(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112241320 B
(45)授权公告日 2023.11.10

(21)申请号 201910955603.X

(22)申请日 2019.10.09

(65)同一申请的已公布的文献号
申请公布号 CN 112241320 A

(43)申请公布日 2021.01.19

(66)本国优先权数据
201910646566.4 2019.07.17 CN

(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 董浩 肖庆航 周臣

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 3/06(2006.01)

(56)对比文件
CN 105677580 A,2016.06.15
US 6578128 B1,2003.06.10
US 6643722 B1,2003.11.04
CN 102033768 A,2011.04.27
CN 102446159 A,2012.05.09
CN 104995603 A,2015.10.21
US 2008046695 A1,2008.02.21

审查员 赵静

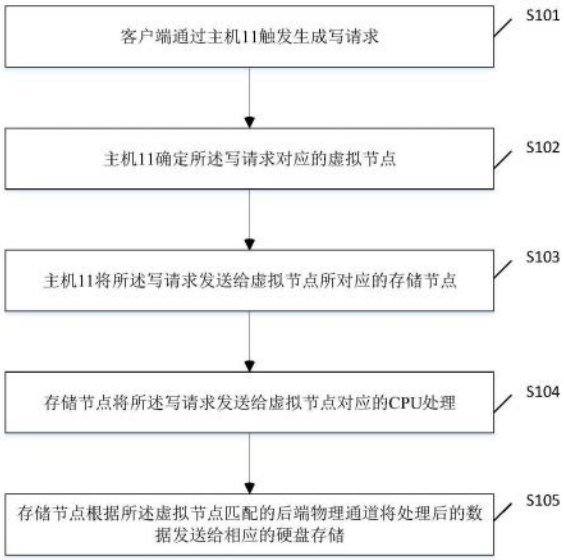
权利要求书3页 说明书15页 附图6页

(54)发明名称

资源分配方法、存储设备和存储系统

(57)摘要

一种资源分配方法和存储设备。存储设备包括硬盘框和多个控制器，每个控制器包括多个处理器，每个处理器包括多个处理器核，所述多个控制器分别和所述硬盘框耦合，所述硬盘框包括多个硬盘。所述多个处理器，用于提供计算资源。所述多个硬盘，用于提供存储空间，所述存储空间所对应的逻辑地址被划分为若干个地址段集合，每个地址段集合包含一个或多个地址段，每个地址段集合分配有所述计算资源中的一部分计算资源，所述一部分计算资源用于执行访问所述地址段集合所包含的地址段的数据访问请求，其中，用于处理不同地址段集合的计算资源来自不同的处理器，或者，用于处理不同地址段集合的计算资源来自不同的处理器核。可以减少处理器之间的带宽资源。



CN 112241320 B

1.一种存储设备,其特征在于,包括硬盘框和多个控制器,每个控制器包括多个处理器和内存,所述内存用于临时存储从硬盘读取的数据,每个处理器包括多个处理器核,所述多个控制器分别和所述硬盘框耦合,所述硬盘框包括多个所述硬盘;

所述多个处理器,用于提供计算资源;

所述多个硬盘,用于提供存储空间,所述存储空间所对应的逻辑地址被划分为若干个地址段集合,每个地址段集合包含一个或多个地址段,每个地址段集合分配有所述计算资源中的一部分计算资源,所述一部分计算资源用于执行访问所述地址段集合所包含的地址段的数据访问请求,其中,为不同地址段集合分配的计算资源来自不同的处理器,或者,为不同地址段集合分配的计算资源来自不同的处理器核。

2.根据权利要求1所述的存储设备,其特征在于,每个处理器具有内存,所述计算资源还包括所述多个处理器包含的内存,其中,一个内存分配给一个地址段集合,且不同地址段集合分配的内存不同。

3.根据权利要求2所述的存储设备,其特征在于,每个地址段集合分配的内存是该地址段集合分配的处理器的本地内存。

4.根据权利要求1所述的存储设备,其特征在于,所述存储设备还包括网络资源,所述网络资源由所述多个控制器与所述硬盘框之间的多个连接提供,每个地址段集合分配有所述网络资源中的一部分网络资源,其中,不同地址段集合分配的连接不同。

5.根据权利要求1所述的存储设备,其特征在于,所述存储空间包括多个块组,所述多个块组提供存储资源,每个地址段集合分配有所述存储资源中的一部分存储资源,其中,不同地址段集合分配的块组不同。

6.根据权利要求1-5任一所述的存储设备,其特征在于,所述存储空间被虚拟化为逻辑单元LUN,所述LUN由所述存储设备提供给用户使用,所述LUN所对应的地址被划分而成若干个逻辑块地址LBA区间,每个地址段为一个LBA区间。

7.根据权利要求1-5任一所述的存储设备,其特征在于,所述存储设备还包括前端接口卡,所述多个控制器分别与所述前端接口卡耦合,所述前端接口卡中保存映射表,所述映射表用于指示每个地址段集合与所分配的计算资源之间的映射关系,和/或每个地址段集合与所分配的网络资源之间的映射关系,和/或每个地址段集合与所分配的硬盘资源之间的映射关系。

8.根据权利要求7所述的存储设备,其特征在于,

所述存储设备中创建有若干个虚拟节点,一个虚拟节点通过预设算法映射至一个地址段集合,所述映射表记录有每个虚拟节点与该虚拟节点所映射的地址段集合所分配的计算资源之间的映射关系,和/或每个虚拟节点与该虚拟节点所映射的地址段集合所分配的网络资源之间的映射关系,和/或每个虚拟节点与该虚拟节点所映射的地址段集合所分配的硬盘资源之间的映射关系。

9.根据权利要求1所述的存储设备,其特征在于,还包括新加入的控制器,所述新加入的控制器包括处理器,所述新加入的控制器与所述硬盘框耦合,

其中,所述新加入的控制器所包含的处理器作为计算资源,被分配给第一地址段集合中的第一地址段,所述第一地址段与所述第一地址段集合所分配的计算资源之间的分配关系被解除。

10.一种资源分配方法,其特征在于,所述方法应用于存储设备中,所述存储设备包括硬盘框和多个控制器,每个控制器包括多个处理器和内存,所述内存用于临时存储从硬盘读取的数据,每个处理器包括多个处理器核,所述多个控制器分别和所述硬盘框耦合,所述硬盘框包括多个所述硬盘;所述方法包括:

提供计算资源和存储空间,所述存储空间所对应的逻辑地址被划分为若干个地址段集合,每个地址段集合包含一个或多个地址段;

为每个地址段集合分配所述计算资源中的一部分计算资源,所述一部分计算资源用于执行访问所述地址段集合所包含的地址段的数据访问请求,其中,为不同地址段集合分配的计算资源来自不同的处理器,或者,为不同地址段集合分配的计算资源来自不同的处理器核。

11.根据权利要求10所述的方法,其特征在于,每个处理器具有内存,所述计算资源还包括所述多个处理器包含的内存,所述方法还包括:

为每个地址段集合分配一个内存,且不同地址段集合分配的内存不同。

12.根据权利要求10所述的方法,其特征在于,每个地址段集合分配的内存是该地址段集合分配的处理器的本地内存。

13.根据权利要求10所述的方法,其特征在于,所述存储设备还包括网络资源,所述网络资源由所述多个控制器与所述硬盘框之间的多个连接提供,所述方法还包括:

为每个地址段集合分配所述网络资源中的一部分网络资源,其中,不同地址段集合分配的连接不同。

14.根据权利要求10所述的方法,其特征在于,所述存储空间包括多个块组,所述多个块组提供存储资源,所述方法还包括:

为每个地址段集合分配所述存储资源中的一部分存储资源,其中,不同地址段集合分配的块组不同。

15.根据权利要求10-14任一所述的方法,其特征在于,所述存储空间被虚拟化为逻辑单元LUN,所述LUN由所述存储设备提供给用户使用,所述LUN所对应的地址被划分而成若干个逻辑块地址LBA区间,每个地址段为一个LBA区间。

16.根据权利要求10-14任一所述的方法,其特征在于,所述存储设备还包括前端接口卡,所述多个控制器分别与所述前端接口卡耦合,所述前端接口卡中保存映射表,所述映射表用于指示每个地址段集合与所分配的计算资源之间的映射关系,和/或每个地址段集合与所分配的网络资源之间的映射关系,和/或每个地址段集合与所分配的硬盘资源之间的映射关系。

17.根据权利要求16所述的方法,其特征在于,所述方法还包括:

接收数据访问请求,所述数据访问请求包括虚拟地址;

根据所述虚拟地址确定所述数据访问请求对应的地址段集合;

在所述映射表中查询为所述数据访问请求对应的地址段集合所分配的计算资源;

利用所述分配的计算资源处理所述数据访问请求。

18.根据权利要求16所述的方法,其特征在于,还包括:

创建若干个虚拟节点,一个虚拟节点通过预设算法映射至一个地址段集合,所述映射表记录有每个虚拟节点与该虚拟节点所映射的地址段集合所分配的计算资源之间的映射

关系，和/或每个虚拟节点与该虚拟节点所映射的地址段集合所分配的网络资源之间的映射关系，和/或每个虚拟节点与该虚拟节点所映射的地址段集合所分配的硬盘资源之间的映射关系。

19.根据权利要求10所述的方法，其特征在于，还包括：

配置新加入的控制器，所述新加入的控制器包括处理器，所述新加入的控制器与所述硬盘框耦合，

将所述新加入的控制器所包含的处理器作为计算资源，分配给一个地址段集合中的第一地址段，并解除所述第一地址段与所述地址段集合所分配的计算资源之间的分配关系。

20.一种存储系统，其特征在于，包括主机和如权利要求1-9任一所述的存储设备。

21.一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存储有指令，当其在存储设备运行时，使得所述存储设备执行如权利要求10-19任一所述的资源分配方法。

资源分配方法、存储设备和存储系统

技术领域

[0001] 本申请涉及存储领域，并且更具体地，涉及一种资源分配方法、存储设备和存储系统。

背景技术

[0002] 当前的存储节点通常具有多个控制器，而每个控制器又包括多个处理器。当存储节点执行一个业务请求时，往往需要多个处理器并行处理由这个业务请求拆分而来的多个子请求，由于这些子请求之间具有关联性，因此会涉及到子请求在多个处理器之间转发和数据交互，占用了处理器之间的带宽资源。

发明内容

[0003] 本申请主要是解决如何节省处理器之间的带宽资源的问题。

[0004] 第一方面提供了一种存储设备，包括硬盘框和多个控制器，每个控制器包括多个处理器，每个处理器包括多个处理器核，所述多个控制器分别和所述硬盘框耦合，所述硬盘框包括多个硬盘。所述多个处理器，用于提供计算资源。所述多个硬盘，用于提供存储空间，所述存储空间所对应的逻辑地址被划分为若干个地址段集合，每个地址段集合包含一个或多个地址段，每个地址段集合分配有所述计算资源中的一部分计算资源，所述一部分计算资源用于执行访问所述地址段集合所包含的地址段的数据访问请求，其中，用于处理不同地址段集合的计算资源来自不同的处理器，或者，用于处理不同地址段集合的计算资源来自不同的处理器核。

[0005] 关于第一方面中的地址段，由于所述存储空间被虚拟化为逻辑单元LUN，所述LUN由所述存储设备提供给用户使用，所述LUN所对应的地址被划分而成的若干个逻辑块地址LBA区间，所以每个地址段为一个LBA区间。

[0006] 第一方面实际上提供了两种具体实现，一种是用于处理不同地址段集合的计算资源来自不同的处理器，由于用于访问某个地址段的数据访问请求都由分配的处理器执行，不同地址段集合分配不同的CPU，可以避免在多个处理器之间转发数据访问请求，节省处理器之间的带宽资源。另一种是用于处理不同地址段集合的计算资源来自不同的处理器核。由于用于访问某个地址段的数据访问请求都由分配的处理器核执行，相对于前一种实现方式，从更细的粒度隔离了不同地址段集合的数据访问请求，数据访问请求在自己分配的处理器核上串行执行，减少了互斥操作，并且可以实现免锁设计。

[0007] 在一种可选的实现方式中，每个处理器具有内存，所述计算资源还包括所述多个处理器包含的多个内存，其中，一个内存分配给一个地址段集合，且不同地址段集合分配的内存不同。

[0008] 在一种可选的实现方式中，每个地址段集合分配的内存是该地址段集合分配的处理器的本地内存。在实际应用场景中，处理器具有自己的内存，也称为处理器的本地内存，本地内存往往和处理器集成在一个组件中，或者与处理器直接或间接耦合。那么在分配内

存时可以优先将该地址段集合所分配的处理器的本地内存分配给该地址段集合，就近使用处理器的本地内存可以避免跨CPU传输数据。

[0009] 在一种可选的实现方式中，所述存储设备还包括网络资源，所述网络资源由所述多个控制器与所述硬盘框之间的多个连接提供，每个地址段集合分配有所述网络资源中的一部分网络资源，其中，不同地址段集合分配的连接不同。

[0010] 在一种可选的实现方式中，所述存储空间包括多个块组，所述多个块组提供存储资源，每个地址段集合分配有所述存储资源中的一部分存储资源，其中，不同地址段集合分配的块组不同。

[0011] 在一种可选的实现方式中，所述存储设备还包括前端接口卡，所述多个控制器分别与所述前端接口卡耦合，所述前端接口卡中保存映射表，所述映射表用于指示每个地址段集合与所分配的计算资源之间的映射关系。

[0012] 可选的，所述映射表还用于指示每个地址段集合与所分配的网络资源之间的映射关系。

[0013] 可选的，所述映射表还用于指示每个地址段集合与所分配的硬盘资源之间的映射关系。

[0014] 在一种可选的实现方式中，所述存储设备中创建有若干个虚拟节点，一个虚拟节点通过预设算法映射至一个地址段集合，所述映射表记录有每个虚拟节点与该虚拟节点所映射的地址段集合所分配的计算资源之间的映射关系。

[0015] 可选的，所述映射表还记录有每个虚拟节点与该虚拟节点所映射的地址段集合所分配的网络资源之间的映射关系。

[0016] 可选的，所述映射表还记录有每个虚拟节点与该虚拟节点所映射的地址段集合所分配的硬盘资源之间的映射关系。

[0017] 在一种可选的实现方式中，所述存储设备还包括新加入的控制器，所述新加入的控制器包括处理器，所述新加入的控制器与所述硬盘框耦合，所述新加入的控制器，用于将所述新加入的控制器所包含的处理器作为计算资源，分配给一个地址段集合中的第一地址段，并解除所述第一地址段与所述地址段集合所分配的计算资源之间的分配关系。当存储系统增加新的资源时，这样的实现可以减少地址段与原有资源之间的映射关系的改变。

[0018] 本申请第二方面提供了一种资源分配方法，该方法应用于第一方面任意一种实现所提供的存储设备中，用于实现所述存储设备的功能。

[0019] 本申请第三方面提供了一种计算机程序产品，当其在存储设备上运行时，使得所述存储设备执行第二方面所提供的资源分配方法。

[0020] 本申请第四方面提供了一种计算机可读存储介质，所述计算机可读存储介质存储有指令，当其在存储设备上运行时，使得所述存储设备执行第二方面所提供的资源分配方法。

[0021] 本申请第五方面提供了一种存储节点，所述存储节点包括多个控制器。其中，所述存储节点中创建有多个虚拟节点，每个虚拟节点预先分配有计算资源，所述计算资源来自计算资源池；所述存储节点接收的若干个数据访问请求被路由到所述多个虚拟节点，其中，每个数据访问请求对应一个虚拟节点，并且每个数据访问请求利用对应的虚拟节点所分配的计算资源执行。所述多个控制器，用于提供计算资源给所述计算资源池，所述计算资源由

所述多个控制器所包含的多个处理器提供，其中，一个处理器分配给一个虚拟节点，且每个虚拟节点分配的处理器不同。或者，一个处理器分配给多个虚拟节点，每个虚拟节点分配的处理器核不同。

[0022] 在第五方面的一种可选的实现方式中，每个处理器具有内存，所述计算资源池还包括所述多个处理器包含的多个内存，其中，一个内存分配给一个虚拟节点，每个虚拟节点分配的内存不同。

[0023] 在第五方面的一种可选的实现方式中，每个虚拟节点分配的内存是该虚拟节点分配的处理器的本地内存。

[0024] 在第五方面的一种可选的实现方式中，所述多个控制器分别与硬盘框耦合，所述硬盘框包括多个硬盘，每个虚拟节点还预先分配有网络资源，所述网络资源来自网络资源池，所述网络资源池由所述多个控制器与所述硬盘框之间的连接提供，每个虚拟节点对应一条或多条连接，每个虚拟节点分配的连接不同。

[0025] 在第五方面的一种可选的实现方式中，每个虚拟节点还预先分配有硬盘资源，所述硬盘资源来自硬盘资源池，所述硬盘资源池由所述多个硬盘提供，每个虚拟节点对应一个或多个块组，每个虚拟节点分配的块组不同。

[0026] 在第五方面的一种可选的实现方式中，所述存储节点还包括前端接口卡，所述前端接口卡与所述多个控制器耦合，所述前端接口卡中保存有每个虚拟节点与分配的计算资源的对应关系，和/或每个虚拟节点与分配的网络资源的对应关系，和/或每个虚拟节点与分配的硬盘资源之间的对应关系。

[0027] 在第五方面的一种可选的实现方式中，所述前端接口卡还用于：接收主机发送的第一数据访问请求，所述数据访问请求包括虚拟地址；根据所述虚拟地址确定所述第一数据访问请求对应的第一虚拟节点；根据所述保存的对应关系确定所述第一虚拟节点分配的第一处理器；将所述第一数据访问请求转发给所述第一处理器；所述第一处理器，用于对所述第一数据访问请求进行处理。

[0028] 在第五方面的一种可选的实现方式中，所述对所述第一数据访问请求进行处理时所涉及的数据或元数据临时保存在所述第一虚拟节点对应的第一内存中。

[0029] 在第五方面的一种可选的实现方式中，所述第一处理器还用于，通过所述第一虚拟节点分配的第一连接，将所述第一数据访问请求要求写入的数据发送给所述硬盘框使得所述数据被写入分配给所述第一虚拟节点的第一块组中，或者通过所述第一连接将所述第一数据访问请求要求读取的数据写入所述第一内存中。

[0030] 本申请第六方面提供另一种资源管理方法，所述方法应用于存储节点中，所述存储节点包括多个控制器，其中，每个控制器包括多个处理器。该方法包括创建多个虚拟节点；为每个虚拟节点分配计算资源，所述存储节点接收的若干个数据访问请求被路由到所述多个虚拟节点，其中，每个数据访问请求对应一个虚拟节点，并且每个数据访问请求利用对应的虚拟节点所分配的计算资源执行；其中，所述多个处理器中的每个处理器分配给所述多个虚拟节点中的一个虚拟节点，每个节点分配的处理器不同。

[0031] 在第六方面一种可选的实现方式中，所述计算资源来自计算资源池，其中，所述计算资源池中的计算资源由所述多个控制器提供，所述计算资源池包括所述多个处理器。

[0032] 在第六方面一种可选的实现方式中，所述创建多个虚拟节点包括：根据所述存储

节点包含的处理器的数量创建所述多个虚拟节点，所述多个虚拟节点的数量小于或等于所述多个存储节点包含的处理器的数量。

[0033] 本申请第七方面提供了一种计算机程序产品，当其在存储节点的控制器上运行时，使得所述控制器执行第六方面所提供的资源分配方法。

[0034] 本申请第八方面提供了一种计算机可读存储介质，所述计算机可读存储介质存储有指令，当其在存储节点的控制器上运行时，使得所述控制器执行第七方面所提供的资源分配方法。

[0035] 本申请第九方面提供了一种存储系统，包括主机和多个存储节点，所述多个存储节点提供的存储空间被划分为若干个地址段集合，每个地址段集合包括一个或多个地址段。所述主机用于生成数据访问请求，所述数据访问请求包括虚拟地址；根据所述虚拟地址确定所述目标数据对应的地址段集合，所述地址段集合指示了处理所述数据访问请求的目标存储节点，所述目标存储节点是所述多个存储节点的其中一个；将所述数据访问请求发送给所述目标存储节点；所述目标存储节点利用预先为所述目标数据对应的地址段集合分配的资源对所述数据访问请求进行处理。其中，所述目标存储节点对所述数据访问请求的处理与第一方面类似。

[0036] 本申请第十方面提供了一种数据处理方法，该方法可以应用在第一方面提供的存储设备中，也可以应用在第五方面提供的存储节点中，也可以应用在存储系统中，该方法包括：接收数据访问请求，该请求中携带虚拟地址。根据虚拟地址可以唯一确定一个地址段集合，或者唯一确定一个虚拟节点，所述地址段集合(或者虚拟节点)分配有计算资源，利用分配的计算资源处理所述数据访问请求。

[0037] 在第十方面的一种可选的实现方式中，所述地址段集合(或者虚拟节点)还分配有网络资源和硬盘资源，所述方法还包括：通过分配的网络资源从所述硬盘资源中读取数据，或者将待写入的数据存储在所述硬盘资源中。

[0038] 按照本申请提供的存储系统，访问不同地址段集合的数据访问请求分配不同的处理器，可以在一定程度上避免在处理器之间转发请求或数据，节省了处理器之间的带宽资源。或者，访问不同地址段集合的数据访问请求分配不同的处理器核，可以进一步地避免在处理器核之间转发请求或数据。另外，访问不同地址段集合的数据访问请求所分配的资源相互独立，可以减少互斥操作，实现免锁设计。

[0039] 本申请第十一方面提供了一种存储节点扩容方法，所述存储节点包括第一控制器，所述第一控制器包括多个处理器，当所述存储节点中增加第二控制器时，创建第二虚拟节点集合，所述第二控制器包括多个处理器，所述第二虚拟节点集合包括多个第二虚拟节点，其中，在创建所述第二虚拟节点集合之前，所述存储节点中有第一虚拟节点集合，所述第一虚拟节点集合包括多个第一虚拟节点，所述存储节点接收的若干个数据访问请求被路由到所述多个第一虚拟节点。将所述第二控制器包括的所述多个处理器添加至计算资源池，所述计算资源池包括所述第一控制器提供的多个处理器。为所述多个第二虚拟节点分配计算资源，所述分配的计算资源来自所述计算资源池，每个第二虚拟节点分配一个处理器，每个第二虚拟节点分配的处理器不同；将路由到所述第一虚拟节点集合的多个数据访问请求中的一部分数据访问请求重新分配给所述第二虚拟节点集合。

[0040] 本申请第十二方面提供了一种存储节点，用于实现第十一方面提供的方法。

[0041] 按照本申请第十一方面提供的存储节点扩容方法和第十二方面提供的存储节点，由于创建了若干个虚拟节点，每个虚拟节点分配有资源，那么当存储节点中增加计算资源增加时，可以相应地增加虚拟节点的数量，将增加的计算资源分配给新增的虚拟节点，并且路由到原有的虚拟节点的一部分数据访问请求可以重新分配给新增的虚拟节点，那么整个存储节点的处理能力也增加了。因此，从而实现了在硬件资源增加时，存储节点的处理能力也随着硬件资源的增加线性增长。

附图说明

[0042] 图1是可应用本发明实施例的技术方案的场景的示意图。

[0043] 图2是本发明实施例提供的存储节点的结构示意图。

[0044] 图3是本发明实施例提供的虚拟节点与处理器资源、内存资源的关系示意图。

[0045] 图4是本发明实施例提供的虚拟节点与网络资源的关系示意图。

[0046] 图5是本发明实施例提供的虚拟节点与硬盘资源的关系示意图。

[0047] 图6是本发明实施例提供的虚拟节点所包含的资源的示意图。

[0048] 图7是本发明实施例提供的处理写请求的流程示意图。

[0049] 图8是本发明实施例提供的CPU核分组调度的示意图。

[0050] 图9是本发明实施例提供的一种扩容示意图。

[0051] 图10是本发明实施例提供的一种扩容方法的流程示意图。

[0052] 图11是本发明实施例提供的另一种扩容示意图。

[0053] 图12是本发明实施例提供的另一种扩容方法的流程示意图。

具体实施方式

[0054] 本申请主要解决如何节省处理器之间的带宽资源的问题。下面将结合附图，对本申请中的技术方案进行描述。

[0055] 图1是可应用本申请实施例的技术方案的场景的示意图。如图1所示，主机11和存储系统通信，存储系统包括多个存储节点(或简称“节点”)100。每个存储节点100是一个存储引擎(简称：引擎)，其中，每个节点100包括多个控制器103。每个控制器103包括多个处理器，每个处理器包括多个处理器核。另外，每个节点100具有前端接口卡101和后端接口卡102。前端接口卡101用于节点100与主机11通信，后端接口卡102用于节点100和多个硬盘框通信，其中每个硬盘框包括多个硬盘107，硬盘用于存储数据，可以是磁盘或者其他类型的存储介质，例如固态硬盘或者叠瓦式磁记录硬盘等。前端接口卡101通过内部网络通道和每个节点所包含的多个控制器103直连，后端接口卡102也通过内部网络通道和每个节点所包含的多个控制器103直连(图1中只示出了部分连接情况)，确保每个节点100内的控制器103可以通过前端接口卡101或后端接口卡102进行业务的收发。另外，每个节点100都可以通过后端接口卡102连接硬盘框，以实现在节点间共享数据。在某些应用场景中，可以将一个或多个存储节点100与硬盘框合称为存储设备。

[0056] 控制器103是一种计算设备，如服务器、台式计算机等。在硬件上，如图2所示，控制器103至少包括处理器104、内存105和总线106。其中，处理器104是一个中央处理器(英文：centra l process ing un it，CPU)，用于处理来自节点100外部的IO请求，或者控制器103

内部生成的请求。在一个控制器103中，可能包含多个处理器104，其中每个处理器包括多个处理器核(图中未示出)。内存105用于临时存储从主机11接收的数据或硬盘读取的数据。控制器103接收主机11发送的多个写请求时，可以将所述多个写请求中的数据暂时保存在内存105中。当内存105的容量达到一定阈值时，将内存105中存储的数据发送给硬盘存储。内存105包括易失性存储器或非易失性存储器或其组合。易失性存储器例如随机访问存储器(英文:random-access memory，RAM)。非易失性存储器例如闪存芯片、软盘、硬盘、固态硬盘(sol id state disk，SSD)、光盘等各种可以存储程序代码的机器可读介质。内存105可具有保电功能，保电功能是指系统发生掉电又重新上电时，内存105中存储的数据也不会丢失。总线106用于控制器103内部各组件之间的通信。

[0057] 控制器103为主机11提供的存储空间来源于多个硬盘107，然而硬盘所提供的存储空间的实际地址并不直接给暴露给控制器103或主机11。在实际应用中，物理存储空间被虚拟化成若干个逻辑单元(logical unit，LU)提供给主机11使用，每个逻辑单元具有唯一的逻辑单元号(logical unit number，LUN)。由于主机11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。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。主机11在生成数据访问请求，通常在该请求中携带LUN ID、LBA和length，为了方便描述，在本实施例中，将LUN ID、LBA和length称为虚拟地址。由上面的描述可知，根据所述虚拟地址可以确定该请求所要访问的LUN以及该LUN的具体位置。控制器103中保存有所述虚拟地址与数据存储在硬盘中的地址之间的对应关系，因此控制器103在接收所述数据访问请求之后，可以根据所述对应关系，确定对应的物理地址，并指示硬盘读取或写入数据。

[0058] 为了保证数据均匀存储在各个存储节点100中，在选择存储节点时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。主机11向存储系统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。应理解，在处理这些数据访问请求时都需要利用存储系统中的CPU资源、内存资源以及其他资源(其中，业界往往将CPU资源和内存资源合并为计算资源)。CPU资源和内存资源是由控制器103提供的，存储节点通常具有多个控制器103，而每个控制器103又包括多个处理器。当存储节点执行一个业务请求时，往往需要多个处理器并行处理由这个业务请求拆分而来的多个子请求，由于这些子请求之间具有关联性，因此会涉及到子请求在多个处理器之间转发和数据交互，占用了处理器之间的带宽资源。

[0059] 为了解决这个问题，本申请实施例为一个地址段集合分配一个CPU，或者分配一个CPU内的一个或多个CPU核，所述地址段集合包含一个或多个地址段，所述地址段之间可以是连续的，也可以是不连续的。只要是用于访问这些地址段的数据访问请求都由所述分配的CPU执行，或者由所述分配的一个或多个CPU核执行。不同地址段集合分配不同的CPU，或者分配同一个CPU中的不同CPU核。

[0060] 进一步地，本申请实施例中，为每个地址段集合分配一个内存，用于访问该地址段

集合所包含的地址段的数据访问请求所涉及的数据(既包括业务数据也包括元数据)都临时保存在所述分配的内存中。具体的,一个内存分配给一个地址段集合,且不同地址段集合分配的内存不同。这里的内存包括但不限于图2中的内存105。在实际应用场景中,处理器具有自己的内存,也称为处理器的本地内存,本地内存往往和处理器集成在一个组件中,或者与处理器直接或间接耦合。那么在分配内存时可以优先将该地址段集合所分配的处理器的本地内存分配给该地址段集合。需要说明的是,上述内存分配方式只是本申请实施例提供的一种实现方式,也可以采用其他实现方式,例如不预先将内存分配给某个地址段集合,任意一个数据访问请求在需要使用内存时再从所述存储系统所包含多个内存中选取。

[0061] 用于处理数据访问请求的资源除了CPU资源、内存资源之外,还可以包括网络资源和硬盘资源。可选的,网络资源和硬盘资源均可以预先分配给不同的地址段集合。

[0062] 当存储系统增加新的资源时,可以将新的资源和原有的资源整合之后,重新分配给各个地址段集合,一种实现方式是重新划分地址段集合,保持地址段的数量不变,增加地址段集合的数量,减少每个地址段集合包含的地址段的数量,然后重新将存储系统所拥有的资源分配给调整后的各个地址段集合,另一种实现方式是,保持每个地址段中部分地址段与原有资源的分配关系,将新增的资源分配该地址段集合中的另一部分地址段。这样的实现可以减少地址段与原有资源之间的映射关系的改变。

[0063] 为了更好地实现针对不同地址段集合的数据访问请求之间的资源隔离,本申请中在存储系统中创建若干个虚拟节点(virtual node),虚拟节点是资源分配的最小单元。存储系统中的资源可以划分为若干个等份,其中每个等份对应一个虚拟节点。具体的,每个虚拟单元均对应有一部分CPU资源、一部分内存资源、一部分网络资源和一部分硬盘资源。例如,存储系统有4个节点100,每个节点内有4个控制器103,每个控制器内有4个CPU,每个CPU有48个CPU核,那么一个节点100共有768个CPU核。一个存储系统如果包含4个节点,则总核数高达3072个。如果每个CPU对应256GB内存,那么一个控制器就有1TB内存,一个节点有4TB内存,存储系统总共有16TB内存。如果将存储系统包含的所有的硬件资源划分为256等份,那么就有256个虚拟节点,每个虚拟节点对应的CPU资源为12个CPU核,且每个虚拟节点对应的内存资源是0.0625TB。如前面的描述,一个分区对应一个地址段,引入虚拟节点之后一个分区集合对应一个虚拟节点,其中,一个分区集合包含多个分区。那么,相应地,一个地址段集合对应一个虚拟节点,其中,一个地址段集合包括多个地址段。换而言之,将一个地址段作为输入,经过预定的算法计算,可以唯一确定到一个分区,并且进一步地唯一确定到一个虚拟节点。假设存储系统中有1024个分区,并且创建了32个虚拟节点,那么,每个虚拟节点对应32个分区集合,其中,每个分区集合包含32个分区。通常情况下,存储系统所包含的分区的数量是不变的。即使存储系统增加或减少了虚拟节点,也只是将1024个分区在增加或减少后的虚拟节点中重新分配而已。

[0064] 应理解,创建虚拟节点并非实现资源隔离的唯一方式,如果没有虚拟节点,也可以按照前面的描述直接将资源分配给各个地址段集合。

[0065] 关于存储系统中虚拟节点的创建,本实施例提供了至少两种创建方式。

[0066] 一种是存储系统在初始化过程中自动创建,具体的过程如下:

[0067] 可以根据系统所包括的(1)存储节点的数量(2)控制器的数量(3)CPU的数量中的任意一种及其组合创建虚拟节点,创建的虚拟节点的数量小于或等于系统所包含的CPU的

数量。然后，为各个虚拟节点分配资源并创建各个虚拟节点与分配的资源之间的映射关系(该部分内容可参考以下图3到图6的描述)，将创建的映射关系保存在主机11和前端接口卡101中。

[0068] 另一种方式是，存储系统在初始化时，存储系统的管理软件为管理员提供界面，管理员通过所述界面选择创建的虚拟节点的数量，存储系统再根据指示创建若干个虚拟节点，为各个虚拟节点分配资源并创建各个虚拟节点与分配的资源之间的映射关系(该部分内容可参考以下图3到图6的描述)，将创建的映射关系保存在主机11和前端接口卡101中。同样的，管理员在选择虚拟节点的数量时可以根据(1)存储节点的数量(2)控制器的数量(3)CPU的数量中的任意一种及其组合，或者其他因素来创建。

[0069] 在以上任意一种创建方式中，都可以存储系统的运行过程中调整虚拟节点的数量，例如在存储系统增加控制器时可以增加虚拟节点的数量，或者在存储系统减少控制器时可以减少虚拟节点的数量，或者在存储系统增加硬盘框时可以增加虚拟节点的数量，或者在存储系统减少硬盘框时可以减少虚拟节点的数量等等。即使没有资源数量的变化，存储系统也可以根据管理员的指定调整虚拟节点的数量。

[0070] 请参考图3，图3是本申请实施例提供的虚拟节点与CPU资源、内存资源的分配关系示意图。如图3所示，存储系统中所有的CPU和所有的内存组成一个计算资源池，将所述计算资源池中包含的CPU资源和内存资源划分为若干个计算资源组，每个计算资源组分配给一个虚拟节点。不同的虚拟节点占用不同的计算资源组。可以是每个计算资源组使用一个CPU，也可以多个计算资源组共用一个CPU。例如，计算资源组0使用CPU_0，计算资源组1使用CPU_1，计算资源组m和计算资源组m+1共用CPU_m，计算资源组n使用CPU_n，m和n都是大于1的整数，且n大于m，可以理解的是计算资源组n和计算资源组m之间还具有一个或多个计算资源组。对于多个计算资源组共用一个CPU的情况，由于一个CPU包含若干个CPU核(例如48个CPU核)，可以将所述一个CPU所包含的若干个CPU核划分为多个核组，每个核组(包括一个或多个CPU核)分配给一个虚拟节点。另外，每个计算资源组还包括内存资源，每个计算资源组所包含的内存资源可以是该计算资源组所包含的CPU本地的内存。这样配置以后，某个虚拟节点对应的数据访问请求运行在分配好的CPU上，那么该虚拟节点可以就近使用该CPU本地的内存资源。所谓CPU本地的内存是指与该CPU位于同一个节点内的内存。具体的，举例来说，计算资源组0使用的内存资源是Mem_0，Mem_0是CPU_0本地的内存；计算资源组1使用的内存资源是Mem_1，Mem_1是CPU_1本地的内存；计算资源组m和计算资源组m+1共用Mem_m，Mem_m是CPU_m本地的内存；计算资源组n使用的内存资源是Mem_n，Mem_n是CPU_n本地的内存。在本实施例中，每个虚拟节点分配一个计算资源组，因此对应不同虚拟节点的业务请求可以使用不同的CPU资源和内存资源，避免了资源竞争。另外，如果按照传统的方式，没有把资源分配给每个虚拟节点，那么执行一个数据访问请求的时候，往往需要多个CPU并行处理由这个请求拆分而来的多个子请求，由于这些子请求之间具有关联性，CPU在处理子请求时会涉及到调度或转发子请求。然而，按照本实施例提供的方式，每个虚拟节点对应一个CPU，或者多个虚拟节点共用一个CPU，那么分配到某个虚拟节点的业务请求都会由一个指定的CPU执行，减少了CPU之间的调度和转发。另外，如果不同的数据访问请求共用一个内存，那么，不可避免地会进行一些互斥操作以实现数据一致性。但按照本实施例的方式，访问不同地址段集合的数据访问请求分配不同的内存，会在一定程度上减少互斥操作。

[0071] 需要说明的是，上述计算资源池只是本实施例提供的一种实现方式，本实施例还可以提供其他实现方式，例如存储系统中部分或所有CPU组成CPU资源，每个虚拟节点分配所述CPU资源中的一部分CPU资源，又例如存储系统中部分或所有内存组成内存资源，每个虚拟节点分配所述内存资源中的一部分内存资源。

[0072] 本申请实施例中的网络资源主要包括控制器103和硬盘框之间的链路资源。各个后端接口卡102上均可以创建多条逻辑链路，每条逻辑链路上可以建立多条连接，这些连接组成网络资源池。请参考图4，图4是本申请提供的虚拟节点与网络资源的分配关系示意图。如图4所示，将所述网络资源池所包含的连接划分为若干链接组，每个链接组使用一条或多条连接，例如，链接组0使用连接_0，连接_0是在逻辑链路_0上建立的；链接组1使用连接_1，连接_1是在逻辑链路_1上建立的；链接组P使用连接_m至连接_n之间的所有连接，m和n都是大于1的整数，且n大于m，连接_m与连接_n之间有一个或多个连接，这些连接是在逻辑链路_n上建立的。每个链接组分配给一个虚拟节点。由于不同的虚拟节点使用不同的连接，由此避免了系统内节点间交换的网络资源竞争。

[0073] 需要说明的是，上述网络资源池只是本实施例提供的一种实现方式，本实施例还可以提供其他实现方式，例如控制器103与硬盘框之间的部分或所有连接组成网络资源，每个虚拟节点分配所述网络资源中的一部分网络资源。

[0074] 本实施例中的硬盘资源主要指存储系统所包含的所有硬盘的容量。请参考图5，图5是本申请实施例提供的虚拟节点与硬盘资源的分配关系示意图。如图5所示，按照设定粒度将硬盘划分为若干个块(chunk)，这些chunk组成了存储池，按照独立硬盘冗余阵列(Redundant Array of Independent Disks，RAID)规则，从不同盘上选择一定数量的chunk，从而组成块组(chunk group，CKG)。例如，块组0包含块_0、块_m和块_n，块_0来自于硬盘_0，块_m来自于硬盘_1，块_n来自于硬盘_n。每个虚拟节点对应一个或多个块组。由于不同虚拟节点使用不同的CKG，从而实现了后端硬盘资源的隔离。

[0075] 需要说明的是，上述存储池只是本实施例提供的一种实现方式，本实施例还可以提供其他实现方式，例如存储系统所包含的部分或全部硬盘组成硬盘资源，每个虚拟节点分配所述硬盘资源中的一部分硬盘资源。

[0076] 综上，每个虚拟节点包含了处理业务所需的CPU资源、内存资源、网络资源以及硬盘资源，如图6所示，虚拟节点0分配的计算资源是计算资源组0，分配的硬盘资源是块组1和块组2，分配的网络资源是链接组m和链接组n。各个虚拟节点之间分配的资源是相互独立的，随着CPU的数量、CPU核的数量线性增加，只要相应地增加虚拟节点的数量，让单个虚拟节点的性能保持一致，就能实现性能随着物理资源的增加而线性扩展。业界将本实施例介绍的这种技术称为CoreFarm。

[0077] 下面介绍存储数据的过程。请参考图7，图7是本实施例提供的处理写请求的流程示意图。如图7所示，包括以下步骤。

[0078] S101、客户端通过主机11触发生成写请求，所述写请求中携带待写入的数据以及所述数据的虚拟地址。虚拟地址是指LUN ID、LBA和length。

[0079] S102、主机11确定所述写请求对应的虚拟节点。

[0080] 具体的，主机11对所述虚拟地址进行哈希计算，从而获得哈希值。所述哈希值与某个分区对应，然后将所述分区的标识按一定规则映射到多个虚拟节点中的某个虚拟节点

(称为目标虚拟节点),规则包括但不限于顺序、随机等算法。为了方便描述,本实施例以目标虚拟节点为图5中的虚拟节点0为例,根据图3至图5的描述,虚拟节点0所分配的硬件资源包括计算资源组0、块组1、块组2、链接组m和链接组n。

[0081] S103、主机11将所述写请求发送给虚拟节点所对应的存储节点。具体的,主机11中保存有关于各个虚拟节点资源分配情况的映射表,该映射表中记录了每个虚拟节点与它所分配的各项资源之间的对应关系(如表1所示)。

[0082]

虚拟节点	计算资源组	块组	链接组
虚拟节点0	计算资源组0	块组1和块组2	链接组m和链接组n
虚拟节点1	计算资源组1	块组0	链接组1
……	……	……	……
虚拟节点p	计算资源组p	块组p	链接组p

[0083] 表1

[0084] 主机11根据表1中记录的虚拟节点0对应的计算资源,确定出该计算资源所在的存储节点。由图5的描述可知,虚拟节点0对应的CPU位于计算资源组0中,进一步地,由图2可知,计算资源组0包括CPU_0和Mem_0,因此,主机11将所述写请求发送给CPU_0所在的存储节点(例如,存储节点0)。如果主机11与存储节点0之间不具有链路,可以通过主机11与其他存储节点之间的链路将所述写请求发送给所述其他存储节点,再由所述其他存储节点转发给存储节点0。如果主机11与存储节点0之间具有一条链路,则直接通过该链路将所述写请求发送给存储节点0。如果主机11与存储节点0之间具有多条链路,则可以通过轮询或者其他方式从多条链路中选择一条链路,通过该选择出的链路将所述写请求发送给存储节点0。

[0085] S104、存储节点接收所述写请求之后,将所述写请求发送给虚拟节点对应的CPU处理。具体的,存储节点的前端接口卡101中保存有所述关于各个虚拟节点资源分配情况的映射表(如表1所示),前端接口卡101可以根据写请求中携带的虚拟地址,确定对应的目标虚拟节点,从而进一步确定目标虚拟节点对应的CPU。继续以目标虚拟节点为虚拟节点0为例,所述虚拟节点0对应的CPU是CPU_0,因此前端接口卡101将所述写请求发送给CPU_0。所述CPU可以对所述写请求中的数据进行相应的处理。处理前和处理后的数据需要暂时存储在内存中,由图2可知,计算资源组0包含的内存资源是Mem_0,因此,数据可以利用Mem_0所指示的内存空间中进行存储。

[0086] S105、存储节点根据所述虚拟节点匹配的后端物理通道将处理后的数据发送给相应的硬盘存储。具体的,当内存Mem_0中存储的数据达到一定水位线时,需要将内存Mem_0中存储的数据写入硬盘进行持久化存储。存储节点可以根据所述映射表查询所述目标虚拟节点对应的块组,将所述待写入的数据写入目标虚拟节点对应的块组中。例如,由图5和表1可知,虚拟节点0对应的块组为块组1和块组2,说明虚拟节点0可以使用块组1的硬盘资源也可以使用块组2的硬盘资源。由图5可知,块组0包含块_0、块_m和块_n,其中,块_0位于硬盘_0中,块_m位于硬盘_1中,块_n位于硬盘_n中。存储节点0可以将所述待写入的数据划分为两个数据分片,并计算获得所述两个数据分片的校验分片,然后通过后端接口卡102将这两个数据分片和一个校验分片分别发送给硬盘_0、硬盘_1和硬盘_n。另外,存储节点0在发送所述数据分片和校验分片需利用网络资源池中的网络资源。例如,虚拟节点0对应的网络资源包括链接组m和链接组n,因此,后端接口卡102可以通过链接组m和/或链接组n包含的多条

连接，将所述数据分片和校验分片并行发送给相应的硬盘。

[0087] 按照图7提供的写请求的处理方法，主机11首先确定该写请求对应的虚拟节点，然后利用预先为该虚拟节点分配的硬件资源处理该请求，由于每个虚拟节点分配的资源是相互独立的，所以当主机11并行处理多个数据处理请求时，多个数据处理请求之间也不会相互干扰。

[0088] 图7是以处理写请求为例予以说明的，除了处理数据读写之外，存储系统还处理其他业务请求，例如数据交换、协议解析和数据刷盘等。在多CPU、多CPU核的存储系统中，这些业务请求往往是由多个CPU或者一个CPU内的多个CPU核串行执行的，而影响线性度的关键就是存储系统中的串行执行以及跨CPU、跨CPU核处理带来的开销。在本实施例中，提供了一种CPU分组调度方法用于解决该问题。请参考图8，图8是虚拟节点内CPU核的分组调度示意图。

[0089] 首先，虚拟节点与一个CPU对应，对应的含义是：一个虚拟节点使用一个CPU，或者，多个虚拟节点共用一个CPU。从而保证了针对同一个虚拟节点的业务请求由同一个CPU处理，因此业务的调度在虚拟节点间保持独立。

[0090] 其次，将虚拟节点对应的一个CPU所包含的多个CPU核按照业务逻辑分成若干业务处理组，每个业务处理组包括一个或多个CPU核。如图7所示，第一业务处理组专用于I/O读写，第二业务处理组专用于数据交换，第三业务处理组用于协议解析，第四业务处理组用于数据刷盘。其中，第三业务处理组和第四业务处理组所包含的CPU核可以共用。具体的，以一个CPU包含48个CPU核为例，假设第一业务处理组分配12个CPU核，第二业务处理组分配12个CPU核，第三业务处理组和第四业务处理组共同分配24个CPU核。

[0091] 按照这样的方式隔离不同的业务请求，在单个业务处理组内，业务请求在自己分配的CPU核上串行执行，可以在一定程度上避免该业务请求与其他业务请求争夺资源，由此减少互斥操作并实现免锁设计，在CPU所包含的CPU核的数量增加时，该CPU的处理能力也能得到线性扩展。另外，将业务请求分组处理之后，业务代码相对于分组之前有所减少，那么业务代码所占用的内存空间也会减小，在总的内存空间不变的情况下，内存可以腾挪出更多的空间用于存储业务数据，由此提高数据的内存命中率。

[0092] 与处理写请求类似，当客户端通过主机11触发读请求时，主机11可以通过所述读请求中携带的待读取数据的虚拟地址确定该请求所对应的虚拟节点，并进一步确定所述虚拟节点对应的存储节点(与S103类似)。主机11将所述读请求发送给虚拟节点对应的存储节点。存储节点接收所述读请求之后，将所述读请求发送给虚拟节点对应的CPU处理(与S104类似)。如果待读取的数据没有在对应的内存中命中，所述虚拟节点对应的CPU可以进一步确定所述虚拟节点对应的网络资源和硬盘资源，然后利用所述对应的网络资源，向对应的硬盘发送请求以读取所述待读取的数据。

[0093] 另外，在实际应用中，通过提升单核能力来提升存储系统能力的代价越来越大，业界目前采用多个节点，每个节点多CPU多核方式来提升存储系统的处理能力。例如在单核能力相差不多的情况下，存储系统中核的数量从48核扩展到768个核，其硬件能力是增长的。但如何让存储系统处理业务的能力能随CPU核数和内存等资源增加而线性扩展也是所有存储设备厂家均需要解决的问题。本实施例提供的扩容方法可以使得存储系统处理业务的能力随着硬件资源增加而线性扩展。下面介绍节点扩容的过程，该过程将结合图3至图6来举

例描述。

[0094] 请参考图9和图10，图9是本申请实施例提供的一种存储系统扩容示意图，图10是本申请实施例提供的一种扩容方法的流程示意图。本实施例以在一个节点内扩展控制器的数量为例予以说明，假设该节点在扩容前包含两个控制器，分别是控制器A和控制器B，增加两个控制器(控制器C和控制器D)后，该节点包含四个控制器。无论是扩容前还是扩容后，其前端接口卡101和后端接口卡102都是各控制器共享的，具体的，如图10所示该扩容方法包含如下步骤。

[0095] S201，控制器C和控制器D加入系统后，控制器C和控制器D分别初始化虚拟节点实例。可以理解的是，控制器的数量增加了，整个节点可提供的CPU资源和内存资源也相应地增加，因此，只要增加虚拟节点的数量，将新增的CPU资源和内存资源分配给新增的虚拟节点就能提高整个节点的处理能力。

[0096] 以控制器C为例，控制器C根据自己所包含的CPU的数量创建多个虚拟节点，由于本实施例中一个虚拟节点分配有一个CPU，因此，虚拟节点的数量可以小于或等于控制器C所包含的CPU的数量。例如，控制器C包含8个CPU，那么控制器C最多可以创建8个虚拟节点。确定虚拟节点的数量之后，再进一步确定新增的虚拟节点与CPU之间的映射关系，以及新增的虚拟节点与内存之间的映射关系。例如，在控制器C内部，虚拟节点x对应CPU_x(x代表一个正整数)，虚拟节点x需要的内存资源可以就近使用CPU_x本地的内存(例如Mem_x)，由此CPU_x和Mem_x组成一个计算资源组分配给虚拟节点x。虚拟节点x+1对应CPU_x+1，虚拟节点x+1需要的内存资源可以就近使用CPU_x+1本地的内存(例如Mem_x+1)，由此CPU_x+1和Mem_x+1组成另一个计算资源组分配给虚拟节点x+1。控制器D创建虚拟节点的方式与控制器C类似。

[0097] 另外，控制器C和控制器D加入系统之后，控制器C和控制器D会分别和后端接口卡102建立物理链路，在这些物理链路上创建多条逻辑链路，每条逻辑链路上又可以建立多条连接，这些连接被添加到图4所示的网络资源池中，以扩充所述网络资源池的网络资源。这些新增的网络资源可以被划分为若干链接组，每个链接组包含一条或多条连接，然后将每个链接组分配给一个新增的虚拟节点，例如虚拟节点x对应链接组x(x代表一个正整数)，虚拟节点x+1对应链接组x+1。

[0098] S202，将归属于控制器A和控制器B的虚拟节点的部分分区迁移至控制器C和控制器D的虚拟节点上。由前面的描述可知，来自主机11的业务请求是根据虚拟地址所对应的分区路由到虚拟节点的，在存储系统所包含的分区的总数不变的情况下，为了让新创建的虚拟节点承担业务请求，就需要将归属于原有的虚拟节点的一部分分区迁移至新创建的虚拟节点上。例如，扩容前一个虚拟节点对应一个分区集合，一个分区集合包含32个分区，扩容后，一个虚拟节点对应24个分区。一种实现方式是重新建立存储系统中所有分区与各个虚拟节点(既包含原有的虚拟节点也包含新增的虚拟节点)的映射关系，另一种实现方式是将原有的分区集合的部分分区迁移至新增的虚拟节点，保留剩下的分区与原有的虚拟节点之间的对应关系。结合上面的例子，那么就需要将原来的分区集合中的8个分区迁移至新增的虚拟节点。需要说明的是，迁移的分区的数量取决于新增的虚拟节点的数量在整个节点所包含的虚拟节点的数量中所占的比例，本实施例不对迁移算法进行限定，只要保证分区在各个虚拟节点中均匀分布即可。

[0099] S203，更新映射表，既包含主机11中保存的映射表也包含前端接口卡101中的映射表。按照S201的描述，新增的虚拟节点分配有CPU资源、内存资源和网络资源。这些新增的分配关系需要记录在映射表中供处理业务请求时使用。由于控制器C和控制器D中没有硬盘，所以新增的虚拟节点所需的硬盘资源仍然来自于图5所示的存储池，具体的，可以将归属于原有的虚拟节点的块组迁移至新增的虚拟节点，本实施例并不对迁移算法进行限定，只要满足每个虚拟节点所分配的块组大致相等即可。

[0100] 更新后的映射表如表2所示。

[0101]

虚拟节点	计算资源组	块组	链接组
虚拟节点0	计算资源组0	块组1和块组2	链接组m和链接组n
虚拟节点1	计算资源组1	块组0	链接组1
……	……	……	……
虚拟节点p	计算资源组p	块组p	链接组p
……	……	……	……
虚拟节点x	计算资源组x	块组p+1	链接组x
虚拟节点x+1	计算资源组x+1	块组p+2	链接组x+1
……	……	……	……

[0102] 表1S204，主机11按照新的分区路由关系发送业务请求。业务请求的处理方式请参考图6所示的处理写请求的流程示意图，这里不再赘述。

[0103] 请参考图11和图12，图11是本申请实施例提供的另一种存储系统扩容示意图，图12是本申请实施例提供的另一种扩容方法的流程示意图。在图11和图12所示的示例中，不仅对节点内的控制器进行了扩充还对硬盘的数量或者硬盘框的数量进行了扩充。仍然以在一个节点内扩展控制器的数量为例予以说明，假设该节点在扩容前包含两个控制器，分别是控制器A和控制器B，增加两个控制器(控制器C和控制器D)后，该节点包含四个控制器。无论是扩容前还是扩容后，其前端接口卡101和后端接口卡102都是各控制器共享的。具体的，如图12所示该扩容方法包含如下步骤。

[0104] S301，控制器C和控制器D加入系统后，控制器C和控制器D分别初始化虚拟节点实例。该步骤可参见图9所示的S201。

[0105] S302，将归属于控制器A和控制器B的虚拟节点的部分分区迁移至控制器C和控制器D的虚拟节点上。该步骤可参见图9所示的S202。

[0106] S303，根据选举算法从新增的控制器C和控制器D选举出一个主控制器，例如控制器C。

[0107] S304，控制器C将新加入的硬盘的空间划分为若干个chunk，并将这些chunk加入存储池。当控制器C或控制器D接收写请求时，所述写请求对应新增的虚拟节点，来自不同硬盘的chunk组成一个块组以容纳所述写请求中携带的数据。由此可见，存储池中新增的多个块组可以分配给新增的虚拟节点，其中，每个虚拟节点使用一个或多块组。

[0108] S305，更新映射表，既包含主机中保存的映射表也包含前端接口卡101中的映射表。该步骤可参见图9所示的S203。与S203不同之处在于，在S203的示例中，没有增加硬盘资源，所以新增的虚拟节点所对应的块组是由系统中原有的虚拟节点的块组迁移而来的，而在S305中的示例中，由于硬盘资源也增加了，所以新增的虚拟节点所对应的块组来自新增

的硬盘资源。

[0109] 按照图11和图12所示的扩容方式，新增的硬盘资源分配给新增的虚拟节点，那么对应所述新增的虚拟节点的写请求中携带的数据可写入所述新增的硬盘中。然而仍然有大量的旧数据仍然保存在原有的硬盘中。为了让系统中存储的数据均匀地分布在各个硬盘上，一种方式是，将部分旧数据迁移至新增的硬盘中存储。另一种方式是，不主动迁移旧数据，而是在对旧数据进行垃圾回收时将所述旧数据中的有效数据(没有被修改的数据)迁移至新增的硬盘中。随着系统运行，垃圾数据越来越多，经过若干次垃圾回收操作之后，慢慢也能达到数据均匀分布的目的。这种的方式的好处在于，由于没有主动迁移数据，可以节省节点之间，或者控制器之间的带宽开销。

[0110] 按照图9-图12所示的两种扩容方法，当系统中的控制器的数量增加时，虚拟节点的数量也相应地增加，并且将新增的资源分配给新增的虚拟节点，在一定程度上减少了对原有的虚拟节点所分配的资源的抢夺，因此，随着硬件资源的数量增加，整个系统的处理能力也相应地增加。

[0111] 在上述实施例中，可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。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程序指令时，全部或部分地产生按照本申请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、存储节点或数据中心通过有线(例如同轴电缆、光纤、数字用户线(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、存储节点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存储节点、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，DVD)、或者半导体介质(例如固态硬盘(Solid State Disk,SSD))等。

[0112] 应理解，在本申请实施例中，术语“第一”等仅仅是为了指代对象，并不表示相应对象的次序。

[0113] 本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

[0114] 所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，上述描述的系统、装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

[0115] 在本申请所提供的几个实施例中，应该理解到，所揭露的系统、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系统，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。

[0116] 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或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。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。

[0117] 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。

[0118] 所述功能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,可以存储在一个计算机可读取存储介质中。基于这样的理解,本申请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部分可以以软件产品的形式体现出来,该计算机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台计算机设备(可以是个人计算机,存储节点,或者网络设备等)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、只读存储器(Read-Only Memory,ROM)、随机存取存储器(Random Access Memory,RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0119] 以上所述,仅为本申请的具体实施方式,但本申请的保护范围并不局限于此,任何熟悉本技术领域的技术人员在本申请揭露的技术范围内,可轻易想到变化或替换,都应涵盖在本申请的保护范围之内。因此,本申请的保护范围应以所述权利要求的保护范围为准。

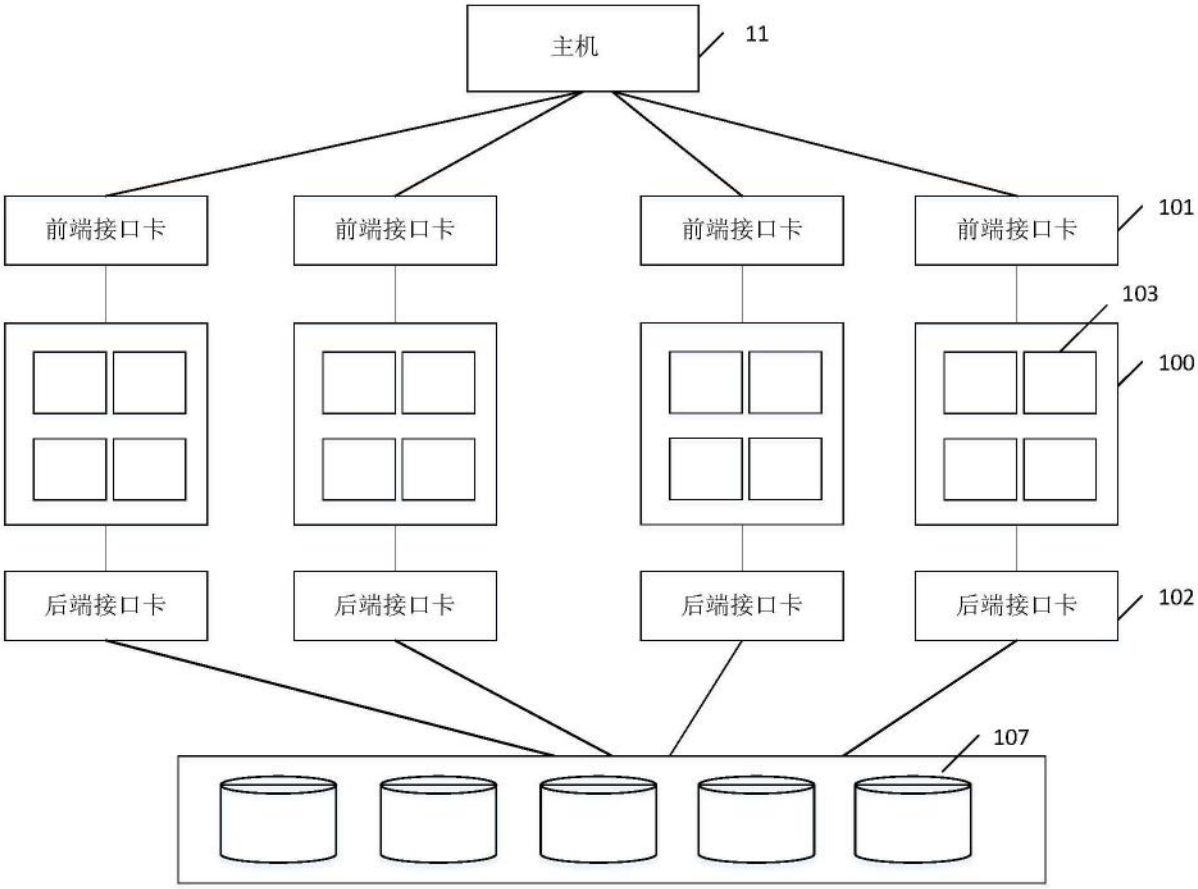

主机
11
前端接口卡
前端接口卡
前端接口卡
前端接口卡
101
103
100
后端接口卡
后端接口卡
后端接口卡
后端接口卡
102
107

图1

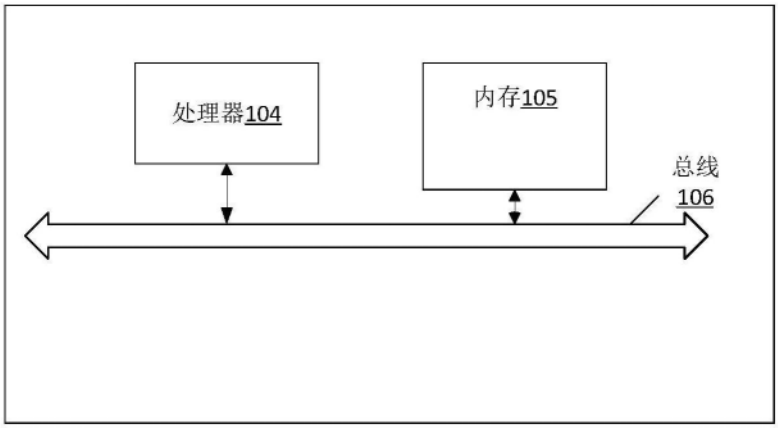

处理器104
内存105
总线
106

图2

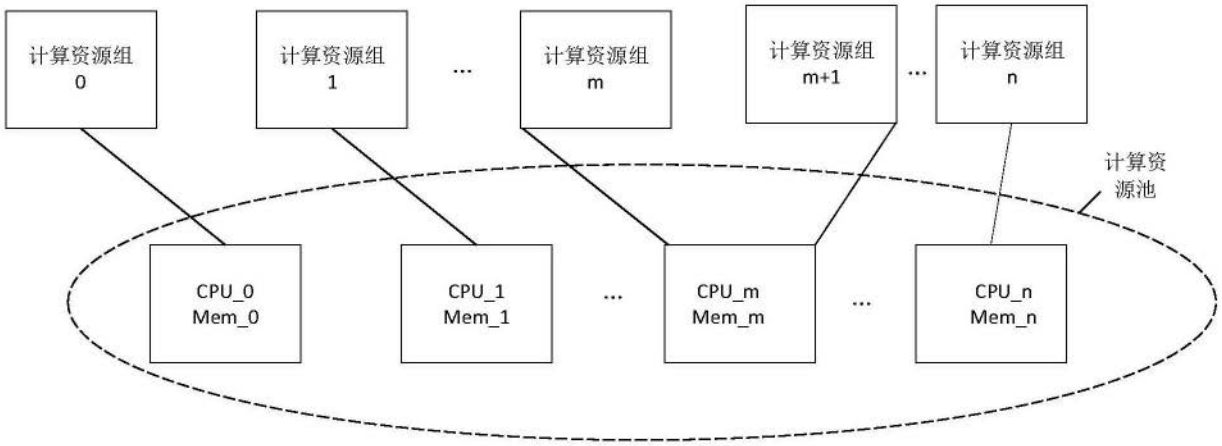

计算资源组
0
计算资源组
1
...
计算资源组
m
计算资源组
m+1
...
计算资源组
n
计算资
源池
CPU_0
Mem_0
CPU_1
Mem_1
...
CPU_m
Mem_m
...
CPU_n
Mem_n

图3

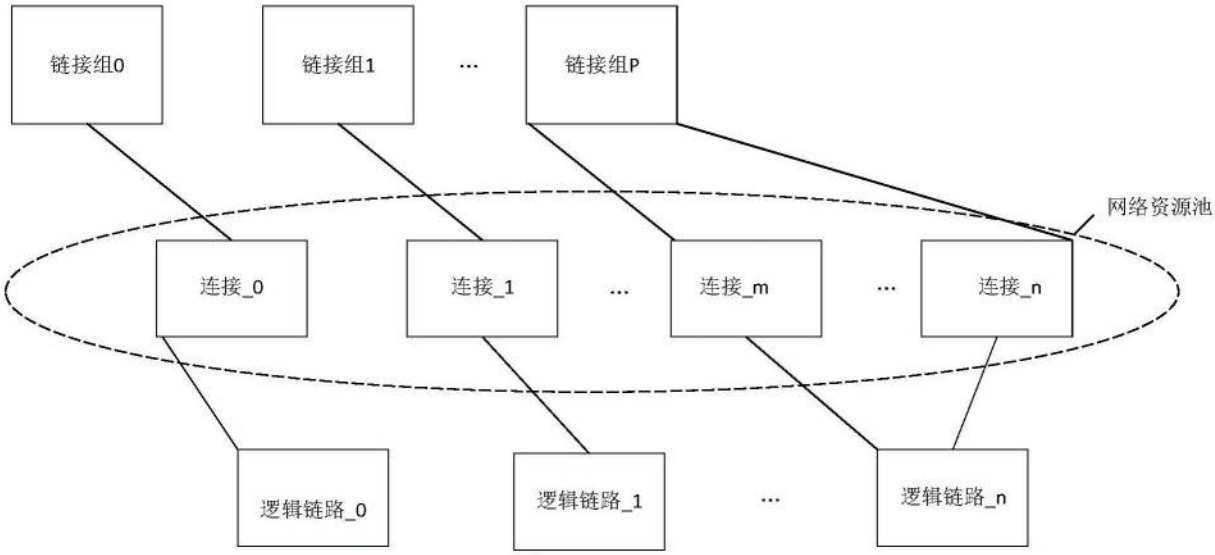

链接组0
链接组1
...
链接组P
网络资源池
连接_0
连接_1
...
连接_m
...
连接_n
逻辑链路_0
逻辑链路_1
...
逻辑链路_n

图4

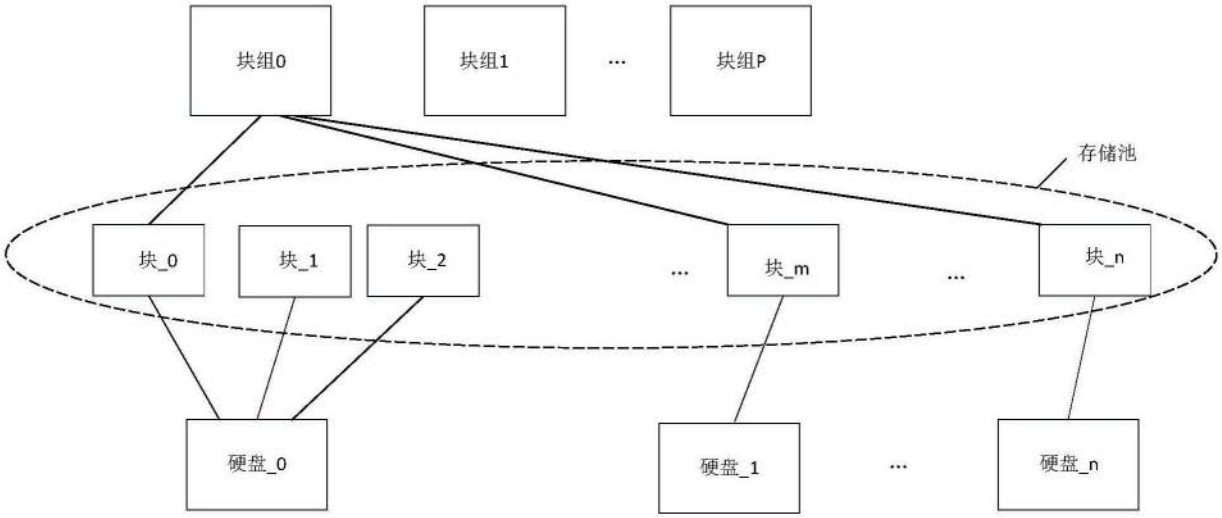

块组0
块组1
...
块组P
存储池
块_0
块_1
块_2
...
块_m
...
块_n
硬盘_0
硬盘_1
...
硬盘_n

图5

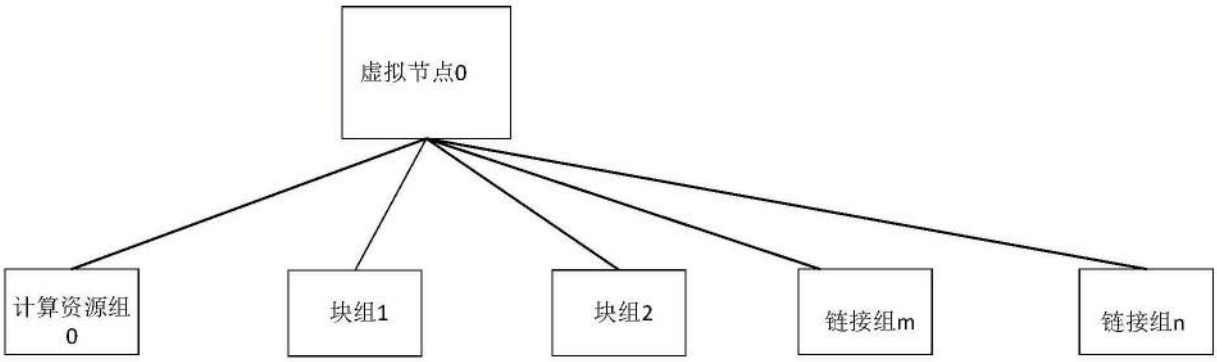

虚拟节点0
计算资源组0
块组1
块组2
链接组m
链接组n

图6

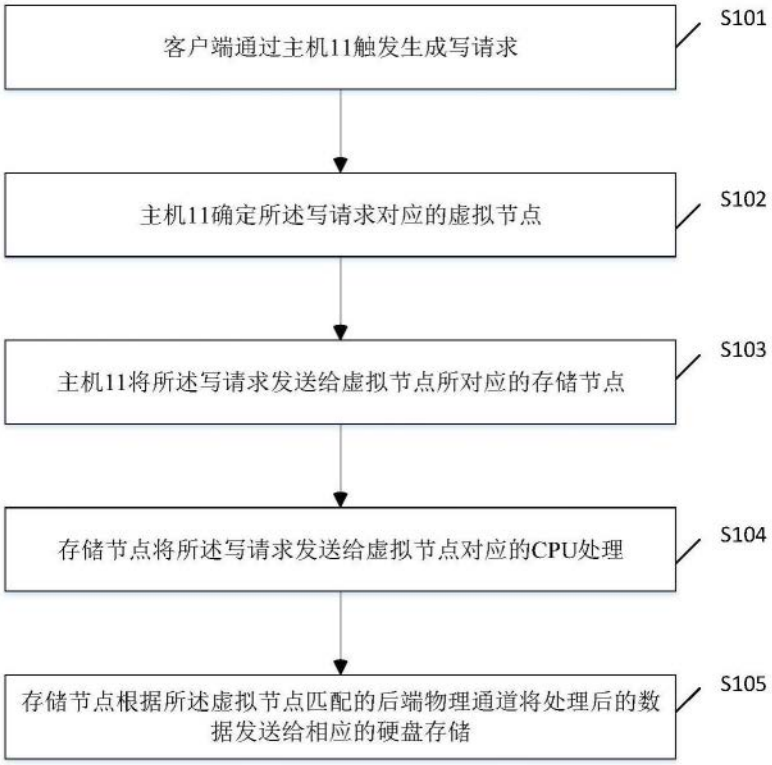

客户端通过主机11触发生成写请求
S101
主机11确定所述写请求对应的虚拟节点
S102
主机11将所述写请求发送给虚拟节点所对应的存储节点
S103
存储节点将所述写请求发送给虚拟节点对应的CPU处理
S104
存储节点根据所述虚拟节点匹配的后端物理通道将处理后的数据发送给相应的硬盘存储
S105

图7

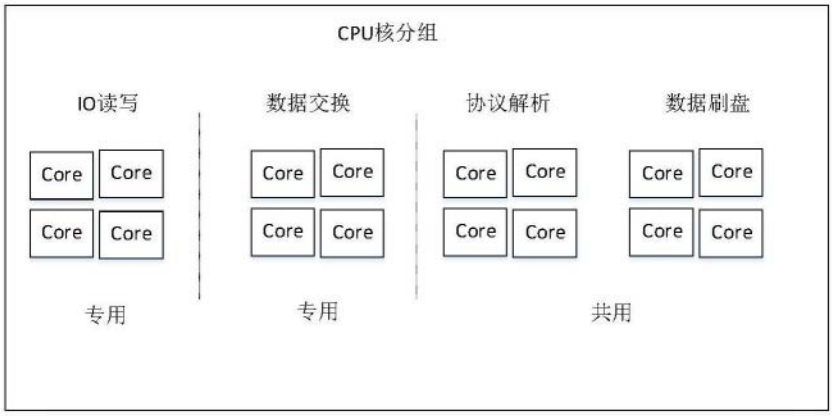

CPU核分组
IO读写
数据交换
协议解析
数据刷盘
Core
专用
专用
共用

图8

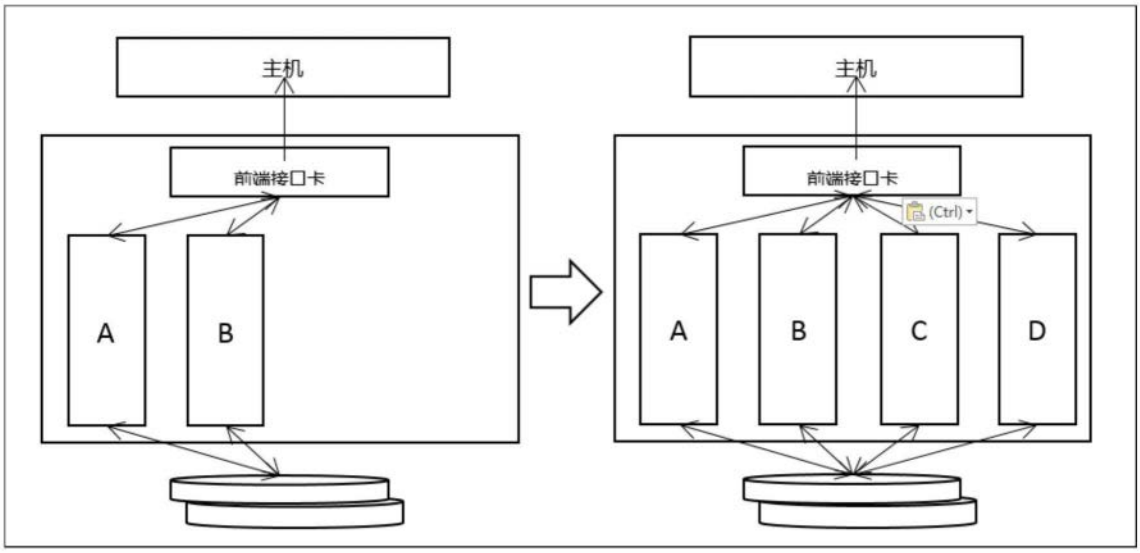

主机
前端接口卡
A
B
主机
前端接口卡
(Ctrl)
A
B
C
D

图9

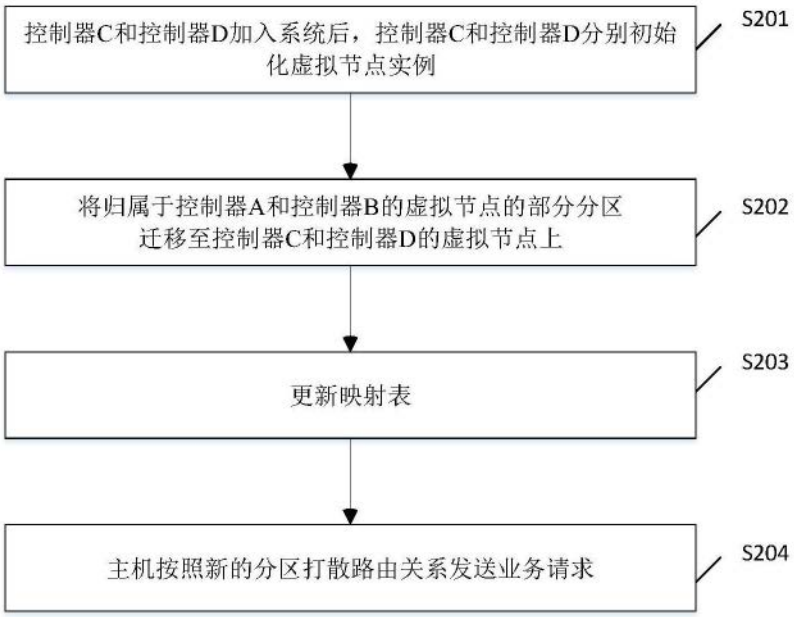

控制器C和控制器D加入系统后，控制器C和控制器D分别初始化虚拟节点实例
S201
将归属于控制器A和控制器B的虚拟节点的部分分区迁移至控制器C和控制器D的虚拟节点上
S202
更新映射表
S203
主机按照新的分区打散路由关系发送业务请求
S204

图10

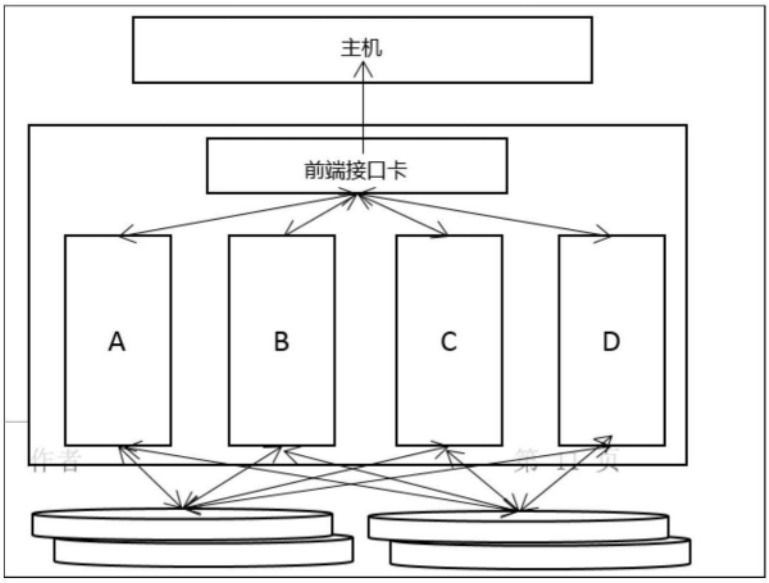

主机
前端接口卡
A
B
C
D

图11

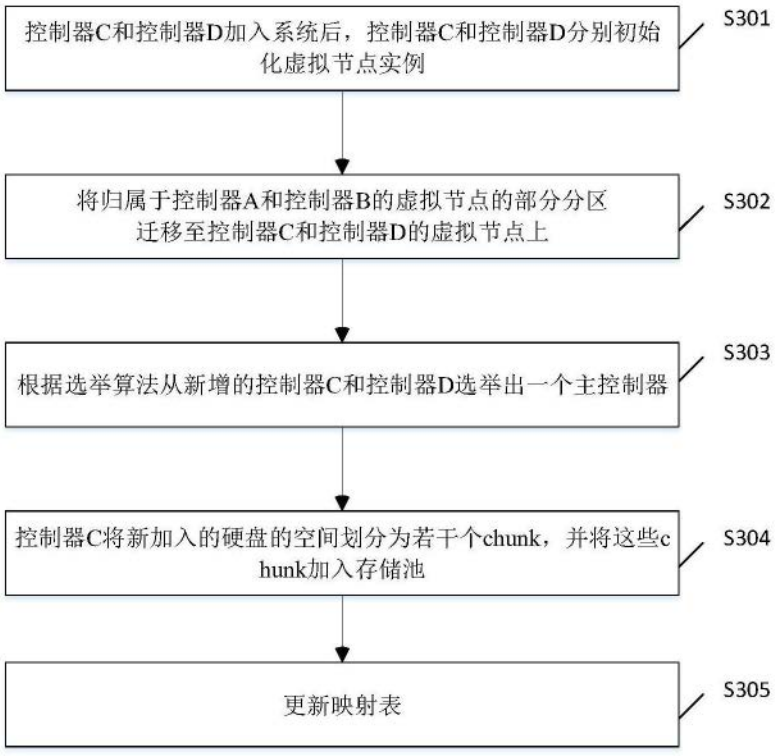

控制器C和控制器D加入系统后，控制器C和控制器D分别初始化虚拟节点实例
S301
将归属于控制器A和控制器B的虚拟节点的部分分区迁移至控制器C和控制器D的虚拟节点上
S302
根据选举算法从新增的控制器C和控制器D选举出一个主控制器
S303
控制器C将新加入的硬盘的空间划分为若干个chunk，并将这些chunk加入存储池
S304
更新映射表
S305

图12