被均匀分配到了控制器A和控制器B之间，能够实现控制器A和控制器B间的负载均衡。当然，除了哈希算法之外，多路径模块也可以采用其他负载均衡算法，将主机301下发的I/O请求均衡分配给各个控制器305，本申请不做具体限定。多路径模块还可以根据每个控制器的单位时间访问量或者每个控制器的CPU利用率等因素，为I/O请求确定相应的控制器305，以实现控制器305间的负载均衡。

[0091] 在可能的实施例中，LUN与控制器间没有归属关系，但同一逻辑单元中的不同分片与控制器之间可以有对应关系；多路径模块可以根据该I/O请求所要访问的分片以及该分片与控制器305的对应关系，将该分配所对应的控制器305确定为用于处理该I/O请求的控制器305，然后由NoF启动器通过NoF协议将该I/O请求发送给确定的这个控制器305。

[0092] 举例来说，存储设备303中的LUN1与各个控制器305间没有归属关系，任意一个控制器305均可以访问LUN1，但是，LUN1中的各分片与各个控制器305间有对应关系。假设LUN1划分为6个分片，分片1、分片3、分片5对应控制器A，而分片2、分片4和分片6对应控制器B，也就是，凡是访问LUN1中的分片1、分片3或分片5的I/O请求，都由控制器A来负责处理，访问LUN1中的分片2、分片4或分片6的I/O请求，都由控制器B来负责处理，这种对应关系将访问LUN1的I/O请求以分片为粒度分配给不同的控制器305，有助于实现控制器A、B间的负载均衡。所以，当多路径模块接收到访问LUN1的I/O请求时，便可以先确定该I/O请求实际要访问

的是LUN1中的哪个分片(假设是分片1),然后根据分片1与控制器A的对应关系,便可以将控制器A确定为用于处理该I/O请求的控制器305,最后将这个I/O请求发送给控制器A。

[0093] 综上所述,在本申请实施例提供的存储系统300中,为主机301配备了相应DPU 302,DPU 302作为主机301的外部PCIe设备,通过PCIe接口连接到主机301上。作为主机301和存储设备303之间的连接枢纽,DPU 302将主机301的I/O请求的多路径分发、网络协议处理、扫描LUN等任务均卸载到自身,减轻了主机301中CPU的压力,简化了主机301的设计,同时能够节约成本。

[0094] 由于主机301不需要安装多路径软件,所以不需要软件开发人员额外开发多路径软件,避免了为众多操作系统开发相应的定制多路径软件所带来的困难。而且,目前几乎所有类型的操作系统均支持NVMe协议,使用主机操作系统中的NVMe驱动器即可实现对DPU 302的驱动,所以,该DPU 302不会受到主机操作系统类型或版本的限制,适用性好,可以在各种主机301中推广使用。主机301可以直接把DPU 302当成本地的NVMe设备,所以主机301无需关注和存储设备303之间的沟通,只需要把生成的I/O请求直接发送给DPU 302即可,对主机301的性能影响较小。

[0095] DPU 302中的NVMe控制器负责实现NVMe协议,与主机301进行NVMe通信,当NVMe控制器接收到主机301通过NVMe协议发来的I/O请求时,将这些I/O请求发送给DPU 302中的多路径模块。多路径模块负责将I/O请求分配给存储设备303中的控制器305,实现I/O请求在多个控制器305间的均衡分发。具体可以使用哈希算法与控制器305进行配合,以分片为单位在控制器305间进行I/O分发,使得控制器305间能够负载均衡,避免I/O请求在控制器305间的大量转发。DPU 302中的NoF启动器通过NVMe-oF协议与存储设备中的控制器进行通信,能够实现高性能的数据存取,也就是把主机301的存储空间拓展到位于远端或者云上的存储设备303,能够简化主机301的设计,尽可能减少主机301本地所需的存储容量,节省成本。

[0096] 由DPU 302负责I/O请求在多个控制器305间的均衡分发,还有一个好处就是可以简化存储设备303的硬件设计,存储设备303侧不需要增加负载均衡的相应设计(比如设计定制芯片来实现I/O请求在多个控制器305件的均衡分发),各控制器305根据自身接到的I/O请求进行处理即可。

[0097] 在可能的实施例中,在上述步骤S402之后,该网络存储方法还包括:存储设备303中的每个控制器305分别处理自身接收到的I/O请求。

[0098] 具体来说,存储设备303中的各个控制器305分别根据接收到的I/O请求中的地址信息,确定出该I/O请求所要访问的地址段,然后在该地址段上执行相应的I/O操作。

[0099] 比如,在控制器A接收到的一个I/O请求中,携带有LUN8的标识、LBA和length作为地址信息,根据LUN8的标识、LBA和length,可以确定该I/O请求要访问的是LUN8中的以LBA为起始地址,length为长度的地址段,然后在这个地址段上执行相应的I/O操作。如果是写请求,就在往这个地址段中写入相应的数据;如果是读请求,就从这个地址段中读取相应的数据。

[0100] 需要说明的是,本申请对控制器305如何根据I/O请求访问存储介质304不做具体限定。比如,控制器A根据I/O请求确定出地址段后,实际还要根据该地址段确定对应的全局地址(由一个地址段可以索引到一个全局地址,全局地址所指示的空间在存储池中是唯一的),然后根据全局地址访问存储介质304中的相应物理位置(根据全局地址也就可以确定

其对应的物理地址，物理地址指示的是该全局地址所代表的空间实际位于哪个存储器上，以及在该存储器中的偏移量，即物理空间的位置）。

[0101] 应理解，在主机301通过DPU 302向存储设备303下发I/O请求之前，存储设备303需要将主机301对应的LUN上报给主机301。下面结合存储系统300，介绍存储设备303向主机301上报LUN的过程进行介绍。

[0102] 请参见图7，图7是本申请实施例提供的一种存储设备303向主机301上报LUN的流程示意图，用于存储系统300，包括以下步骤：

[0103] S701、DPU 302向存储设备303发送上报命令，上报命令指示存储设备303将主机301对应的LUN发送给DPU 302。

[0104] 需要说明的是，存储设备303中的存储介质304是被虚拟化为存储池，然后划分为LUN提供给主机301使用的，而上述主机301对应的LUN，指的就是存储设备303分配给主机301的那部分LUN。比如，存储设备303可以根据主机301的资源需求量或者其他因素，向主机301提供一个或多个LUN。

[0105] 还需要说明的是，上报某个/某些LUN，指的是把该LUN的相关信息进行上报，LUN的相关信息可以包括LUN的标识、逻辑地址、容量大小等，本申请不做具体限定。

[0106] 在可能的实施例中，DPU302可以分别通过多条路径，将所述上报命令发送给存储设备303。

[0107] 需要说明的是，存储设备303中的每个控制器305都可以有一条路径到DPU 302，存储设备303中的任意一个控制器305与DPU 302之间的路径，都可以归结为该存储设备303和DPU 302之间的路径。由于存储设备303中有多个控制器305，所以存储设备303和DPU302之间会有多条路径。于是，DPU 302分别通过多条路径将上报命令发送给存储设备303，可以是把上报命令分别发送给存储设备303中的多个控制器305。

[0108] 在可能的实施例中，在DPU 302发送给存储设备303的上报命令中，携带有主机301的标识和/或传输所述上报命令的DPU 302的端口信息。

[0109] 在可能的实施例中，DPU 302可以感知到存储设备303中的LUN配置的变化(比如有新分配给主机301的LUN)，主动向存储设备303发送上报命令，也就是说，DPU 302可以替主机301实现扫描LUN的功能，以便主机301能够感知到存储设备303分配给主机301的逻辑单元的变化。

[0110] S702、存储设备303根据上报命令确定主机301对应的LUN，通过存储设备303中的控制器305将主机301对应的LUN的信息发送给DPU 302。

[0111] 在一具体实施例中，存储设备303将主机301对应的LUN的信息，分别通过存储设备303和DPU 302之间的多条路径发送给DPU 302。应理解，由于LUN与控制器305之间无归属关系，所以同一个LUN要分别通过多个控制器305进行上报，也就是通过多条路径上报，以便DPU 302为同一个LUN确定多条访问路径。

[0112] 在可能的实施例中，存储设备303根据上报命令确定主机301对应的LUN后，生成LUN上报信息，然后分别通过多条路径将LUN上报信息发送给DPU 302。其中，LUN上报信息包括主机301对应的LUN的信息，比如LUN的标识、逻辑地址范围、容量大小等。可以理解的是，存储设备303通过存储设备303和DPU 302之间的多条路径，将LUN上报信息分别发送到DPU 302，所以DPU 302会收到来自不同路径的LUN上报信息。或者说，多个控制器305会分别将

LUN上报信息发送给DPU 302,DPU 302会收到来自不同控制器305的LUN上报信息。

[0113] 在可能的实施例中,在LUN上报信息的传输过程中,LUN上报信息所经过的控制器305会将各自的端口的标识(或者控制器305的标识)加入LUN上报信息中,也就是在LUN上报信息中增加相应的路径信息。比如,通过存储设备303中的控制器A发送给DPU 302的LUN上报信息中,携带有控制器A的端口的标识,通过控制器B发送给DPU 302的LUN上报信息中,携带有控制器B的端口标识,以便区分是来自不同路径的LUN上报信息。

[0114] S703、DPU 302根据存储设备303中的控制器发送的LUN的信息,为主机301对应的每个LUN生成对应的设备,将生成的设备的信息上报给主机301。

[0115] 在可能的实施例中,DPU 302中的NoF启动器负责接收各个控制器305发送的LUN上报信息,然后分别为每个控制器305发送的LUN上报信息中指示的每一个LUN,生成一个对应的设备(是虚拟设备)。接着,多路径模块根据LUN UUID,将NoF启动器生成的设备中对应同一个LUN的设备进行聚合,得到聚合后的设备。NVMe控制器将聚合后的设备分别用NVMe协议中定义的命名空间(namespace,NS)来表示,分配相应的命名空间标识,然后上报给主机301。

[0116] 在可能的实施例中,多路径模块将存储设备303上报某个LUN的路径,设置为访问该LUN的路径。

[0117] 比如,假设控制器A与DPU 302之间的路径为路径1,控制器B与DPU 302之间的路径为路径2。LUN1的相关信息分别通过控制器A和控制器B发送给了DPU 302,于是DPU 302中的多路径模块会把路径1和路径2均设为访问LUN1的路径。后续当DPU 302收到访问LUN1的I/O请求时,就可以根据预设的策略(比如某种负载均衡策略)在这两条路径中选择进行选择,进而通过确定出来的那条路径把该I/O请求发送给存储设备303中该路径所对应的控制器305。

[0118] S704、主机301根据DPU 302上报的设备的信息,创建相应的虚拟存储设备。

[0119] 应理解,主机301根据接收到的设备的信息生成相应的虚拟存储设备,相当于把DPU 302上报的设备给抽象化了,不直接暴露给主机301中的应用,应用只能感知到创建的虚拟存储设备,对这些虚拟存储设备进行访问。

[0120] 在可能是实施例中,当主机301中的NVMe启动器接收到DPU 302中的NVMe控制器发送的命名空间的信息后,将其注册为主机301中的块设备(块设备是某种虚拟存储设备,用于提供给主机301中的应用进行操作),包括为该块设备分配对应的名称,建立该块设备与命名空间的映射关系等,还可以在该块设备中记录其他信息,如逻辑地址空间、容量大小等等。

[0121] 下面结合图8,对图7中的上报LUN的流程进行举例说明。

[0122] 如图8所示,假设存储设备303(为了便于看图,未标出)中有两个控制器305,分别是控制器A和控制器B,控制器A和控制器B均有一条路径到DPU 302,两条路径都属于存储设备303与DPU 302之间的路径。存储设备303中划分的LUN1和LUN2这两个逻辑单元,均是主机301对应的LUN,由于LUN和控制器305之间没有归属关系,所以任意一个控制器305都可以对LUN1和LUN2进行访问控制。

[0123] 假设DPU 302将上报命令发送给了存储设备303。存储设备303根据上报命令中的主机301的标识,可以确定出主机301对应的LUN1和LUN2,然后分别通过控制器A和控制器B,

将LUN1和LUN2上报给DPU 302,即通过两条不同路径将LUN1和LUN2的信息发送给DPU 302。

[0124] 如图8所示,DPU 302中的NoF启动器,根据控制器A发送的LUN1的信息,为LUN1生成了一个对应的设备,命名为NoF11,根据控制器B发送的LUN1的信息,又为LUN1生成了另一个对应的设备,命名为NoF12。可以看出,由于LUN1是通过两条不同路径上报的,LUN1在NoF启动器这里被识别为了两个设备,而这两个设备对应的实际都是LUN1。同样的,NoF启动器分别根据控制器A和控制器B发送的LUN2的信息,也会为LUN2生成两个对应的设备,分别为NoF21和NoF22。

[0125] 多路径模块根据LUN UUID,可以识别出NoF11和NoF12对应的其实是同一个LUN(即LUN1),然后将这两个设备聚合为设备Dev1,所以Dev1对应的也是的LUN1。同样的,多路径模块也会将NoF21和NoF22聚合为同一个设备Dev2。由于NVMe控制器是用于实现NVMe协议的,所以NVMe控制器需要采用NVMe协议中定义的命名空间来表示Dev1和Dev2,将它们分别标识为NS1和NS2,NS1和NS2实际上分别对应是存储设备303中的LUN1和LUN2。然后,NVMe控制器把NS1和NS2的信息上报给主机301。

[0126] 主机301中的NVMe驱动器接收DPU 302上报的NS1和NS2的信息后,将NS1和NS2注册到块设备层,生成两个对应的块设备,分别用nvme0n1和nvme0n2表示,以提供给上层应用进行操作。

[0127] 需要说明的是,上述例子中的各种设备的名称均为示例,不构成限定。

[0128] 综上所述,在本申请实施例提供的存储系统300中,主机301不需要安装多路径软件,只要配备有相应DPU 302即可,所以不需要软件开发人员额外开发多路径软件,避免了为众多操作系统开发定制多路径软件所带来的困难。在本申请实施例中,DPU 302为主机301执行扫描LUN的功能,向存储设备303发送上报命令;DPU 302还负责对存储设备303通过多条路径上报的LUN(的信息)进行聚合,然后将聚合后得到的设备的信息上报给主机301。所以,主机301不需要关注与存储设备303之间的通信,将网络协议处理、扫描LUN等功能均卸载到DPU302,减轻了主机301中处理器(比如CPU)的压力,还能够简化主机301的设计,尽量减少主机301本地的存储容量,节约成本。

[0129] 本申请实施例还提供一种数据处理单元,该数据处理单元可以是前述任一实施例中的DPU 302。该数据处理单元用于:接收主机301发送的输入/输出请求,其中,数据处理单元通过PCIe接口连接到主机301,该输入/输出请求用于访问存储设备303中的一个逻辑单元,存储设备303包括多个逻辑单元和多个控制器305;从多个控制器305中确定用于该处理输入/输出请求的控制器305,以实现多个控制器305之间的负载均衡,向确定的该控制器305发送该输入/输出请求。

[0130] 需要说明的是,数据处理单元是一种专用处理器,这里可以理解为是数据处理单元芯片。该芯片可以作为主机外部的PCIe设备,通过PCIe接口连接到主机上。

[0131] 在可能的实施例中,数据处理单元还用于:通过NVMe协议与主机301进行通信;通过NVMe-oF协议与存储设备303中的控制器305进行通信。

[0132] 在可能的实施例中,数据处理单元还用于:根据哈希算法,从多个控制器305中确定用于处理该输入/输出请求的控制器305,以实现多个控制器305之间的负载均衡。

[0133] 在可能的实施例中,数据处理单元还用于:向存储设备303发送上报命令,上报命令指示存储设备303将主机301对应的逻辑单元的信息发送给数据处理单元;接收存储设备

303中的每个控制器305发送的主机301对应的逻辑单元的信息;根据主机301对应的逻辑单元的信息,为主机301对应的逻辑单元生成对应的设备,将生成的设备的信息发送给主机301。

[0134] 图9是本申请实施例提供的一种计算机系统900的结构示意图。计算机系统900包括主机301以及上述任一实施例中的DPU 302,DPU 302通过PCIe接口连接到所述主机301。也就是说,计算机系统900包括主机301部分以及外部设备(DPU 302作为外部设备)。

[0135] 主机301包括处理器901、存储器902以及通信接口903。其中,处理器901、存储器902以及通信接口903可以通过内部总线904相互连接,也可通过无线传输等其他手段实现通信。本申请实施例以通过总线904连接为例,总线904可以是外设部件互连标准(peripheral component interconnect,PCI)、快捷外围部件互连标准(peripheral component interconnect express,PCIe)总线、、扩展工业标准结构(extended industry standard architecture,EISA)总线、统一总线(unified bus,Ubus或UB)、计算机快速链接(compute express link,CXL)或者缓存一致互联协议(cache coherent interconnect for accelerators,CCIX)等。总线904除包括数据总线之外,还可以包括地址总线、电源总线、控制总线和状态信号总线等。但是为了清楚说明起见,在图中仅用一条粗线表示,将各种总线都标为总线904,但并不表示仅有一根总线或一种类型的总线。

[0136] 处理器901可以由至少一个通用处理器构成,例如中央处理器(central processing unit,CPU),或者CPU和硬件芯片的组合。上述硬件芯片可以是专用集成电路(application-specific integrated circuit,ASIC)、可编程逻辑器件(programmable logic device,PLD)或其组合。上述PLD可以是复杂可编程逻辑器件(complex programmable logic device,CPLD)、现场可编程逻辑门阵列(field-programmable gate array,FPGA)、通用阵列逻辑(generic array logic,GAL)或其任意组合。处理器901执行各种类型的数字存储指令,例如存储在存储器902中的软件或者固件程序,它能使计算机系统900提供多种服务。

[0137] 存储器902用于存储程序代码,并由处理器901来控制执行。

[0138] 存储器902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902也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(flash memory)、硬盘(hard disk drive,HDD)或固态硬盘(solid-state drive,SSD);存储器902还可以包括上述种类的组合。存储器902可以存储有程序代码,具体可以用于执行图7中的网络存储方法的任一实施例或者图8中的任一实施例,这里不再进行赘述。

[0139] 通信接口903中包括至少一个PCIe接口以及其他通信接口,可以为有线接口(例如以太网接口),可以为内部接口、有线接口(例如以太网接口)或无线接口(例如蜂窝网络接口或使用无线局域网接口),用于与与其他设备或模块进行通信。其中,DPU 302通过主机301的PCIe接口连接到主机301的处理器901,以执行图7中的网络存储方法中的任一实施例或者图8中的上报LUN的任一实施例。

[0140] 需要说明的,图9仅仅是本申请实施例的一种可能的实现方式,实际应用中,计算机系统900还可以包括更多或更少的部件,这里不作限制。关于本申请实施例中未出示或未描述的内容,可参见前述图7或图8的任一实施例中的相关阐述,这里不再赘述。

[0141] 本申请实施例还提供一种计算机可读存储介质，计算机可读存储介质中存储有指令，当其在处理器上运行时，图7或者图8的任一实施例的方法得以实现。

[0142] 本申请实施例还提供一种计算机程序产品，当计算机程序产品在处理器上运行时，图7或图8的任一实施例的方法得以实现。

[0143] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程，是可以通过计算机程序来指令相关的硬件来完成，所述的程序可存储于一计算机可读取存储介质中，该程序在执行时，可包括如上述各方法的实施例的流程。其中，所述的存储介质可为磁碟、光盘、只读存储记忆体(read-only memory，ROM)或随机存储记忆体(random access memory，RAM)等。

[0144] 以上所揭露的仅为本申请一种较佳实施例而已，当然不能以此来限定本申请之权利范围，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流程，并依本申请权利要求所作的等同变化，仍属于发明所涵盖的范围。

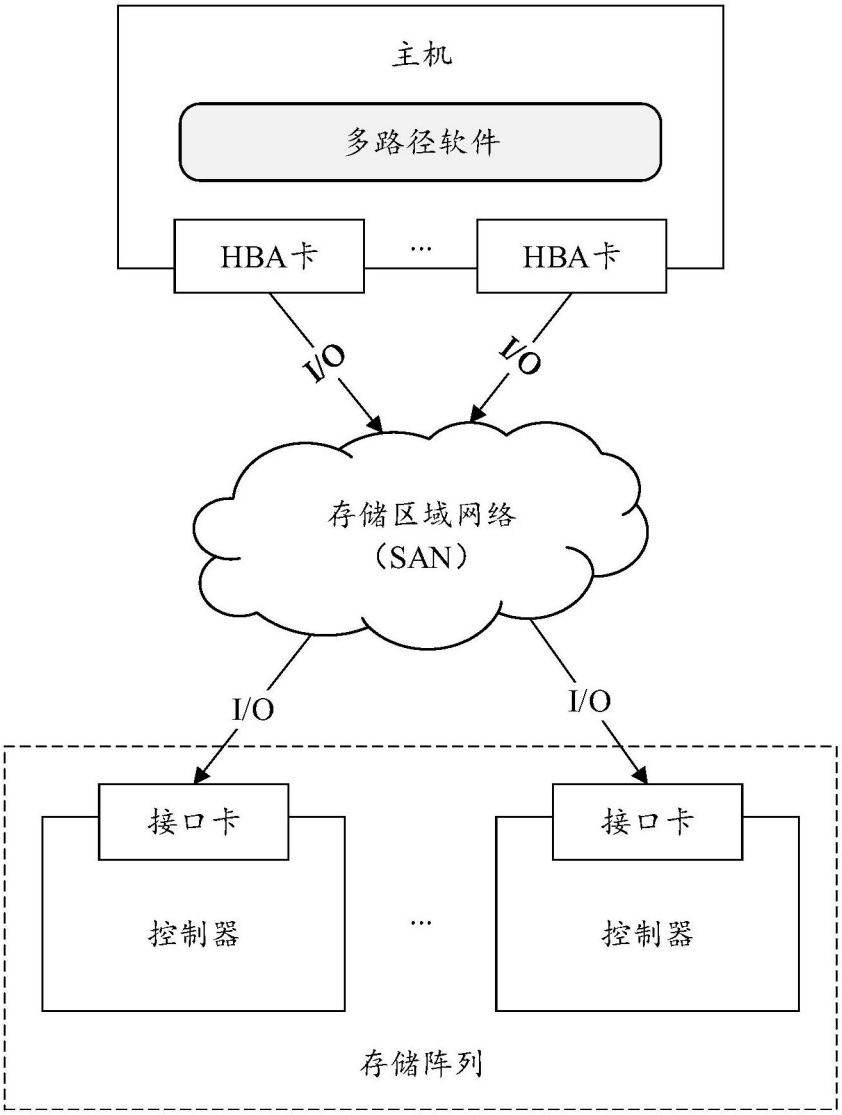


图1

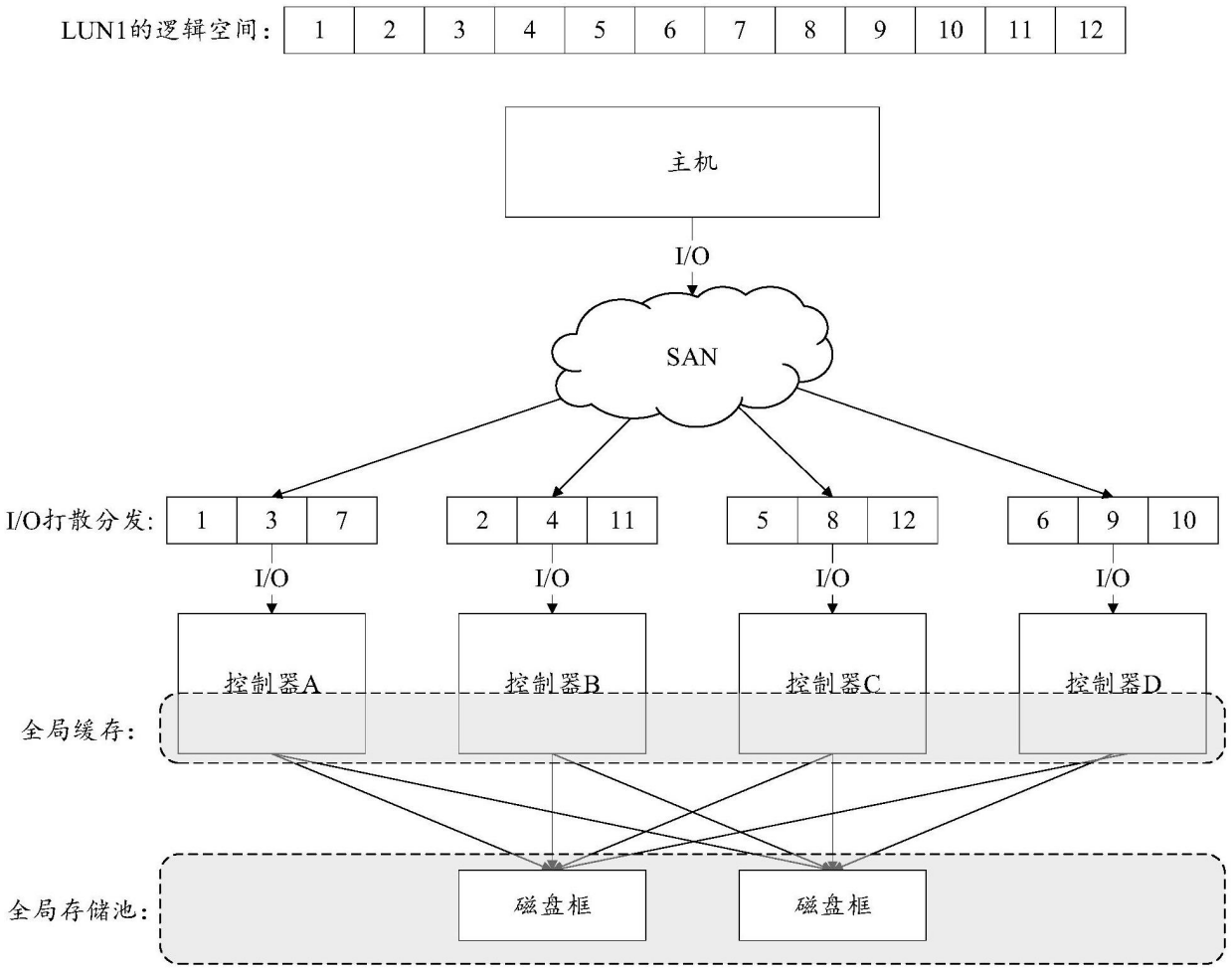

LUN1的逻辑空间：
1
2
3
4
5
6
7
8
9
10
11
12
主机
I/O
SAN
I/O打散分发：
1
3
7
2
4
11
5
8
12
6
9
10
I/O
I/O
I/O
I/O
控制器A
控制器B
控制器C
控制器D
全局缓存：
全局存储池：
磁盘框
磁盘框

图2

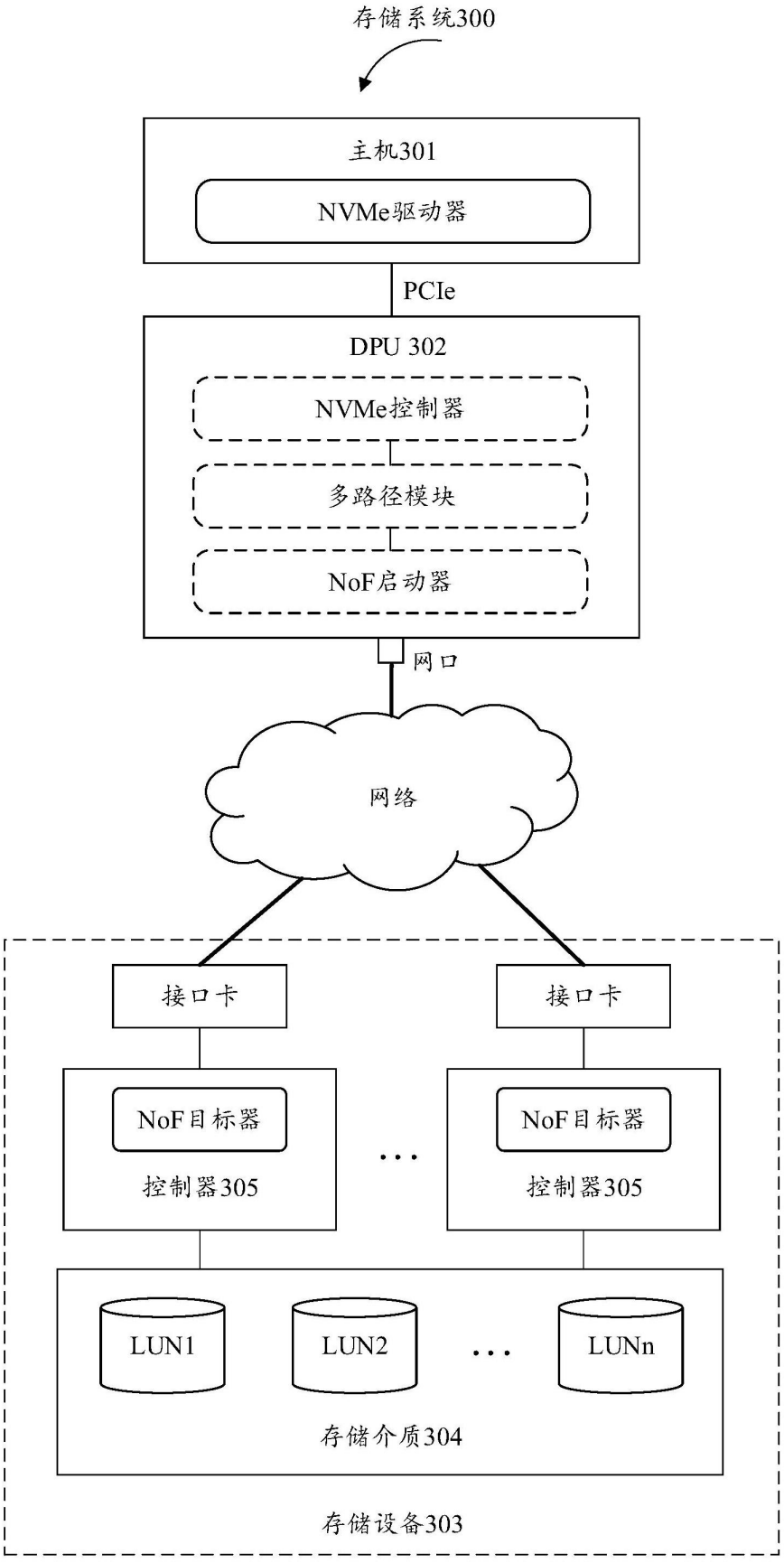

存储系统300
主机301
NVMe驱动器
PCIe
DPU 302
NVMe控制器
多路径模块
NoF启动器
网口
网络
接口卡
接口卡
NoF目标器
控制器305
NoF目标器
控制器305
LUN1
LUN2
LUNn
存储介质304
存储设备303

图3

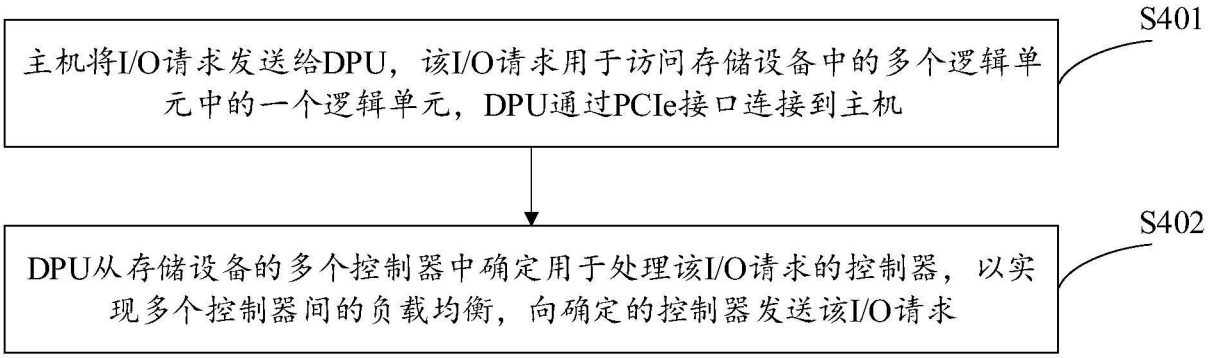

S401
主机将I/O请求发送给DPU，该I/O请求用于访问存储设备中的多个逻辑单元中的一个逻辑单元，DPU通过PCIe接口连接到主机
S402
DPU从存储设备的多个控制器中确定用于处理该I/O请求的控制器，以实现多个控制器间的负载均衡，向确定的控制器发送该I/O请求

图4

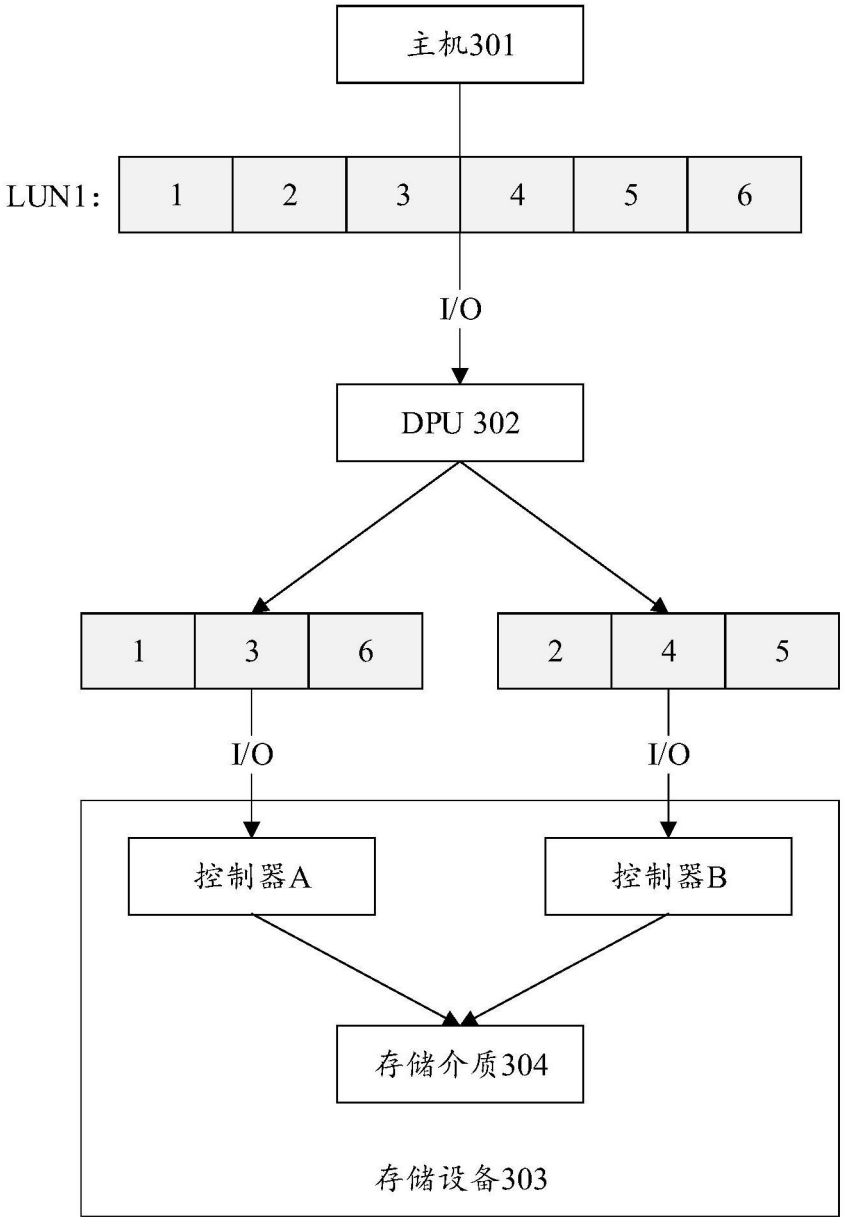

主机301
LUN1:
1
2
3
4
5
6
I/O
DPU 302
1
3
6
2
4
5
I/O
I/O
控制器A
控制器B
存储介质304
存储设备303

图5

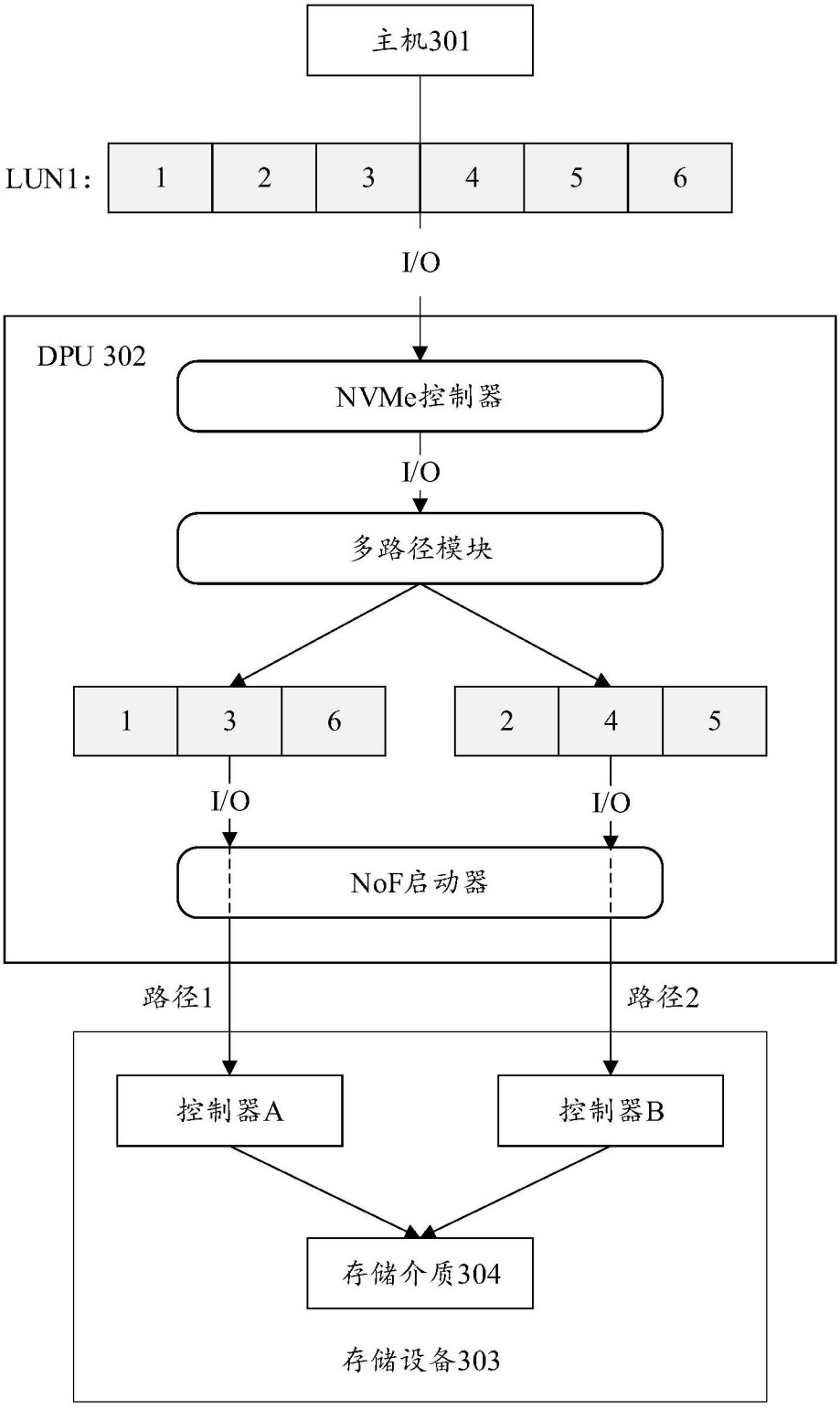

主机301
LUN1:
1
2
3
4
5
6
I/O
DPU 302
NVMe控制器
I/O
多路径模块
1
3
6
2
4
5
I/O
I/O
NoF启动器
路径1
路径2
控制器A
控制器B
存储介质304
存储设备303

图6

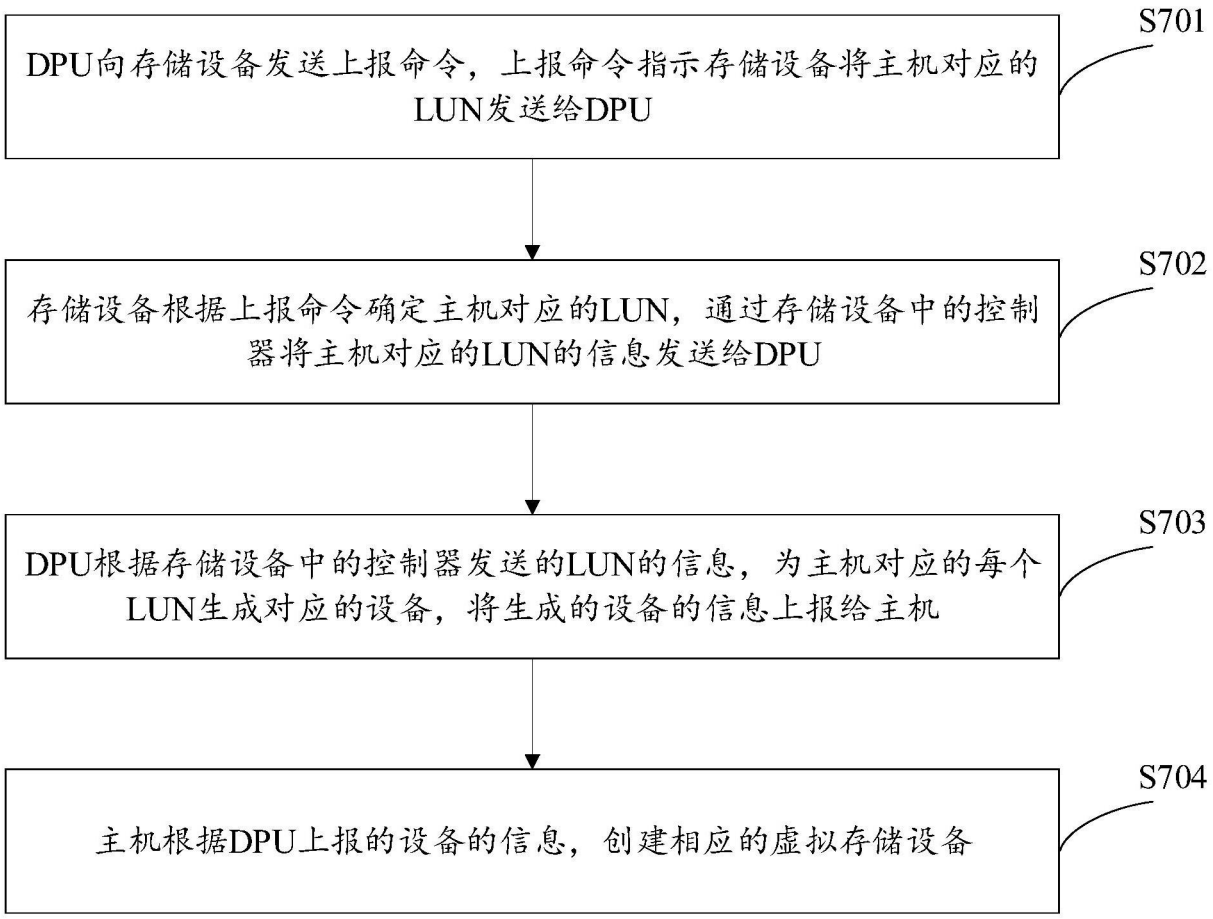

S701
DPU向存储设备发送上报命令，上报命令指示存储设备将主机对应的LUN发送给DPU
S702
存储设备根据上报命令确定主机对应的LUN，通过存储设备中的控制器将主机对应的LUN的信息发送给DPU
S703
DPU根据存储设备中的控制器发送的LUN的信息，为主机对应的每个LUN生成对应的设备，将生成的设备的信息上报给主机
S704
主机根据DPU上报的设备的信息，创建相应的虚拟存储设备

图7

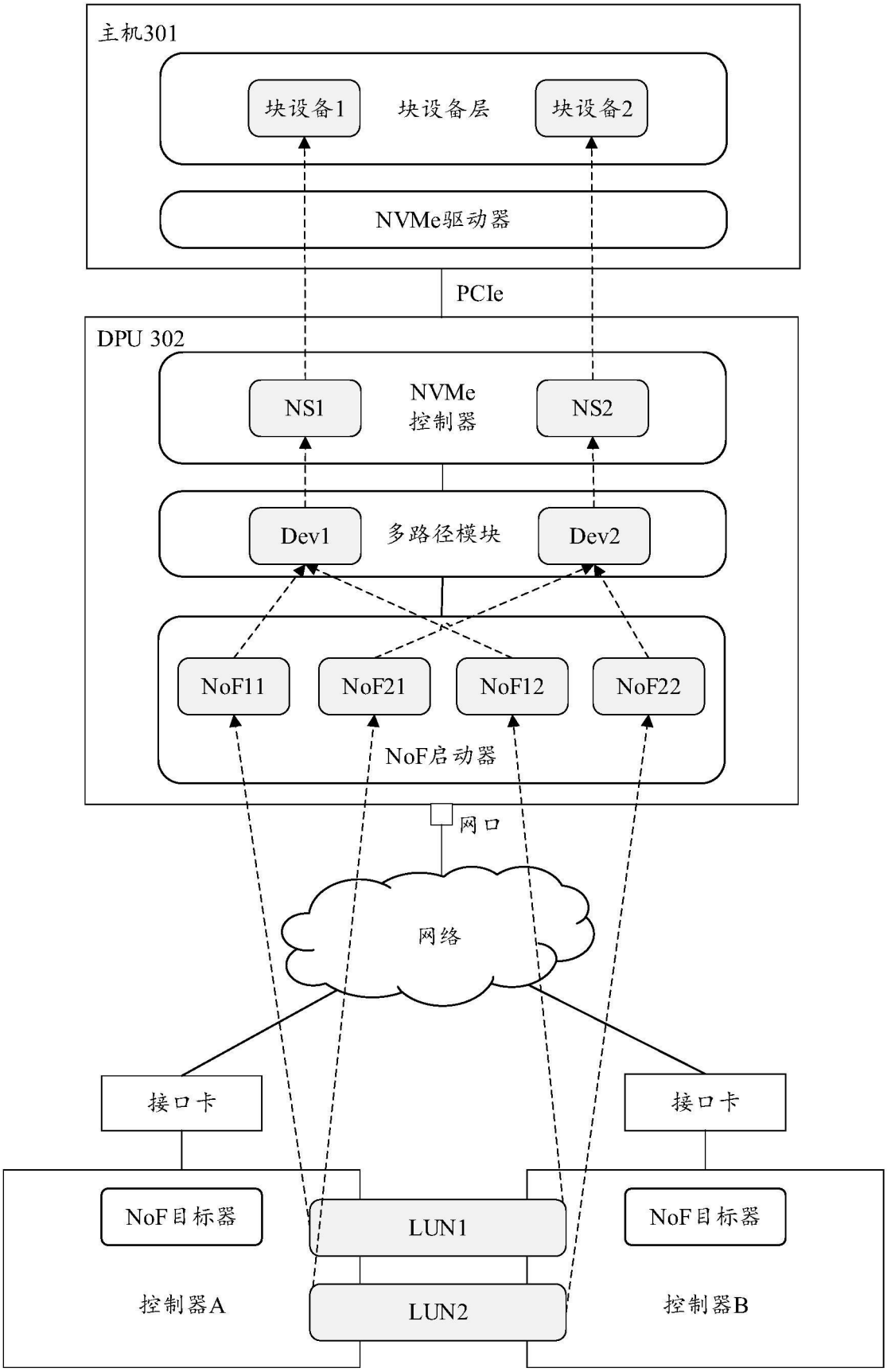

主机301
块设备1
块设备层
块设备2
NVMe驱动器
PCIe
DPU 302
NS1
NVMe
控制器
NS2
Dev1
多路径模块
Dev2
NoF11
NoF21
NoF12
NoF22
NoF启动器
网口
网络
接口卡
接口卡
NoF目标器
LUN1
NoF目标器
控制器A
LUN2
控制器B

图8

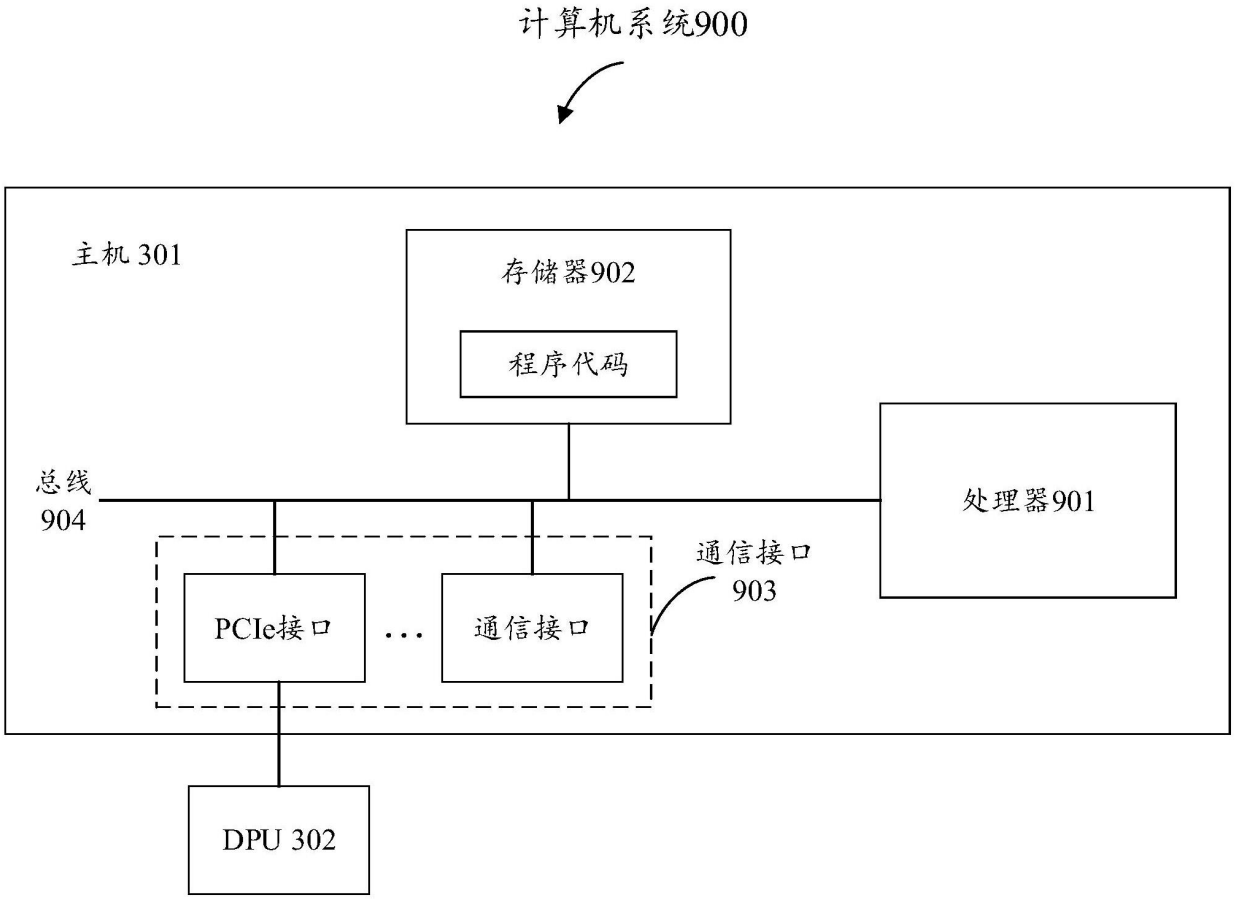

计算机系统900
主机 301
存储器902
程序代码
处理器901
总线
904
通信接口
903
PCIe接口
...
通信接口
DPU 302

图9