

分布式数据库架构系统、元数据节点及其方法

技术领域

本公开涉及分布式数据库领域，且更具体地，涉及分布式数据库架构系统、用于分布式数据库架构系统的元数据节点、用于分布式数据库架构系统的方法、和用于分布式数据库架构系统的元数据节点的方法。

背景技术

随着互联网的蓬勃发展、数据量和用户量的增长，数据的分析和处理越来越重要。现有的分布式数据库的体系架构在很多场景下仍无法实现低延时和高效访问以及快速升级，仍存在改进空间。

发明内容

鉴于以上，本公开提供了一种分布式数据库架构系统，其能够实现计算节点和存储节点的灵活调配以及数据库系统的快速升级，同时满足低延迟的数据访问。

根据本公开的一个方面，提供了一种分布式数据库架构系统，包括多个计算节点、多个存储节点和元数据节点，其中元数据节点被配置为：记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系，针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中，以及对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。多个计算节点被配置为：根据针对每个数据流的对应关系从多个存储节点读取每个数据流或者向多个存储节点存储每个数据流；以及多个存储节点被配置为：将每个数据流的一个或多个数据块分别存储到针对一个或多个数据块的存储位置中。

可选的，元数据节点还被配置为：响应于两个以上的计算节点中的第一计算节点需要读取第一数据流，生成用于第一计算节点的、对第一数据流的引

用；响应于第一计算节点不再需要读取第一数据流，删除用于第一计算节点的、对第一数据流的引用；以及响应于对第一数据流的所有引用都被删除，指示多个存储节点删除第一数据流。

- 可选的，元数据节点还被配置为：响应于多个计算节点中的第二计算节点期望存储第一数据流的存储请求，为第一数据流所划分成的一个或多个第一数据块分配位于多个存储节点中的一个或多个第一存储节点中的一个或多个第一存储位置，向第二计算节点返回用于存储一个或多个第一数据块的一个或多个第一存储位置的信息，以及记录一个或多个第一数据块中的每个第一数据块与用于存储每个第一数据块的第一存储位置之间的对应关系。第二计算节点被配置为：将第一数据流划分为一个或多个第一数据块，以及根据一个或多个第一存储位置的信息向每个第一存储位置对应的第一存储节点发送一个或多个第一数据块中的对应第一数据块。第一存储节点被配置为：将从第二计算节点接收到的第一数据块存储到对应的第一存储位置中。

- 可选的，元数据节点还被配置为：记录新增加的存储节点的信息，并响应于多个计算节点中的第三计算节点期望存储数据流的请求，向第三计算节点分配包括位于新增加的存储节点中的存储位置的存储位置。

可选的，分布式数据库架构系统还包括控制节点，其中控制节点被配置为：根据新增加的存储节点向元数据节点发送新增加的存储节点的信息。

- 可选的，分布式数据库架构系统还包括控制节点，其中，控制节点被配置为：根据多个计算节点的负载分布，将多个计算节点中的第四计算节点处理的第二数据流调度到多个计算节点中的第五计算节点处理，以及控制元数据节点在调度期间建立用于第五计算节点的、对第二数据流的引用，并且在调度完成后删除用于第四计算节点的、对第二数据流的引用。

- 可选的，分布式数据库架构系统还包括控制节点，其中控制节点被配置为：控制元数据节点将用于处理第一数据流集合的第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用复制以创建用于新增的或者原有的第二计算节点组中的每个计算节点的、对第一数据流集合中的数据流的引用。

可选的，控制节点还被配置为：控制第一数据流集合从第一计算节点组

下线;以及元数据节点还被配置为:删除用于第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用。

可选的,控制节点还被配置为:控制第一计算节点组中的每个计算节点将内存中的数据同步到第二计算节点组中的计算节点的内存中。

- 5 可选的,控制节点还被配置为:在第一计算节点组的内存中的数据与第二计算节点组的内存中的数据之间的差少于第一预定阈值或第一计算节点组与第二计算节点组之间的同步延迟低于第二预定阈值时,控制第一计算节点组停止向其内存中写入数据,在同步完成后,控制第二计算节点组上线,并控制第一计算节点组下线,以及在同步完成后,控制元数据节点删除用于第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用。

可选的,元数据节点还被配置为:对每个引用存储多个副本。

可选的,多个计算节点部署于多个云服务中;和/或多个存储节点部署于多个云服务中。

- 根据本公开的另一方面,提供了一种用于分布式数据库架构系统的元数据节点,分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点,其中元数据节点被配置为:记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系,针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中;对要被两个以上的计算节点读取的第一数据流记录两个以上的引用,每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。

- 根据本公开的另一方面,提供了一种用于分布式数据库架构系统的方法,分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点,其中,方法包括:由元数据节点:记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系,针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中,以及对要被两个以上的计算节点读取的第一数据流记录两个以上的引用,每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流;由多个计算节点:根据针对每个数据流的对应关

系从多个存储节点读取每个数据流或者向多个存储节点存储每个数据流；以及由多个存储节点：将每个数据流的一个或多个数据块分别存储到针对一个或多个数据块的存储位置资源中。

根据本公开的另一方面，提供了一种用于分布式数据库架构系统的元数据节点的方法，分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点，其中方法包括：记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系，针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中；对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。

根据本公开的实施例，通过元数据节点来为每个数据流的数据块分配可以位于不同存储节点中的存储位置，实现了计算节点与存储节点的解绑和灵活调度，并且通过元数据节点记录每个数据流的每个数据块与存储该数据块的存储位置之间的对应关系，使得存储的数据流能够被快速读取。此外，通过对单个数据流创建多个引用分别用于不同的计算节点，使得不同的计算节点能够独立地访问同一数据流，进而能够实现对数据库的灵活调配和快速升级，并能够通过多个引用来管理数据流的生命周期。

附图说明

为了更清楚地说明本公开实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本公开的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图 1 示出了现有的一种分布式数据库架构。

图 2 示出了根据本公开的实施例的分布式数据库架构系统的结构示意图。

图 3 示出了根据本公开的实施例的由存储节点以数据块的方式将数据流存储到各个数据块的存储位置中的示意图。

图 4 示出了根据本公开的实施例的计算节点、元数据节点与存储节点之

间的关系的示意图。

图 5 示出了根据本公开的实施例的写入一个数据流的过程的示意图。

图 6 示出了根据本公开的实施例的在新增存储节点之后如何快速进行读写的示意图。

5 图 7 示出了根据本公开的实施例的数据库快速零拷贝多副本架构的示意图。

图 8 示出了根据本公开的实施例的基于快速数据库克隆的平滑升级的过程的示意图。

图 9 示出了根据本公开的实施例的用于分布式数据库架构系统的方法的流程图。
10

图 10 示出了根据本公开的实施例的用于分布式数据库架构系统的元数据节点的方法的流程图。

具体实施方式

15 现在将详细参照本公开的具体实施例，在附图中例示了本公开的例子。尽管将结合具体实施例描述本公开，但将理解，不是想要将本公开限于描述的实施例。相反，想要覆盖由所附权利要求限定的在本公开的精神和范围内包括的变更、修改和等价物。应注意，这里描述的方法步骤都可以由任何功能块或功能布置来实现，且任何功能块或功能布置可被实现为物理实体或逻辑实体、或者
20 两者两者的组合。

目前的分布式数据库体系架构将计算节点和存储节点绑定，无法实现计算节点和存储节点的灵活调配以及数据库系统的快速升级。例如，当改变或扩展计算节点或存储节点时，需要移动大量数据，耗时较长。例如，通常的数据库采用键-值（key-value）方式存储数据，其中一个键作为唯一标识符与一个
25 值相关联，在查询时，通过键找到该数据的存储位置来找到该数据。但这种数据库在遇到扩增存储节点时，无法立即提供对存储节点的写入服务，因为需要给数据分配键。

图 1 示出了现有的一种分布式数据库架构系统 100。如图 1 所示，分布式数据库架构系统 100 包括计算节点 110、存储节点 120。每个存储节点 120 与

一个计算节点 110 相对应，计算节点 110 控制存储节点 120 中的数据读取和写入。当对计算节点 110 新增存储节点 121，或者将计算节点 110 绑定的存储节点 120 改变为存储节点 121 时，无法立即在存储节点 121 中进行数据写入服务，需要等数据库的一部分数据从已有存储节点 120 移动到存储节点 121 后，才能在存储节点 121 中提供数据写入服务，因而需要花费较长的时间。此外，现有的对象存储的分布式数据库架构无法实现低延迟的访问。

鉴于以上，本公开提出一个新的分布式数据库架构系统，其能够实现计算节点和存储节点的灵活调配以及数据库系统的快速升级，同时满足低延迟的数据访问。本公开的分布式数据库架构系统适用于需要处理大量数据和低延迟访问的场景，给用户高性能、高可用性和灵活性的数据库服务。本公开的分布式数据库架构系统能够基于一个或多个云服务提供厂商提供的标准云服务产品来实现。

图 2 示出了根据本公开的实施例的分布式数据库架构系统 200 的结构示意图。

分布式数据库架构系统 200 可以包括多个计算节点 210、多个存储节点 220 和元数据节点 230。作为示意，图 2 中仅示出了一个计算节点和一个存储节点。可选的，分布式数据库架构系统 200 还可以包括控制节点 240。控制节点 240 可用于向云服务商申请云服务资源来构造计算节点 210、存储节点 220 和元数据节点 230，并可以进行一系列控制，例如监控负载、扩缩容节点、升降级数据库等。

如图 2 中所示，计算节点 210、存储节点 220 和元数据节点 230 可以部署于云服务商提供的云服务资源中。例如，计算节点 210 可以是云服务上分配的計算资源，其可以由例如中央处理单元（CPU）等任意具有处理能力的硬件提供。存储节点 220 可以是云服务上分配的持久性存储资源，其可以由例如磁硬盘和/或固态硬盘（SSD）等任意具有持久存储能力的硬件提供。元数据节点 230 可以是云服务上提供元数据存储和处理的资源，例如其可以由具有处理和存储能力的硬件提供。元数据节点 230 也可以通过计算节点和存储节点来实现。控制节点 240 可以是任意计算资源（例如云服务上分配的計算资源），其可以由例如中央处理单元（CPU）等任意具有处理能力的硬件提供。

控制节点 240 也可以实现在计算节点上。上述提供资源的硬件可以配合相应的软件提供服务。根据本公开的实施例，多个计算节点可以部署于一个或多个云服务中。多个存储节点也可以部署于一个或多个云服务中。

元数据节点 230 可以被配置为记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系。针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中。

在本公开的实施例中，数据流是指数据为了传输或存储而流化后的数据串。数据流可以任意形式的流，例如，字符流、字节流、比特流等。例如，在对数据库的存取中，通常可将数据库中的数据转换为数据流，例如，字节流。一个数据流可以表示一次流化存储的数据所对应的数据流。例如，一个数据流可以对应于数据库（例如数据表）中的一个记录所对应的数据流。在本公开的实施例中，数据库可以是数据流的集合。

数据流可被划分为一个或多个数据块，以便以数据块为单位分散地存储在一个或多个存储节点中的一个或多个存储位置中。通常，数据块是数据流的一段，一个数据流具有多个数据块。在本公开的实施例中，数据流的数据块数量可以是任意的，无上限。在数据流比较小的情况下，一个数据流也可以仅被划分为一个数据块。数据块的尺寸可以根据具体应用场景进行确定。元数据节点、计算节点和存储节点都根据确定的数据块尺寸进行处理。每个数据块可以存储到一个存储位置中。在本公开的实施例中，不同的数据块可以存储到相同或不同存储节点的存储位置中。在一个数据流被划分为多个数据块的情况下，一个数据流的不同数据块可以被存储到相同或不同数据节点的存储位置中。因此，在本公开的实施例中，一个数据流可以存储到不同的存储节点中，实现了数据流的分散和灵活存储。而且，处理数据流的计算节点可以访问多个存储节点，同一个存储节点也可以被多个计算节点访问，消除了计算节点和存储节点的绑定关系。此外，数据流是流化后的数据，因此，数据流及其所划分出的数据块的存取不依赖于键值关系，从而当需要将数据流的数据块存储到新的存储节点时，可以直接存储，而不需要先将其他数据节点的部分数据拷贝到新的数据节点中或者在新的数据节点中创建键值关系，因而可以实现存储节点

的快速扩容。

为了方便对存储的数据流进行快速读取，元数据节点 230 可以记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系，作为元数据之一。换言之，元数据节点 230 以数据块为单位为每个数据块对应地记录存储该数据库的存储位置，即，根据本公开实施例的元数据是数据块级别的元数据。例如，该对应关系可以是数据流中的数据块的标识与该数据块的存储位置的标识之间的映射关系。根据本公开实施例的存储位置可以具有不同的粒度，例如存储节点粒度，存储节点中存储盘粒度，或者存储盘中更具体的存储位置的粒度。在采用存储节点粒度的情况下，元数据节点仅分配和记录存储某个数据块的存储节点，而具体存储到该存储节点的哪个存储盘及该存储盘中的哪个具体位置可以由该存储节点决定。在此情况下，上述存储位置的标识可以仅包括存储节点的标识。在采用存储节点中存储盘粒度的情况下，元数据节点分配和记录存储某个数据块的存储节点及该存储节点中存储该数据块的存储盘，而具体存储到该存储盘中的哪个具体位置可以由该存储节点决定。在此情况下，上述存储位置的标识可以包括存储节点的标识以及存储节点中的存储盘的标识。在采用存储盘中更具体的存储位置的粒度的情况下，位置的标识除了包括存储节点的标识以及存储节点中的存储盘的标识之外还需要包括更具体的存储位置信息。基于上述对应关系，可以快速寻找到存储某个数据流的位置，实现数据的快速读取。该对应关系的实现方式不限于上述具体实施例，只要实现上述目的即可。

以下表 1 和表 2 示出了上述对应关系的两个示例。表 1 示出了存储位置为存储节点粒度的示例，表 2 示出了存储位置为存储盘粒度的示例。

表 1

数据流标识	数据块标识	存储位置
1	1	存储节点 1
1	2	存储节点 1
1	3	存储节点 2
2	1	存储节点 1
2	2	存储节点 2

如表 1 所示，数据流 1 被划分成 3 个数据块 1、2、3，其中数据块 1、2 存储在存储节点 1 中，数据块 3 存储在存储节点 2 中。因此，对应关系可以包括数据流标识 1 对应的数据块标识 1、2、3 分别对应于存储节点 1、1、2。另外，数据流 2 被划分成 2 个数据块 1、2，数据块 1 存储在存储节点 1 中，
5 数据块 2 存储在存储节点 2 中。因此，对应关系可以包括数据流标识 2 对应的数据块标识 1、2 分别对应于存储节点 1、2。

表 2

数据流标识	数据块标识	存储位置
1	1	存储节点 1 中的磁盘 C (1-C)
1	2	存储节点 1 中的磁盘 D (1-D)
1	3	存储节点 2 中的磁盘 C (2-C)
2	1	存储节点 1 中的磁盘 D (1-D)
2	2	存储节点 2 中的磁盘 C (2-C)

如表 2 所示，数据流 1 被划分成 3 个数据块 1、2、3，其中数据块 1、2 分别存储在存储节点 1 的磁盘 C 和 D (例如，可以分别以 1-C、1-D 来标识)
10 中，数据块 3 存储在存储节点 2 中的磁盘 C (例如，可以以 2-C 来标识) 中。因此，对应关系可以包括数据流标识 1 对应的数据块标识 1、2、3 分别对应于存储位置标识 1-C、1-D、2-C。另外，数据流 2 被划分成 2 个数据块 1、2，数据块 1 存储在存储节点 1 中的磁盘 D (例如，可以以 3-D 来标识) 中，数据块 2 存储在存储节点 2 中的磁盘 C (例如，可以以 4-C 来标识) 中。因此，
15 对应关系可以包括数据流标识 2 对应的数据块标识 1、2 分别对应于存储位置 3-D、4-C。

需要说明的是，上述表 1 和表 2 仅是对应关系的示例，本公开不限于此，对应关系还可以采用其他形式。例如，上述表 1 和表 2 中的数据流标识和数据块标识可以合并，例如，数据流 1 的数据块 1 可以表示为数据块 1-1，数据
20 流 1 的数据块 2 可以表示为数据块 1-2，数据流 1 的数据块 3 可以表示为数据块 1-3，数据流 2 的数据块 1 可以表示为数据块 2-1，数据流 2 的数据块 2 可以表示为数据块 2-2。

可选的，元数据节点中的这种对应关系以及其他元数据可以被复制为多

个副本, 保证元数据的可靠性, 在某个副本损坏时可以通过其他副本提供元数据。此外, 元数据节点还可以根据数据冗余策略, 将每个数据块的一个或者多个副本分配到匹配的存储节点中。在副本缺失的情况, 元数据节点可以向存储节点发起数据块的复制请求, 以满足数据的冗余策略。

- 5 多个计算节点 210 可以被配置为根据针对每个数据流的对应关系从多个存储节点读取每个数据流或者向多个存储节点存储每个数据流。根据本公开的实施例, 由于通过元数据节点来记录每个数据流所划分成的一个或多个数据块与存储这些数据块的存储位置之间的对应关系, 且多个计算节点都可以访问元数据节点中的对应关系, 从而多个计算节点都可以根据该对应关系来
- 10 访问多个存储节点中的数据流的这些数据块。因此, 计算节点和存储节点的绑定关系被消除, 一个计算节点可以根据需要访问多个存储节点, 同一个存储节点也可以根据需求被多个计算节点访问。这可以通过元数据节点来对数据流的各个数据块分配存储位置而实现。

- 对于存储操作而言, 元数据节点 230 可以基于某个计算节点期望存储数
- 15 据流的请求, 为该数据流所划分成的一个或多个第一数据块分配一个或多个存储位置, 并向该计算节点返回与所分配的存储位置的信息。从而, 该计算节点可以根据每个存储位置的信息向对应的存储节点发送对应的数据块, 以便将该数据库存储在对应的存储位置中。由此可见, 计算节点可以根据元数据节点所分配的存储节点, 将相应的数据块存储到对应的数据节点中, 从而可以不
- 20 限于存储到固定的存储节点中。元数据节点可以根据系统中的存储节点及其存储状态等信息灵活分配存储位置。需要说明的是, 上述存储节点的标识信息也被包括在数据流的数据块与存储位置的对应关系中, 从而计算节点根据对应关系可以确定对应的存储节点。

- 对于读取操作而言, 若某个计算节点期望读取某个数据流, 则可以从元
- 25 数据节点中获取该数据流的数据块与存储位置之间的对应关系, 继而从对应的存储节点的相应存储位置中读取该数据流的各个数据块, 从而读取整个数据流。

相应地, 多个存储节点 220 可以被配置为将每个数据流的一个或多个数据块分别存储到针对一个或多个数据块的存储位置中。例如, 从计算节点接收

到存储某个数据块的请求的存储节点，可以根据请求中包含的存储位置信息将该数据块存储到某个存储地址。例如，如果存储位置信息包括目标存储盘的信息，则存储节点将该数据库存储到该目标存储盘中，具体的存储地址由存储节点根据自身的存储策略确定；如果存储位置信息本身包括更具体的存储地址，则存储节点将该数据库存储到存储位置信息中包含的存储地址中；如果存储位置信息仅有存储节点信息而没有更进一步的信息，则由存储节点根据自身的存储策略确定具体的存储盘和地址。

图 3 示出了根据本公开的实施例的由存储节点以数据块的方式将数据流存储到数据块资源中的示意图。

10 存储节点是存储数据的节点，其可以包括且负责管理存储介质。存储介质可以是任何形式的持久性存储介质，例如磁盘、固态硬盘等。存储节点将数据块存储到存储介质中的存储地址中。存储节点可以处理数据块的读和写，还可以处理数据块的删除、复制请求，以及对数据块的完整性进行校验，保证数据的持久性和正确性。存储节点可以支持多计算节点同时读取数据，包括同时
15 读取同一份数据。

如图 3 所示，数据节点中的存储介质以数据块的方式将数据流进行存储。一个数据流可以被划分为多个数据块，从而存储到多个存储节点中。数据流所划分成的数据块与存储数据块的存储位置之间存在对应关系。如此，一个数据流可以存储到不同的数据节点中，实现了数据流的分散和灵活存储。而且，同
20 一个计算节点可以访问多个存储节点，同一个存储节点也可以被多个计算节点访问，消除了计算节点和存储节点的绑定关系，实现了计算节点和存储节点之间的灵活调度。

返回图 2，元数据节点 230 还可以被配置为对要被两个以上的计算节点读取的第一数据流记录两个以上的引用。每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。根据
25 本公开的实施例，可以对一个数据流创建多个引用，每个引用用于一个计算节点，从而多个引用可以用于多个不同的计算节点。对数据流的引用表示对该数据流的关联关系，例如，其可以是数据流的一个标识符，该标识符关联到该数据流。由于引用能够关联到数据流，从而根据引用能够获得数据流的数据块与

存储该数据流的数据块的存储位置之间的对应关系，继而能够对该数据流进行存储和读取。对一个数据流记录多个引用例如可以是对该数据流记录多个标识符，每个标识符用于一个计算节点。这些标识符的值可以相同也可以不同。对同一个数据流针对不同的计算节点设置各自的引用（例如标识符）的情况下，从计算节点的角度，表现为其独自享有了该数据流。从而，当某个计算节点需要读取某个数据流时，只要对该计算节点增加一个用于该计算节点的引用。相反，若该计算节点不再需要读取该数据流时，只要删除用于该计算节点的引用。由此，可以实现对数据库的灵活调配和快速升级。

此外，元数据节点还可以通过对不同计算节点设置不同引用来管理数据流的生命周期以及某个计算节点能够访问该数据流的生命周期。具体地，元数据节点可以被配置为：响应于两个以上的计算节点中的第一计算节点需要读取第一数据流，生成用于第一计算节点的、对第一数据流的引用；响应于第一计算节点不再需要读取第一数据流，删除用于第一计算节点的、对第一数据流的引用；以及响应于对第一数据流的所有引用都被删除，指示多个存储节点删除第一数据流。

数据流的生命周期从数据流被存储开始，即从元数据节点创建第一个引用开始，到数据流被删除为止，即到该数据流的所有引用都被删除为止。每个计算节点访问该数据流的生命周期从该计算节点需要访问该数据流开始，即从为该计算节点创建引用开始，到该计算节点不再需要访问该数据流为止，即到删除用于该计算节点的引用为止。元数据节点通过针对不同计算节点创建各自的引用，能够清楚地知道哪些计算节点可能会访问该数据流，并可以灵活地管理该数据流的生命周期以及每个计算节点访问该数据流的生命周期。

以下表 3 示出了上述引用的一个例子，但本公开不限于此。

表 3

引用	计算节点	数据流
计算节点 1-数据流 1	计算节点 1	数据流 1
计算节点 1-数据流 2	计算节点 1	数据流 2
计算节点 2-数据流 1	计算节点 2	数据流 1
计算节点 2-数据流 2	计算节点 2	数据流 2

计算节点 3-数据流 3	计算节点 3	数据流 3
计算节点 4-数据流 3	计算节点 4	数据流 3

如表 3 所示，例如，如果计算节点 1 和 2 都想要访问（读取）数据流 1，则为计算节点 1 建立对数据流 1 的引用（例如，标识为“计算节点 1-数据流 1”），同时为计算节点 2 建立对数据流 1 的引用（例如，标识为“计算节点 2-数据流 1”）。如此，计算节点 1 和 2 可以同时访问数据流 1。如果计算节点 1 和 2 都想要访问数据流 2，则为计算节点 1 建立对数据流 2 的引用（例如，标识为“计算节点 1-数据流 2”），同时为计算节点 2 建立对数据流 2 的引用（例如，标识为“计算节点 2-数据流 2”）。如此，计算节点 1 和 2 可以同时访问数据流 2。对于计算节点 3 和 4 也是类似，在此不赘述。当然，引用的形式不限于表 3 中示例的形式，引用只需要能指示服务于哪个计算节点且用于读取哪个数据流即可。

可选的，元数据节点还可以对每个引用存储多个副本，以满足数据冗余要求，以便在一个副本发生损坏时可以从其他副本获取引用信息。

图 4 示出了根据本公开的实施例的计算节点、元数据节点与存储节点之间的关系的示例示意图。

如图 4 所示，元数据节点 430 记录了第一数据流所划分的多个数据块 4211、4212、4221 与存储节点 421、422 中用于存储第一数据流的多个数据块 4211、4212、4221 的多个存储位置之间的对应关系 437，以及第二数据流与存储节点 422、423 中用于存储第二数据流所划分的多个数据块 4222、4231、4232 的多个存储位置之间的对应关系 438。

例如，对应关系 437 用于将第一数据流的 2 个数据块 4211 和 4212 存储在存储节点 421 中以及将另一数据块 4221 存储在存储节点 422 中。相应地，对应关系 437 可以用于从存储节点 421 及存储节点 422 中读取 3 个数据块 4211 和 4212、4221，以完整读取第一数据流。对应关系 438 用于将第二数据流的 2 个数据块 4231 和 4232 存储在存储节点 423 中以及将另一数据块 4222 存储在存储节点 422 中。相应地，对应关系 438 可以用于从在存储节点 423 及存储节点 422 中读取 3 个数据块 4231 和 4232、4222，以完整读取第二数据流。

在现有技术的分布式存储服务中，计算节点与其能够访问的存储节点的关系是绑定的，并且一个计算节点所处理的数据流也难以存储到多个存储节点中。如果新增一个存储节点，则需要移动数据到新增的存储节点以便将原有的计算节点绑定到新增的存储节点，这无法实现快速扩容或迁移。然而，根据
5 本公开的实施例，计算节点和存储节点可以通过在元数据节点中建立和记录的数据流的数据块与存储位置之间的关系实现计算节点与存储节点的灵活分配。

此外，如图 4 所示，元数据节点 430 还可以对要被两个以上计算节点读取的第一数据流记录两个以上引用，作为元数据的一部分。每个引用被两个以
10 上计算节点中的每个计算节点用来获取第一数据流的对应关系以读取第一数据流。对第一数据流的引用的数量可以与要访问第一数据流的计算节点的数量相同，对该第一数据流的一个引用对应于要访问该第一数据流的一个计算节点，专用于该计算节点来使用该对应关系，使得各个计算节点可以独立地使用
15 该引用来存取数据，互不干扰。

例如，元数据节点 430 为第一数据流记录了多个引用 431、432、433、434 分别用于计算节点 411、412、413、414 来获取针对第一数据流的对应关系 437 以便访问该数据流的各个数据块 4211、4212、4221。引用 431、432、433、434 指向相同的数据流（第一数据流），但分别用于计算节点 411、412、413、414。
从而，每个计算节点 411、412、413、414 可以同时获取到第一数据流的各个
20 数据块的对应关系 437，从而能够同时且独立地访问第一数据流中的数据块 4211、4212、4221。即，一个数据流可以同时被多个计算节点访问。从计算节点的角度，多个引用表现为每个计算节点都有各自的数据流，从而具有各自的数据库，即一个或多个数据流构成的数据流集合。从数据库的角度，多个引用表现为数据库被以零拷贝的方式复制为多个数据库，分别用于不同的计算节
25 点。不同的计算节点各自具有自己的引用相对于共用同一个引用，可以实现对数据库的灵活调配和快速升级。

另外，一个计算节点也可以使用针对多个数据流的多个引用来访问（即读取）多个数据流，例如，计算节点 413 要访问第一数据流和第二数据流，则元数据节点 430 为计算节点 413 创建引用 433 来访问第一数据流，且创建引用

435 来访问第二数据流。

引用的具体形式可以多种多样。例如，引用可以是数据流的一个标识符，其关联到该数据流。此外，引用还可以附加包括使用该引用的计算节点的标识符，从而指示哪个计算节点使用该引用。当然，也可以隐式表示哪个计算节点使用该引用，例如，记录在特定位置的引用隐含表示特定于某个计算节点，或者该引用的值本身隐含表示特定于某个计算节点。此外，引用也可以包含数据流的数据块与存储位置之间的对应关系，例如，可以是该对应关系本身，即可以为每个要访问该数据流的计算节点记录一个对应关系副本专用于该计算节点。针对引用的操作是一种纯元数据的操作，不涉及数据的拷贝，可以快速地完成。在为某个计算节点建立对一个数据流的引用之后，该计算节点将会通过该引用看见一个独立的数据流。当某个数据流的所有引用都被释放（即，删除）后，元数据节点会对数据流包含的数据块所在的存储节点发出删除请求，并且删除该数据流和数据块的元数据，从而可以通过引用管理数据流的生命周期以及某个计算节点能够访问该数据流的生命周期。

根据本公开的其他实施例，一个引用也可以被多个计算节点使用来读取同一数据流，此时，如果某个计算节点不再使用该引用，该引用也不会被删除而是保留给其他计算节点继续使用。例如，在一种应用场景中，如果一个计算节点对该数据流的吞吐量较大的情况下，可以临时建立其它计算节点来使用该计算节点的引用分担该计算节点的吞吐量。在不需要其它计算节点来分担时，可以使其它计算节点下线，但仍保留该数据流的引用供该一个计算节点使用。

根据本公开的实施例，通过元数据节点来为每个数据流的数据块分配可以位于不同存储节点中的存储位置，实现了计算节点与存储节点的解绑和灵活调度，并且通过元数据节点记录每个数据流的每个数据块与存储该数据块的存储位置之间的对应关系，使得存储的数据流能够被快速读取。此外，通过对单个数据流创建多个引用分别用于不同的计算节点，使得不同的计算节点能够独立地访问同一数据流，进而能够实现对数据库的灵活调配和快速升级。

图 5 示出了根据本公开的实施例的写入一个数据流的过程的示意图。

图 5 中以第二计算节点 511 发起写入（存储）第一数据流为例进行说明。

首先，第二计算节点 511 向元数据节点 530 发送期望存储第一数据流的存储请求。

元数据节点 530 响应于第二计算节点 510 期望存储第一数据流的存储请求，为第一数据流所划分成的一个或多个第一数据块分配位于多个存储节点中的一个或多个第一存储节点 521、522 中的一个或多个第一存储位置，并向第二计算节点 510 返回用于存储一个或多个第一数据块的一个或多个第一存储位置的信息，以及记录一个或多个第一数据块中的每个第一数据块与用于存储每个第一数据块的第一存储位置之间的对应关系。例如，第一数据流被划分为两个数据块，第一个数据块被分配存储于第一存储节点 521 中，从而可以针对第一个数据块返回第一存储节点 521 的标识信息 ID1；第二个数据块被分配存储于第一存储节点 522 中，从而可以针对第二个数据块返回第一存储节点 522 的标识信息 ID2。当然，若上述存储位置除存储节点之外还包括更具体的存储位置，则返回的存储位置的信息还可以该更具体的存储位置的信息，例如存储节点中的存储盘的信息。

第二计算节点 510 将第一数据流划分为一个或多个第一数据块（图 5 中为两个数据块），并且根据一个或多个第一存储位置的信息向每个第一存储位置对应的第一存储节点 521、522 发送一个或多个第一数据块中的对应第一数据块。例如，第一个数据块的存储位置位于第一存储节点 521 中，则在存储第一个数据块时，第二计算节点 510 根据第一存储节点 521 的标识信息 ID1 将其发送到第一存储节点 521 中。第二个数据块对应的存储位置位于第一存储节点 522 中，则在存储第二个数据块时，第二计算节点 510 根据第一存储节点 522 的标识信息 ID2 将其发送到第一存储节点 522 中。

在一个实施例中，在第二计算节点 510 向对应第一存储节点 521、522 发送对应数据块之前，可以将对应数据块写到内存中，直到内存中的数据达到预定数据量，第二计算节点 510 再异步地将内存中缓存的数据块写入第一存储节点。

最后，第一存储节点 521、522 将从第二计算节点 510 接收到的第一数据块存储到对应的第一存储位置中。

下面介绍在不同场景下根据本公开的实施例的具体实施方式及技术效果。

场景 1——存储节点快速扩容

随着时间推移，数据库中存储的数据量可能快速增长。如果存储空间不足，数据库将无法容纳新数据，导致系统崩溃或无法正常工作。同时，当数据库存储空间接近满时，数据库性能会受到影响。读取和写入操作可能变慢，查询响应时间增加，导致用户体验下降。可见快速扩容存储空间对于确保系统正常运行、提高性能和保护数据安全非常重要。

因此，在增加存储节点以扩容存储空间的情况下，元数据节点可以被配置为：记录新增加的存储节点的信息，并响应于多个计算节点中的第三计算节点期望存储数据流的请求，向第三计算节点分配包括位于新增加的存储节点中的存储位置的存储位置。在一个实施方式中，可以由控制节点向元数据节点提供新增加的存储节点的信息，即，控制节点可以被配置为根据新增加的存储节点向元数据节点发送新增加的存储节点的信息。在另一实施例方式中，也可以由新增加的存储节点自己向元数据节点发送它自己的信息。

图 6 示出了根据本公开的实施例的在新增存储节点之后如何快速进行读取和写入的示意图。

假设新增了存储节点 623，相比于传统的方式要将已有存储节点 621、622 中的一部分数据移动到新增存储节点 623 中，根据本公开的实施例，可以不移动数据，而是由元数据节点 630 响应于多个计算节点 611、612、613、614 中的某个计算节点 613 期望存储数据流的请求，向计算节点 613 分配包括位于新增加的存储节点 623 中的存储位置的存储位置，以便存储数据流的数据块 6231、6232、6222。如图 6 所示，数据块 6231、6232 位于新增加的存储节点 623 中，而数据块 6222 位于原有存储节点 622 中。当然，也可以所有数据块都存储于新增加的存储节点 623 中。元数据节点 630 记录该数据流的各个数据块与存储该数据流的各个数据块的存储位置之间的对应关系 638。

此外，元数据节点 630 还可以为该数据流记录多个引用 635、636，分别被计算节点 613、614 用来获取针对该数据流的对应关系 638 以访问该数据流。

如此，根据先前阐述的数据流的存储和读取操作，可以立即在新增加的存储节点中进行数据流的存储操作，并可以继而进行读取操作。相比于现有数据库架构在存储节点扩容时不能立即提供相应的写入服务的问题，根据本公

开的实施例，在存储层向计算节点暴露的是对数据块级别的一种抽象，一个数据块可以被分配到任意存储节点中的任意存储位置上存储，所以在新增加一个存储节点的情况下，可以立刻提供这种数据块级的写入服务。

场景 2——零拷贝动态负载均衡

- 5 随着数据的不断写入，某些数据库(例如表)包含的数据流在计算节点中分布会变化，不同数据的请求的频率也会发生变化，导致某些计算节点可能需要处理更多的数据量，从而导致性能瓶颈。根据本公开的实施例能够在不拷贝数据块的情况下执行动态负载均衡机制，避免因为数据分布导致的性能瓶颈。

- 具体地，控制节点被配置为：根据多个计算节点的负载分布，将多个计算
10 节点中的第四计算节点处理的第二数据流调度到多个计算节点中的第五计算节点处理，以及控制元数据节点在调度期间建立用于第五计算节点的、对第二数据流的引用，并且在调度完成后删除用于第四计算节点的、对第二数据流的引用。

- 例如，控制节点可以获得每个计算节点实时记录的本计算节点具有引用
15 的数据量和它所带来的负载，从而得知多个计算节点的负载分布。如果控制节点发现这种负载分布不是最优的，或存在负载分布不均衡的情况，例如某些计算节点处的负载明显高于其他计算节点的负载，则需要将负载高的那些计算节点（例如第四计算节点）上的部分数据流（例如第二数据流）调度到其他计算节点来处理（例如第五计算节点）。

- 20 在建立了用于第五计算节点的、对第二数据流的引用，并且在调度完成后，第五计算节点可以在不需要拷贝或移动第二数据流的情况下接管该第二数据流，并且可以删除用于第四计算节点的、对第二数据流的引用。如此，依托于元数据节点提供的高效的数据流的引用的机制，在数据流零拷贝的基础上，数据流能很快被其他计算节点接管，从而消除系统的性能瓶颈。

- 25 此外，在调度过程中，由于第四计算节点对该第二数据流的原有引用仍然有效，第四计算节点仍然可访问第二数据流，因此，调度过程不会影响到第四计算节点对第二数据流的读取请求的处理。而在调度完成后，用户对第二数据流的读取请求会自动路由到新的第五计算节点上，而第四计算节点的老的引用会被删除，此时第五计算节点完全接管该第二数据流。

场景 3——计算节点组灵活实现数据库的资源共享和独享

数据库资源共享可以减少硬件和软件的成本，资源独享可以保证服务的稳定性，提供更高的安全性。根据本公开的实施例，可以把两个数据库（数据流集合）分配到同一组计算节点，让它们共享同一组计算资源（即，资源共享），
5 例如，对两个数据库的引用可以由同一个计算节点来读取；也可以把两个数据库分配到两组不同的计算节点，物理上保证它们的隔离（即，资源独享）。这里，数据库中的一个数据条目可以流化为一个数据流，因此，一个数据库可以表示为一个数据流集合，一个数据流集合可以包括一个或多个数据流。
根据本公开的实施例可以零拷贝数据块地将一个数据库从一组计算节点迁移
10 到另一组计算节点，如此，能够在数据库共享资源和独享资源之间快速切换。

具体的，控制节点可以被配置为：控制元数据节点将用于处理第一数据流集合的第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用复制以创建用于原有的第二计算节点组中的每个计算节点的、对第一数据流集合中的数据流的引用；并且控制第一数据流集合从第一计算节点组下
15 线。元数据节点可以被配置为：删除用于第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用。如此，可以零拷贝数据块地将第一数据流集合从第一计算节点组迁移到第二计算节点组，由第二计算机节点组对第一数据流集合进行访问和向外提供服务。

场景 4——数据库快速零拷贝多副本架构

20 当数据库通过水平扩容仍然满足不了用户请求时，需要垂直扩容，传统的数据库需要将数据进行一次物理拷贝才可以扩容，这种做法不仅浪费了存储空间，同时速度也比较慢。根据本公开的实施例，可以通过多个计算节点引用同一份数据并且提供服务来增加数据库的服务能力。换言之，根据本公开的实施例，可以通过增加计算节点而不拷贝数据流来达到增加处理数据库的服务
25 能力的目的。

图 7 示出了根据本公开的实施例的数据库快速零拷贝多副本架构的示意图。

如图 7 所示，首先，控制节点 740 可以申请云资源来选择或者构造新的计算节点组 712，例如，前述实施例中的新增的第二计算节点组。具体地，控

制节点 740 可以被配置为：控制元数据节点 730 将用于处理第一数据流集合（第一数据库）的第一计算节点组 711 的每个计算节点的、对第一数据流集合中的数据流的引用复制以创建用于新增的第二计算节点组 712 中的每个计算节点的、对第一数据流集合中的数据流的引用。

5 同时，为了使得新增的第二计算节点组 712 能够获得第一计算节点组 711 的完全信息，控制节点还被配置为：控制第一计算节点组 711 中的每个计算节点将内存中的数据同步到第二计算节点组 712 中的计算节点的内存中。

例如，第一计算节点组 711 的内存可能加载了将要存储到存储节点中的数据，为了能完全同步第一计算节点组 711 的当前进程，第二计算节点组 712
10 可以获得第一计算节点组 711 的内存中的数据并将其存储到自己的内存中，以根据内存中的数据进行数据库的操作。在第二计算节点组 712 的内存得到了第一计算节点组 711 的内存的数据之后，能够直接从自己的内存中直接读取该数据，提高了数据的读取速度。例如，同步的方式可以是：控制节点 740
15 控制第二计算节点组 712 通过发送网络请求的方式向第一计算节点组 711 的每个计算节点获取当前的内存数据，并同步加载到第二计算节点组 712 的内存中。

如此，由于元数据节点 730 同时记录了第一计算节点组 711 对第一数据流集合的引用和第二计算节点组 712 对第一数据流集合的引用，并且内存中的实时数据也进行了同步，所以，第一计算节点组 711 和第二计算节点组 712
20 都能够对第一数据流集合进行处理，实现了数据库的快速零拷贝多副本。

场景 5——零拷贝快速克隆可读可写的数据库

用户在数据库的使用过程中，希望能够快速地在一份数据中进行多个不同的实验，并且希望实验之间中互不干扰。根据本公开的实施例，可以零拷贝地快速克隆一份可读可写的数据库，提供给不同的业务方同时进行实验，用户可以
25 在新的克隆后的数据库上继续发起新的读写请求，也可以继续克隆新的数据库。这一使用体验大大提高了业务迭代的效率，同时降低了使用成本。

根据本公开的实施例，为了实现零拷贝地快速克隆一份可读可写的数据库，只需要对该数据库中的所有数据流都新复制一份新的引用，该新的引用从计算节点的角度表现为一份新的数据块。当需要该新的数据块加载给某个计

算节点时，只需要将相应引用分配给该计算节点即可。

场景 6——基于快速数据库克隆的平滑升级

数据库平滑升级可以保证系统的正常运行，减少停机时间和数据丢失的风险，提高系统可用性和用户体验，同时还可以带来性能和功能的改进。但是，
5 在现有技术中，对于各节点的状态需要协同服务的系统升级，没有特别好的解决方案，既保障高服务级别协议（Service Level Agreement, SLA），又降低数据存储和管理成本。

图 8 示出了根据本公开的实施例的基于快速数据库克隆的平滑升级的过程的示意图。

10 如图 8 所示，控制节点 860 可以接收客户端 850 的指令而确定要从计算第一计算节点组 811 升级到第二计算节点组 812。第二计算节点组 812 可以是控制节点 860 从云服务新申请的一组计算节点。

类似于场景 4，控制节点 860 可以被配置为：控制元数据节点将用于处理第一数据流集合的第一计算节点组 811 的每个计算节点的、对第一数据流集合中的数据流的引用复制以创建用于新增的第二计算节点组 812 中的每个计算节点的、对第一数据流集合中的数据流的引用；控制第一计算节点组 811 中的每个计算节点将内存中的数据同步到第二计算节点组 812 中的计算节点的内存中。
15

为了提升用户体验和数据库平滑升级，控制节点 860 还可以被配置为：
20 在第一计算节点组 811 的内存中的数据与第二计算节点组 812 的内存中的数据之间的差少于第一预定阈值或第一计算节点组与第二计算节点组之间的同步延迟低于第二预定阈值时，控制第一计算节点组 811 停止向其内存中写入数据，在同步完成后，控制第二计算节点组 812 上线，并控制第一计算节点组 811 下线，以及在同步完成后，控制元数据节点删除用于第一计算节点组 811 的每个计算节点的、对第一数据流集合中的数据流的引用。在本公开的实施例中，计算节点的上线表示该计算节点可以对相应的数据库进行读写操作。
25

该第一预定阈值或第二预定阈值决定了不可写操作的时间，阈值越大，停止写后需要同步数据的数据量越大，同步的时间越久；阈值越小，停止写后需要同步数据的数据量越小，同步的时间越短。同步完成后，第二计算节点组

上线,可以通过第二计算节点组进行后续数据的写入,例如先写入到第二节点组的内存中然后在存储到存储节点中,或者直接存储到存储节点中。因此,从用户的角度,仅在第一计算节点组停止写入至第二计算节点组同步完成上线前不能进行写入操作。因此,通过设置合适的阈值,用户可以仅在很短的时间
5 内不能进行写操作,使得用户甚至感知不到该停止过程,提高了用户体验。

例如,当在第一计算节点组 811 的内存中的数据与第二计算节点组 812 的内存中的数据之间的差较少时,由于两个计算节点组内存中相差的数据量较少,那么此时停止写处理,使得第一计算节点组能将内存中相差较少的数据很快同步到第二计算节点组的内存中。从而,第二计算节点组快速地在同步完
10 成后上线,继而进行后续数据的写操作。通过设置合适的第一阈值使得该过程时间很短,用户可能甚至感知不到停止写的过程,从而给用户带来更好的使用体验。

再例如,如果第一计算节点组 811 与第二计算节点组 812 之间的同步延迟较低,则第一计算节点组 811 的内存中的数据与第二计算节点组 812 的内存中的数据由于能够快速同步因而差别较小,此时适合停止第一计算节点组
15 811 的写操作。而且由于同步延迟小,内存中的差别数据被进一步同步的速度也较快,因而后续进一步同步的时间更短。因而,此时合适停止第一计算节点组 811 的写操作而完成第一计算节点组 811 与第二计算节点组 812 的同步,以便给用户带来更好的使用体验。

上述第一阈值和第二阈值可以基于系统需求而根据经验或实验进行设置。
20 例如,根据预期停止写操作的最大时间长度(例如,1秒)参照历史经验值确定两个计算节点组之间的数据差或者同步延迟作为第一阈值或第二阈值。

此外,可以通过各种技术手段确定第一计算节点组的内存中的数据与第二计算节点组的内存中的数据之间的差,或者第一计算节点组与第二计算节点组之间的同步延迟。例如,计算节点组的每个计算节点可以监视写入内存中的数据,并将相应数据报告给汇总节点(例如控制节点或者某个计算节点),
25 从而汇总节点能够确定各个计算节点的内存中的数据差。再例如,可以通过第一计算节点组的某个计算节点发送数据的时刻与第二计算节点组的某个计算节点接收到该数据的时刻之间的差来确定同步延迟。在本公开的实施例中,同

步延迟表示从一个节点发出数据到另一个节点完整接收到该数据的时间差，包括接收数据时的解码和处理时间。

需要说明的是，在本公开的实施例中，“计算节点组”的部件和操作表示的是该计算节点组中的计算节点的部件和操作。例如，“第一/第二计算节点组”

5 的内存表示的是“第一/第二计算节点组”中的计算节点的内存，其可以是所有计算节点的内存也可以是代表性的计算节点的内存。再例如，第一计算节点组与第二计算节点组之间的同步延迟可以是两个计算节点组中的所有对应计算节点之间的同步延迟的平均，也可以是两个计算节点组中的代表性的一个或多个计算节点之间的同步延迟。

10 通过上述过程，在不需要将第一数据流集合（数据块）进行实际拷贝的情况下平滑地将数据库服务从第一计算节点组 811 升级到第二计算节点组。在上述整个升级流程中，可以一直为用户提供数据库的读取服务，而且写处理也仅有短暂的停止，大约在秒级，因此，用户几乎感受不到卡顿，为用户带来了良好的体验。

15 **场景 7——数据库的快速备份和恢复**

数据库备份是保护数据安全、提供数据恢复和灾难恢复能力的重要手段，对于保障业务连续性和数据可靠性至关重要。

传统的数据库备份时，需要物理地拷贝数据，极大地浪费了存储空间，同时恢复阶段也需要将备份的数据再次拷贝，恢复的时间比较久。

20 根据本公开的实施例不需要真实的拷贝数据，只需要对数据增加一个引用，就可以进行备份，极大节省了存储空间，并且在数据库的恢复过程中，通过同样的引用技术，可以通过将计算节点与引用相关联，很快地把备份加载到计算节点组中，从而大大加速了数据库恢复的过程。

25 根据本公开的实施例，还提供了一种用于分布式数据库架构系统的元数据节点，分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点，其中元数据节点被配置为：记录每个数据流与存储每个数据流的一个或多个数据块资源之间的对应关系，一个或多个数据块资源位于多个存储节点的一个或多个中，并且一个或多个数据块资源中的每个存储每个数据流所划分的一个或多个数据块之一；对要被两个以上的计算节点读取的第一数据流记

录两个以上的引用，每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。

上述关于分布式数据库架构系统的描述中涉及到元数据节点的描述同样适用于这里，不再赘述。

- 5 图 9 示出了根据本公开的实施例的用于分布式数据库架构系统的方法 900 的流程图。分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点。如图 9 所示，方法 900 包括步骤 910-步骤 930。

10 在步骤 910 中，由元数据节点记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系，针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中；以及对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。

15 在步骤 920 中，由多个计算节点根据针对每个数据流的对应关系从多个存储节点读取每个数据流或者向多个存储节点存储每个数据流。

 在步骤 930 中，由多个存储节点将每个数据流的一个或多个数据块分别存储到针对一个或多个数据块的存储位置中。

20 在一些实施例中，方法 900 还包括：由元数据节点响应于两个以上的计算节点中的第一计算节点需要读取第一数据流，生成用于第一计算节点的、对第一数据流的引用；响应于第一计算节点不再需要读取第一数据流，删除用于第一计算节点的、对第一数据流的引用；以及响应于对第一数据流的所有引用都被删除，指示多个存储节点删除第一数据流。

25 在一些实施例中，方法 900 还包括：由元数据节点响应于多个计算节点中的第二计算节点期望存储第一数据流的存储请求，为第一数据流所划分成的一个或多个第一数据块分配位于多个存储节点中的一个或多个第一存储节点中的一个或多个第一存储位置，向第二计算节点返回用于存储一个或多个第一数据块的一个或多个第一存储位置的信息，以及记录一个或多个第一数据块中的每个第一数据块与用于存储每个第一数据块的第一存储位置之间的对应关系；由第二计算节点将第一数据流划分为一个或多个第一数据块，以及

根据一个或多个第一存储位置的信息向每个第一存储位置对应的第一存储节点发送一个或多个第一数据块中的对应第一数据块；以及由第一存储节点将从第二计算节点接收到的第一数据块存储到对应的第一存储位置中。

- 5 在一些实施例中，方法 900 还包括：由元数据节点记录新增加的存储节点的信息，并响应于多个计算节点中的第三计算节点期望存储数据流的请求，向第三计算节点分配包括位于新增加的存储节点中的存储位置的存储位置。

在一些实施例中，分布式数据库架构系统还包括控制节点，其中方法 900 还包括：由控制节点根据新增加的存储节点向元数据节点发送新增加的存储节点的信息。

- 10 在一些实施例中，分布式数据库架构系统还包括控制节点，其中，方法 900 还包括：由控制节点：根据多个计算节点的负载分布，将多个计算节点中的第四计算节点处理的第二数据流调度到多个计算节点中的第五计算节点处理，以及控制元数据节点在调度期间建立用于第五计算节点的、对第二数据流的引用，并且在调度完成后删除用于第四计算节点的、对第二数据流的引用。

- 15 在一些实施例中，分布式数据库架构系统还包括控制节点，其中方法 900 还包括：由控制节点控制元数据节点将用于处理第一数据流集合的第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用复制以创建用于新增的或者原有的第二计算节点组中的每个计算节点的、对第一数据流集合中的数据流的引用。

- 20 在一些实施例中，方法 900 还包括：由控制节点控制第一数据流集合从第一计算节点组下线；以及由元数据节点删除用于第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用。

- 25 在一些实施例中，方法 900 还包括：由控制节点控制第一计算节点组中的每个计算节点将内存中的数据同步到第二计算节点组中的计算节点的内存中。

在一些实施例中，方法 900 还包括：由控制点：在第一计算节点组的内存中的数据与第二计算节点组的内存中的数据之间的差少于第一预定阈值或第一计算节点组与第二计算节点组之间的同步延迟低于第二预定阈值时，控制第一计算节点组停止向其内存中写入数据，在同步完成后，控制第二计算节点

组上线，并控制第一计算节点组下线，以及在同步完成后，控制元数据节点删除用于第一计算节点组的每个计算节点的、对第一数据流集合中的数据流的引用。

5 在一些实施例中，方法 900 还包括：由元数据节点对每个引用存储多个副本。

在一些实施例中，多个计算节点部署于一个或多个云服务中；和/或多个存储节点部署于一个或多个云服务中。

上文关于分布式数据库架构系统的说明同样适用于用于分布式数据库架构系统的方法。

10 图 10 示出了用于分布式数据库架构系统的元数据节点的方法 1000 的流程图。分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点。如图 10 所示，方法 1000 包括步骤 1010-步骤 1020。

15 在步骤 1010 中，元数据节点记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储每个数据块的存储位置之间的对应关系，针对每个数据流的一个或多个数据块的存储位置位于多个存储节点的一个或多个中。

在步骤 1020，元数据节点记录对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被两个以上的计算节点中的一个计算节点用来获取针对第一数据流的对应关系以读取第一数据流。

20 上文关于分布式数据库架构系统、元数据节点以及用于分布式数据库架构系统的方法的说明同样适用于用于元数据节点的方法。

25 综上，根据本公开的各个实施例，实现了计算节点与存储节点的解绑和灵活调度，快速读取存储的数据流，以及不同的计算节点能够独立地访问同一数据流，进而能够实现对数据库的灵活调配和快速升级。根据本公开的各个实施例，还可以解决存储节点的快速扩容、计算节点不拷贝数据地动态负载均衡、数据库的资源共享和独享、数据库粒度的快速零拷贝多副本、不拷贝数据地快速克隆可读可写的数据库、数据库克隆的快速平滑升级、数据库粒度的快速备份和恢复等等。

需要说明的是，上述的具体实施例仅是例子而非限制，且本领域技术人员

可以根据本公开的构思从上述分开描述的各个实施例中合并和组合一些步骤和装置来实现本公开的效果，这种合并和组合而成的实施例也被包括在本公开中，在此不一一描述这种合并和组合。此外，本公开中的“第一”、“第二”、

“第三”、“第四”仅是为了标记某个或某些元素，并不表示它们的重要性或特殊性，也不表示它们的不同。例如，第一计算节点和第二计算节点可以是不同的计算节点也可以是相同的计算节点。

在本公开中提及的优点、优势、效果等仅是示例而非限制，不能认为这些优点、优势、效果等是本公开的各个实施例必须具备的。另外，上述公开的具体细节仅是为了示例的作用和便于理解的作用，而非限制，上述细节并不限制本公开为必须采用上述具体的细节来实现。

本公开中涉及的器件、装置、设备、系统的方框图仅作为例示性的例子并且不意图要求或暗示必须按照方框图示出的方式进行连接、布置、配置。如本领域技术人员将认识到的，可以按任意方式连接、布置、配置这些器件、装置、设备、系统。

本公开中的步骤流程图以及以上方法描述仅作为例示性的例子并且不意图要求或暗示必须按照给出的顺序进行各个实施例的步骤。如本领域技术人员将认识到的，可以按任意顺序进行以上实施例中的步骤的顺序。诸如“其后”、“然后”、“接下来”等等的词语不意图限制步骤的顺序；这些词语仅用于引导读者通读这些方法的描述。

以上描述的方法的各个操作可以通过能够进行相应的功能的任何适当的手段而进行。该手段可以包括各种硬件和/或软件组件和/或模块，包括但不限于硬件的电路、专用集成电路（ASIC）或处理器。

本公开还可以包括计算机程序产品，其中计算机程序产品可以进行在此给出的方法、步骤和操作。例如，这样的计算机程序产品可以是计算机软件包、计算机代码指令、具有有形存储（和/或编码）在其上的计算机指令的计算机可读的有形介质，该指令可由处理器执行以进行在此描述的操作。

权利要求书

1. 一种分布式数据库架构系统，包括多个计算节点、多个存储节点和元数据节点，其中
- 5 所述元数据节点被配置为：
- 记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储所述每个数据块的存储位置之间的对应关系，针对每个数据流的所述一个或多个数据块的存储位置位于所述多个存储节点的一个或多个中，以及对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，
- 10 每个引用被所述两个以上的计算节点中的一个计算节点用来获取针对所述第一数据流的所述对应关系以读取所述第一数据流；
- 所述多个计算节点被配置为：
- 根据针对每个数据流的所述对应关系从所述多个存储节点读取所述每个数据流或者向所述多个存储节点存储所述每个数据流；以及
- 15 所述多个存储节点被配置为：
- 将每个数据流的所述一个或多个数据块分别存储到针对所述一个或多个数据块的存储位置中。
2. 根据权利要求 1 所述的分布式数据库架构系统，其中，所述元数据节点还
- 20 还被配置为：
- 响应于所述两个以上的计算节点中的第一计算节点需要读取所述第一数据流，生成用于所述第一计算节点的、对所述第一数据流的引用；
- 响应于所述第一计算节点不再需要读取所述第一数据流，删除用于所述第一计算节点的、对所述第一数据流的引用；以及
- 25 响应于对所述第一数据流的所有引用都被删除，指示所述多个存储节点删除所述第一数据流。
3. 根据权利要求 1 所述的分布式数据库架构系统，其中
- 所述元数据节点还被配置为：

响应于所述多个计算节点中的第二计算节点期望存储所述第一数据流的存储请求，为所述第一数据流所划分成的一个或多个第一数据块分配位于所述多个存储节点中的一个或多个第一存储节点中的一个或多个第一存储位置，

- 5 向所述第二计算节点返回用于存储所述一个或多个第一数据块的所述一个或多个第一存储位置的信息，以及

记录所述一个或多个第一数据块中的每个第一数据块与用于存储所述每个第一数据块的第一存储位置之间的对应关系；

所述第二计算节点被配置为：

- 10 将所述第一数据流划分为所述一个或多个第一数据块，以及
根据所述一个或多个第一存储位置的信息向每个第一存储位置对应的第一存储节点发送所述一个或多个第一数据块中的对应第一数据块；以及
所述第一存储节点被配置为：

- 15 将从所述第二计算节点接收到的第一数据块存储到对应的第一存储位置中。

4. 根据权利要求1所述的分布式数据库架构系统，其中

- 20 所述元数据节点还被配置为：记录新增加的存储节点的信息，并响应于所述多个计算节点中的第三计算节点期望存储数据流的请求，向所述第三计算节点分配包括位于所述新增加的存储节点中的存储位置的存储位置。

5. 根据权利要求4所述的分布式数据库架构系统，还包括控制节点，其中

- 25 所述控制节点被配置为：根据所述新增加的存储节点向所述元数据节点发送所述新增加的存储节点的信息。

6. 根据权利要求1所述的分布式数据库架构系统，还包括控制节点，其中，

所述控制节点被配置为：

根据所述多个计算节点的负载分布，将所述多个计算节点中的第四计算节点处理的第二数据流调度到所述多个计算节点中的第五计算节点处理，以及

- 5 控制所述元数据节点在所述调度期间建立用于所述第五计算节点的、对所述第二数据流的引用，并且在所述调度完成后删除用于所述第四计算节点的、对所述第二数据流的引用。

7. 根据权利要求 1 所述的分布式数据库架构系统，还包括控制节点，其中
- 10 所述控制节点被配置为：

控制所述元数据节点将用于处理第一数据流集合的第一计算节点组的每个计算节点的、对所述第一数据流集合中的数据流的引用复制以创建用于新增的或者原有的第三计算节点组中的每个计算节点的、对所述第一数据流集合中的数据流的引用。

- 15
8. 根据权利要求 7 所述的分布式数据库架构系统，其中
- 所述控制节点还被配置为：控制所述第一数据流集合从所述第一计算节点组下线；以及

- 20 所述元数据节点还被配置为：删除用于所述第一计算节点组的每个计算节点的、对所述第一数据流集合中的数据流的引用。

9. 根据权利要求 7 所述的分布式数据库架构系统，其中
- 所述控制节点还被配置为：控制所述第一计算节点组中的每个计算节点将内存中的数据同步到第二计算节点组中的计算节点的内存中。

- 25
10. 根据权利要求 9 所述的分布式数据库架构系统，其中
- 所述控制节点还被配置为：

在所述第一计算节点组的内存中的数据与所述第二计算节点组的内存中的数据之间的差少于第一预定阈值或所述第一计算节点组与所述第二计

算节点组之间的同步延迟低于第二预定阈值时，控制所述第一计算节点组停止向其内存中写入数据，

在所述同步完成后，控制所述第二计算节点组上线，并控制所述第一计算节点组下线，以及

- 5 在所述同步完成后，控制所述元数据节点删除用于所述第一计算节点组的每个计算节点的、对所述第一数据流集合中的数据流的引用。

11. 根据权利要求 1 所述的分布式数据库架构系统，其中
所述元数据节点还被配置为：对每个引用存储多个副本。

10

12. 根据权利要求 1 所述的分布式数据库架构系统，其中
所述多个计算节点部署于一个或多个云服务中；和/或
所述多个存储节点部署于一个或多个云服务中。

- 15 13. 一种用于分布式数据库架构系统的元数据节点，所述分布式数据库架构系统包括多个计算节点、多个存储节点和所述元数据节点，其中所述元数据节点被配置为：

记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储所述每个数据块的存储位置之间的对应关系，针对每个数据流的所述一个
20 或多个数据块的存储位置位于所述多个存储节点的一个或多个中；以及

对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被所述两个以上的计算节点中的一个计算节点用来获取针对所述第一数据流的所述对应关系以读取所述第一数据流。

- 25 14. 一种用于分布式数据库架构系统的方法，所述分布式数据库架构系统包括多个计算节点、多个存储节点和元数据节点，其中，所述方法包括：

由所述元数据节点：

记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储所述每个数据块的存储位置之间的对应关系，针对每个数据流的所述

一个或多个数据块的存储位置位于所述多个存储节点的一个或多个中，以及
对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，
每个引用被所述两个以上的计算节点中的一个计算节点用来获取针对所述第一数据流的所述对应关系以读取所述第一数据流；

5 由所述多个计算节点：

根据针对每个数据流的所述对应关系从所述多个存储节点读取所述
每个数据流或者向所述多个存储节点存储所述每个数据流；以及

由所述多个存储节点：

10 将每个数据流的所述一个或多个数据块分别存储到针对所述一个或
多个数据块的存储位置中。

15. 一种用于分布式数据库架构系统的元数据节点的方法，所述分布式数据库架构系统包括多个计算节点、多个存储节点和所述元数据节点，其中所述方法包括：

15 记录每个数据流所划分成的一个或多个数据块中的每个数据块与用于存储所述每个数据块的存储位置之间的对应关系，针对每个数据流的所述一个或多个数据块的存储位置位于所述多个存储节点的一个或多个中；以及

20 对要被两个以上的计算节点读取的第一数据流记录两个以上的引用，每个引用被所述两个以上的计算节点中的一个计算节点用来获取针对所述第一数据流的所述对应关系以读取所述第一数据流。