

内存缓存池维护方法、设备、系统及存储介质

本申请要求于 2023 年 03 月 27 日提交中国专利局、申请号为 202310313455.8、发明名称为“内存缓存池维护方法、设备、系统及存储介质”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及云计算技术领域，尤其涉及一种内存缓存池维护方法、设备、系统及存储介质。

背景技术

随着云原生（cloud-native）技术的发展，出现了基于云原生的数据库类应用。基于云原生的数据库类应用利用云上的软硬件资源，能够实现存储资源和计算资源的池化和按需分配，具有更高的资源利用率和更高的可用性。但是，不同数据库类应用对存储资源和计算资源的需求会有所不同，因此，根据数据库类应用的负载变化情况动态按需对计算资源和内存资源进行扩缩容，对于获得稳定的高吞吐率和低延迟，同时提高资源利用率，节约资源成本具有重要意义。

为此，基于云原生的数据库类应用采用了存算分离的架构，在存算分离架构中，计算节点主要提供计算资源，远端内存节点作为计算节点上内存缓存池（memory buffer pool）资源的扩展，负责提供远程内存缓存池资源，持久存储节点提供持久型存储资源，构成三个独立的资源池，计算节点上的计算资源和远程内存节点提供的内存缓存池资源分离，每个资源池的扩缩容是独立进行的，有利于实现更精细化的资源按需请求和弹性。

在存算分离架构中，一方面依赖持久型存储资源来提供可靠性和容错机制，一方面采用内存缓存池缓存持久型存储中的数据用来提供快速响应能力。而且，远程内存缓存池中的数据可被多个计算节点共享，这有利于降低存储成本，但也会带来缓存数据一致性问题。缓存数据一致性的维护和延迟直接影响数据库类应用的吞吐率和响应延迟，对应用性能和可扩展性至关重要。如何在维护缓存数据一致性的过程中，降低数据传输的延迟是存算分离架构面临的主要技术问题。

发明内容

本申请的多个方面提供一种内存缓存池维护方法、设备、系统及存储介质，用以在维护缓存数据一致性的过程中降低数据传输延迟。

本申请实施例提供一种计算节点，包括：第一计算单元和第一处理单元，所述第一计算单元与所述第一处理单元互联；所述第一计算单元，用于运行上层应用，并为所述上层应用提供用于缓存页面数据的本地内存缓存池，所述本地内存缓存池能够被所述第一处理单元直接访问；所述第一处理单元，还与至少一个内存节点中的第二处理单元网络互联，用于与所述第二处理单元进行数据交互，以在本地内存缓存池与所述至少一个内存节点为所述上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步。

本申请实施例提供一种内存节点，包括：第二计算单元和第二处理单元，所述第二计算单元与所述第二处理单元互联；所述第二计算单元，用于为至少一个计算节点上运行的上层应用提供共享内存缓存池，所述共享内存缓存池能够被所述第二处理单元直接访问；所述第二处理单元，还与至少一个计算节点中的第一处理单元网络互联，用于与所述第一处理单元进行数据交互，以在共享内存缓存池与所述至少一个计算节点为所述上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步。

5 本申请实施例还提供一种内存缓存池维护方法，应用于计算节点，所述计算节点包括第一计算单元和第一处理单元，所述第一处理单元与所述第一计算单元互联，以及与至少一个内存节点中的第二处理单元网络互联，所述方法包括：所述第一处理单元与所述至少一个内存节点中的第二处理单元进行数据交互，以在所述第一计算单元为上层应用提供的本地内存缓存池与所述至少一个内存节点为所述上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步；其中，所述本地内存缓存池能够被所述第一处理单元直接访问。

10 本申请实施例还提供一种内存缓存池维护方法，应用于内存节点，所述内存节点包括第二计算单元和第二处理单元，所述第二处理单元与所述第二计算单元互联，以及与至少一个计算节点中的第一处理单元网络互联，所述方法包括：所述第二处理单元与所述至少一个计算节点中的第一处理单元进行数据交互，以在所述内存节点为所述上层应用提供的共享内存缓存池与所述至少一个计算节点为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步；其中，所述共享内存缓存池能够被所述第二处理单元直接访问。

15 本申请实施例还提供一种存算分离系统，包括：上述多个计算节点、上述至少一个内存节点以及至少一个存储节点；其中，所述至少一个存储节点，用于持久化存储所述共享内存缓存池和/或本地内存缓存池中的页面数据。

20 本申请实施例还提供一种处理单元，所述处理单元包括：处理模块、网卡模块和存储模块，所述存储模块中存储有计算机程序，所述处理模块执行所述存储模块中存储的计算机程序，以用于实现内存缓存池维护方法中的步骤。

本申请实施例还提供一种存储有计算机程序的计算机可读存储介质，当所述计算机程序被处理器执行时，致使所述处理器能够实现内存缓存池维护方法中的步骤。

附图说明

25 在附图中，除非另外规定，否则贯穿多个附图相同的附图标记表示相同或相似的部件或元素。这些附图不一定是按照比例绘制的。应该理解，这些附图仅描绘了根据本申请公开的一些实施方式，而不应将其视为是对本申请范围的限制。

30 此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中包括如下图 1-图 10。

- 图 1 示出了本申请实施例提供的一种存算分离系统的结构示意图。
- 图 2 示出了本申请实施例提供的另一种存算分离系统的结构示意图。
- 图 3 示出了本申请实施例提供的示例性的缓存池一致性维护的原理图。
- 图 4 示出了本申请实施例提供的示例性的集成 DSA 的 CIPU 从节点上页状态更新流程图。
- 图 5 示出了本申请实施例提供的示例性的缓存池一致性维护逻辑原理图。
- 图 6 示出了在一些情形中的双写流程与本申请实施例提供的示例性的双写流程的对比图。
- 图 7 示出了本申请实施例提供的示例性的一种内存缓存池维护方法的信令交互图。
- 图 8 示出了本申请实施例提供的另一种内存缓存池维护方法的信令交互图。
- 图 9 示出了本申请实施例提供的另一种内存缓存池维护方法的信令交互图。
- 图 10 示出了本申请实施例提供的一种处理单元的结构示意图。

具体实施方式

为使本申请的目的、技术方案和优点更加清楚，下面将结合本申请具体实施例及相应的附图对本申请技术方案进行清楚、完整地描述。显然，所描述的实施例仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

5 需要说明的是，本申请所涉及的用户信息（包括但不限于用户设备信息、用户个人信息等）和数据（包括但不限于用于分析的数据、存储的数据、展示的数据等），均为经用户授权或者经过各方充分授权的信息和数据，并且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准，并提供有相应的操作入口，供用户选择授权或者拒绝。

10 针对现有存算分离架构所面临的如何在维护缓存数据一致性过程中降低数据传输延迟的技术问题，在本申请实施例中，针对存算分离架构，在计算节点和内存节点中增设与计算单元互联的处理单元，将内存缓存数据的一致性维护任务下推到处理单元上执行，而不是在计算单元上完成，由于处理单元具有对本地内存缓存池或共享内存缓存池直接访问的能力，使得内存缓存数据一致性维护过程中的数据传输过程大都在处理单元内部完成，
15 无需上传至计算单元，能够消除计算单元和网卡之间进行数据传输的延迟，达到在维护缓存数据一致性的过程中降低数据传输延迟的目的。

以下结合附图，详细说明本申请各实施例提供的技术方案。

图1为本申请实施例提供的一种存算分离系统的结构示意图。如图1所示，该系统包括：多个计算节点10、至少一个内存节点20以及至少一个存储节点30。

20 在本实施例中，并不限定计算节点10、内存节点20以及存储节点30的设备实现形态。例如，计算节点10可以是任何具有计算能力的设备形态，例如可以是但不限于：智能手机、笔记本电脑、台式电脑、穿戴设备、传统服务器、云服务器、服务器集群等。内存节点20可以是任何能够提供内存资源的设备形态，例如可以是但不限于：计算机、传统服务器、云服务器等。相应地，存储节点30可以是任何能够提供持久化存储资源的设备形态，
25 例如各种存储设备。

在本实施例中，计算节点10主要提供计算能力，用于运行上层应用，并为上层应用提供相对有限的本地内存缓存池，以容纳上层应用需要高频访问的各种数据。在本实施例中，并不限定上层应用的类型，可以是各种具有内存和持久化存储需求的应用程序，例如可以是各种云计算服务、数据库实例、流计算服务、视频处理服务等，当然不同上层应用
30 产生的数据也会有所不同。在本实施例中，存储节点30负责为上层应用提供持久化和可靠的共享存储资源，例如以上层应用是数据库实例为例，可以存储数据库实例中的预写日志（Write-Ahead Logging, WAL）数据、数据库表和检查点等。内存节点20作为本地内存缓存池和持久共享存储中的中间层，主要用于为上层应用提供共享内存缓存池，共享内存缓存池是对本地内存缓存池的拓展。在本实施例中，在本地内存缓存池和共享内存缓存池中以内存页面（page）方式进行管理，并将这些内存缓存池中的数据称为页面数据。
35

如图1所示，计算节点10、内存节点20以及存储节点30之间通过网络互联，网络互联方式包括但不限于：以太网、远程直接数据存取（Remote Direct Memory Access, RDMA）等。其中，多个计算节点10可分布式部署在不同位置，相应地，在内存节点20为多个的情况下，也可以分布式部署在不同位置，以及在存储节点30为多个的情况下，也可以分
40 布式部署在不同位置。例如，在图1中，多个计算节点10形成计算集群，至少一个内存节点20形成内存存储集群，至少一个存储节点30可形成持久化存储集群。需要说明的是，以集群形态对各类节点进行管理仅为示例，并不限于此。

在本实施例中，计算节点10包括第一计算单元11和具有网络通信能力和计算能力的第一处理单元12，第一计算单元11和第一处理单元12互联；内存节点20包括第二计算

单元 21 和具有网络通信能力和计算能力的第二处理单元 22, 第二计算单元 21 和第二处理单元 22 互联。第一计算单元 11 和第二计算单元 21 可以是任何具有计算能力的器件, 例如包括但不限于: 中央处理器 (Central Processing Unit, CPU)、数据处理器 (Data Processing Unit, DPU) 等芯片或模组。第一处理单元 12 和第二处理单元 22 可以是任何具有网络通信能力和计算能力的器件, 例如可以是一些集成有网卡模块的云基础设施单元 (Cloud Infrastructure Processing Unit, CIPU), 智能网卡 (Smart NIC), 网络接口卡 (Network Interface Card, NIC)、DPU、基础设施处理器 (Infrastructure Processing Unit, IPU) 等芯片或模组。在图 1 中, 以第一计算单元 11 和第二计算单元 21 是 CPU, 第一处理单元 12 和第二处理单元 22 是 CIPU 为例进行图示, 但并不限于此。

在本实施例中, 第一计算单元 11 和第二计算单元 21 的功能类似, 主要为其所属节点提供计算能力, 但会因为计算节点 10 和内存节点 20 的功能不同, 在计算能力和内存 (Memory) 的配置上有所不同。同理, 第一处理单元 12 和第二处理单元 22 的功能类似, 主要为其所属节点提供内存缓存一致性维护服务, 但会因为计算节点 10 和内存节点 20 的功能不同, 在计算能力、网络能力和内存的配置上有所不同。

在本实施例中, 如图 2 所示, 第一计算单元 11 负责运行其所属计算节点 10 上的上层应用, 并为上层应用提供用于缓存页面数据的本地内存缓存池, 并负责对本地内存缓存池的管理, 例如负责本地内存缓存池的分配、初始化和配置等操作。在一实施方式中, 在上层应用运行过程中, 第一计算单元 11 还可以接收用户请求并对用户请求进行响应, 根据上层应用的不同, 用户请求以及对用户请求进行响应的方式均会有所不同, 本申请实施例对此不做关注。上层应用可以是任何运行在计算节点 10 上, 且需要通过本地内存缓存池和由内存节点 20 提供的共享内存缓存池进行数据缓存的应用程序, 例如可以是但不限于数据库实例。在计算节点 10 中, 第一计算单元 11 可以运行一个或多个上层应用, 并且会为每个上层应用提供对应的本地内存缓存池。相应地, 如图 2 所示, 内存节点 20 中的第二计算单元 21 负责为至少一个计算节点 10 上运行的上层应用提供共享内存缓存池, 并负责对共享内存缓存池的管理, 例如负责共享内存缓存池的分配、初始化和配置等操作。需要说明的是, 内存节点 20 与计算节点 10 之间的对应关系可以是一对一、多对一或一对多, 也就是说, 一个内存节点 20 可以为一个或多个计算节点 10 上的上层应用提供共享内存缓存池, 一个计算节点 10 可以使用一个或多个内存节点 20 为其提供共享内存缓存池。对计算节点 10 和内存节点 20 来说, 可以预先维护彼此之间的对应关系。

在一实施方式中, 如图 2 所示, 第一计算单元 11 还维护有本地内存缓存池的元数据, 这些元数据是对本地内存缓存池进行管理所需的数据, 包括但不限于: 空闲列表、各种链表, 页面状态信息 (如已失效 (Invalidation)、独占 (unique)、共享 (share) 等)、以及地址映射表等。相应地, 如图 2 所示, 第二计算单元 21 上维护有共享内存缓存池的元数据, 这些元数据是对共享内存缓存池进行管理所需的数据, 包括但不限于: 空闲列表、各种链表, 页面状态信息、以及地址映射表等。其中, 无论是本地内存缓存池的元数据, 还是共享内存缓存池的元数据, 这些元数据中的各种链表包括但不限于: 页面访问信息维护链表, 如最近最少使用 (Least Recently Used, LRU) 链表, 用于存储脏数据的链表、空闲 (free) 链表等。地址映射表用于存储页面数据的物理地址与逻辑地址之间的映射关系。随着对本地内存缓存池或共享内存缓存池的访问, 相应元数据也会被同步更新。

在本实施例中, 在计算节点 10 中增设第一处理单元 12, 第一处理单元 12 通过互联通道与第一计算单元 11 互联, 基于该互联通道可以直接访问本地内存缓存池及其元数据, 而无需经过第一计算单元 11。例如, 第一处理单元 12 可以直接从本地内存缓存池中读取页面数据, 或者直接向本地内存缓存池中写入页面数据; 也可以直接从本地内存缓存池的元数据中读取与内存缓存数据一致性维护过程相关的元数据, 或者, 对其中与内存缓存数据

一致性维护过程相关的元数据进行修改。相应地，在内存节点 20 上增设第二处理单元 22，第二处理单元 22 通过互联通道与第二计算单元 21 互联，基于该互联通道可以直接访问共享内存缓存池及其元数据，而无需经过第二计算单元 21。例如，第二处理单元 22 可以直接从其所属内存节点 20 中的共享内存缓存池中读取页面数据，或者直接向共享内存缓存池中写入页面数据；也可以直接从共享内存缓存池的元数据中读取与内存缓存数据一致性维护过程相关的元数据，或者，对其中与内存缓存数据一致性维护过程相关的元数据进行修改。

在本实施例中，并不限定互联通道的实现方式，例如可以是但不限于：外围组件快速互连标准（Peripheral Component Interconnect express, PCIe）通道，或者缓存一致性通道。在一实施方式中，第一处理单元 12 与第一计算单元 11 之间，以及第二处理单元 22 与第二计算单元 21 之间采用缓存一致性通道进行互联。本申请实施例可以采用一些已有的缓存（cache）一致性通道，例如超级通道互联（Ultra Path Interconnect, UPI）通道、计算标准连接（Compute Express Link, CXL）通道或加速器缓存一致性互连（Cache Coherent Interconnect for Accelerators, CCIX）通道，也可以采用自研的缓存一致性通道，对此不做限定。

随着上层应用的运行，其所产生的数据会越来越多，在本地内存缓存池不足的情况下，可通过页面替换机制将旧的页面数据换出本地内存缓存池，被换出的页面数据可以被缓存至一个或多个内存节点 20 所提供的共享内存缓存池中。其中，页面替换机制可以由第一计算单元 11 执行，也可以由第一处理单元 12 执行，对此不做限定。在页面替换机制由第一计算单元 11 执行的情况下，第一计算单元 11 可以根据页面替换机制的相关要求监测本地内存缓存池中页面数据的使用情况或属性信息，发现需要从本地内存缓存池中替换出去的页面数据，然后通知第一处理单元 12，由第一处理单元 12 从本地内存缓存池中直接读取需要替换出去的页面数据并将该页面数据提供给相应内存节点 20 中的第二处理单元 22，由内存节点 20 中的第二处理单元 22 将该页面数据写入内存节点 20 的共享内存缓存池中。在一实施方式中，基于第一处理单元 12 对本地内存缓存池的直接访问能力，该页面替换机制可以由第一处理单元 12 执行，例如，第一处理单元 12 可以根据页面替换机制的相关要求监测本地内存缓存池中页面数据的使用情况或属性信息，发现需要从本地内存缓存池中替换出去的页面数据，然后从本地内存缓存池中直接读取需要替换出去的页面数据并将该页面数据提供给相应内存节点 20 中的第二处理单元 22，由内存节点 20 中的第二处理单元 22 将该页面数据写入内存节点 20 的共享内存缓存池中。关于页面替换机制的具体实现方式，本申请实施例也不做限定。例如，可以是传统的最近最少使用（Least Recently Used, LRU）方式，也可以结合上层应用的一些信息灵活定义所使用的页面替换机制，使得第一计算单元 11 或第一处理单元 12 可以基于页面数据的一些属性信息，例如页面类型等进行页面替换。

另外，在存算分离架构中，同一上层应用可能需要同时运行在多个不同的计算节点 10 上，或者需要多个计算节点 10 之间相互协作，这意味着多个计算节点 10 之间需要共享相同的页面数据，这就涉及共享内存缓存池与本地内存缓存池之间数据一致性的问题，确保各个计算节点 10 都能够使用正确的页面数据。

在本实施例中，对任一计算节点 10 来说，其第一处理单元 12 具有网络通信能力和计算能力，与至少一个内存节点 20 中的第二处理单元 22 网络互联，能够与至少一个内存节点 20 中的第二处理单元 22 配合，这里的配合主要是指第一处理单元 12 与第二处理单元 22 进行数据交互，以在本地内存缓存池与至少一个内存节点 20 为上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步，进而实现内存缓存数据一致性。相应地，对任一内存节点 20 来说，其第二处理单元 22 具有网络通信能力和计算能力，与至少一个

计算节点 10 中的第一处理单元 12 网络互联, 能够与第一处理单元 12 进行数据交互, 以在共享内存缓存池与至少一个计算节点 10 为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步, 进而实现数据一致性。

5 在一些实施方式中, 如图 2 所示, 第一处理单元 12 上可以运行本地缓存池管理器实现内存缓存一致性维护功能, 在一实施例中, 本地缓存池管理器负责本地内存缓存池的管理, 在本地内存缓存池需要刷新脏页面数据或者读取新页面数据等操作时, 执行到共享内存缓存池的快速读写, 以及在读写过程中进行页面状态的同步。相应地, 如图 2 所示, 第二处理单元 22 上可以运行远端缓存池管理器实现内存缓存一致性维护功能, 例如, 远端缓存池管理器一方面负责接收本地缓存池管理器的检查点、刷新脏页面数据、读取新页面数据等请求并基于对共享内存缓存池的访问对这些请求进行响应, 以及在读写过程中进行页面状态的同步。例如, 本第缓存池管理器还负责在计算节点发生故障时, 利用远端的共享内存缓存池中的页面数据和检查点进行计算节点的快速故障恢复; 相应地, 远端缓存池管理器还负责在计算节点发生故障时, 从持久共享的存储节点 30 中读取检查点、WAL 日志等为计算节点进行故障恢复提供数据支持。

15 在本实施例中, 第一处理单元 12 是硬件结构, 其通过互联通道 (如缓存一致性通道) 能够直接访问其所在计算节点 10 上的本地内存缓存池, 本地内存缓存池中存储的是上层应用产生的用户态数据, 也就是说, 第一处理单元 12 能够共享上层应用产生的用户态数据, 而不是内核态数据, 且对用户态数据的共享是不需要经过计算节点 10 的操作系统 (Operating System, OS) 的。由此, 在计算节点 10 中增设第一处理单元 12, 使得内存缓存数据一致性维护任务能够在应用层实现, 旁路操作系统的协议栈, 在缓存数据一致性维护过程中能够充分利用上层应用中的数据访问特点, 提高数据一致性维护的传输性能。例如, 在页面替换机制中, 可以考虑更多应用层的数据访问特点, 而不是单纯的依据页面被访问的时间和次数等信息, 例如根据页面数据的类型可以确定一些访问率较低的页面数据, 对这些页面数据可以优先替换出本地内存缓存池中, 而不依赖于这些页面数据的访问时间

20 和次数等信息。

25 接下来, 重点针对第一处理单元 12 与第二处理单元 22 相互配合, 在本地内存缓存池与共享内存缓存池之间进行页面数据和/或页面状态同步的过程进行详细说明。其中, 第一处理单元 12 与第二处理单元 22 相互配合, 在本地内存缓存池与共享内存缓存池之间进行页面数据和/或页面状态同步的情况可以概括为以下几种情况: 情况 1: 向共享内存缓存池中写入页面数据; 情况 2: 从共享内存缓存池中读取页面数据; 情况 3: 对页面数据加读锁处理; 情况 4: 对页面数据加写锁处理; 情况 5: 对页面数据进行作废处理。

30 下面从任一计算节点与任一内存节点之间进行数据一致性维护的角度, 对上述情况 1-5 进行详细说明。

对于情况 1: 第一计算节点的第一处理单元 12 从本地内存缓存池中读取第一页面数据, 并将第一页面数据发送给第一内存节点中的第二处理单元 22; 第一内存节点中的第二处理单元 22 将第一页面数据写入第一内存节点中的共享内存缓存池中。

对于情况 2: 第二内存节点中的第二处理单元 22 从第二内存节点的共享内存缓存池中读取第二页面数据并发送给第一计算节点的第一处理单元 12; 第一计算节点的第一处理单元 12 接收第二内存节点中的第二处理单元 22 发送的第二页面数据, 将第二页面数据写入本地内存缓存池中。

40 对于情况 3: 接续于情况 2, 第一计算节点的第一处理单元 12 在接收第二页面数据之前, 向第二内存节点中的第二处理单元 22 发送加读锁的请求, 以使第二内存节点中的第二处理单元 22 为第二页面数据添加读锁。

对于情况 4 和 5: 在第三计算节点中的第一计算单元 12 改写本地内存缓存池中的第三

页面数据之前，向第三内存节点中的第二处理单元 22 发送对第三页面数据进行加写锁和作废的请求；第三内存节点中的第二处理单元 22 对共享内存缓存池中的第三页面数据添加写锁并通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理。

5 在一些实施方式中，在第三计算节点中的第一计算单元 12 改写本地内存缓存池中的第三页面数据之前，向第三内存节点中的第二处理单元 22 发送对第三页面数据进行加写锁的请求；第三内存节点中的第二处理单元 22 对共享内存缓存池中的第三页面数据添加写锁，并返回加写锁的确认信息；第三计算节点中的第一计算单元 12 向第三内存节点中的第二处理单元 22 发送对第三页面数据进行作废的请求；第三内存节点中的第二处理单元 22 通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理，并在收到其它计算节点返回的确认信息后向第三内存节点中的第二处理单元 22 返回作废的确认信息。

在上文中，第一计算节点、第二计算节点和第三计算节点是任一计算节点，可以是同一计算节点，也可以是不同的计算节点；相应地，第一内存节点、第二内存节点和第三内存节点是任一内存节点，且可以是同一内存节点，也可以是不同的内存节点。

15 在本实施例中，并未限定第一处理单元和第二处理单元的内部实现形态。在一实施例中，如图 2 所示，第一处理单元 12 包括第一处理模块 121 和第一网卡模块 122；相应地，第二处理单元 22 包括第二处理模块 221 和第二网卡模块 222。在一实施方式中，第一处理模块 121 和第二处理模块 221 可以是任何具有计算能力的器件，例如领域专用体系结构 (Domain Specific Architecture, DSA) 模块或芯片。第一网卡模块 122 和第二网卡模块 222 可以是 NIC，但并不限于此。

下面结合图 2 所示第一处理单元 12 和第二处理单元 22 的详细实现结构，对上述情况 1-5 的过程做进一步详细说明。

25 针对向共享内存缓存池中写入页面数据的情况 1，第一计算节点中的第一处理模块从本地内存缓存池中读取第一页面数据，经第一计算节点中的第一网卡模块和第一内存节点中的第二网卡模块将第一页面数据传输给第一内存节点中的第二处理模块；第一内存节点中的第二处理模块将第一页面数据写入第一内存节点中的共享内存缓存池，实现从本地内存缓存池向共享内存缓存池写入页面数据。

30 示例性的，为了提高向共享内存缓存池中写入页面数据的写入性能，第一计算节点中的第一处理模块，用于：从本地内存缓存池中读取第一页面数据，并通过第一网卡模块发送给第一内存节点中的第二网卡模块，以供第一内存节点中的第二处理模块将第一页面数据写入第一内存节点中的共享内存缓存池中。

35 相应地，第一内存节点中的第二处理单元中的第二处理模块，用于：通过该第二处理单元中的第二网卡模块接收第一计算节点中的第一网卡模块发送的第一页面数据，将第一页面数据写入第一内存节点中的共享内存缓存池中。

40 在本实施例中，并不限定第一计算节点的第一处理模块在何种条件下从本地内存缓存池中读取第一页面数据。在一实施例中，页面替换机制由第一计算节点中的第一计算单元执行，则第一计算节点中的第一计算单元可以根据页面替换机制确定本地内存缓存池中是否存在需要淘汰且处于已修改状态的页面数据，若存在，将需要淘汰且处于已修改状态的页面数据作为第一页面数据，然后向第一计算节点中的第一处理模块发送脏页面回写指令；相应地，第一计算节点中的第一处理模块具体用于响应第一计算节点中第一计算单元发送的脏页面回写指令，从本地内存缓存池中读取第一页面数据。关于“已修改状态”的描述可参见下文关于页面数据状态示例中的说明，在此暂不详述。或者，在一实施例中，页面替换机制有第一计算节点中的第一处理模块执行，则第一计算节点中的第一处理模块

可以根据页面替换机制确定本地内存缓存池中是否存在需要淘汰且处于已修改状态的第一页面数据,若存在,则从本地内存缓存池中读取所述第一页面数据。在前述示例中,第一页面数据为需要回写的脏页面数据,但需要向共享内存缓存池中写入页面数据的场景并不限于回写脏页面数据这一种场景。

5 针对从共享内存缓存池中读取页面数据的情况 2,第二内存节点中的第二处理模块从共享内存缓存池中读取第二页面数据,经第二内存节点中的第二网卡模块传输给第二计算节点中的第一网卡模块,第二计算节点中的第一网卡模块将第二页面数据提供给第二计算节点中的第一处理模块,第二计算节点中的第一处理模块将第二页面数据写入至第二计算节点中的本地内存缓存池,实现从内存节点侧的共享内存缓存池中读取页面数据。

10 示例性的,为了提高从共享内存缓存池中读取页面数据的读取性能,第二内存节点中的第二处理模块,用于:从共享内存缓存池中读取第二页面数据,并通过第二内存节点中的第二网卡模块将第二页面数据发送给第二计算节点中的第一网卡模块,以供第二计算节点中的第一处理模块将第二页面数据写入第二计算节点的本地内存缓存池中。相应地,第二计算节点中的第一处理模块,用于:通过第二计算节点中第一网卡模块接收第二内存节点中的第二网卡模块发送的第二页面数据,将第二页面数据写入本地内存缓存池中。

15 在本实施例中,并不限定第二内存节点中的第二处理模块在何种条件下从共享内存缓存池中读取第二页面数据。在一实施例中,第二计算节点中的第一处理模块可响应第一计算节点中的第一计算单元发送的页面读指令,通过第一计算节点的第一网卡模块向第二内存节点中的第二处理模块发送页面读请求,以使第二内存节点中的第二处理模块从第二内存节点的共享内存缓存池中读取第二页面数据并通过第二内存节点中的第二网卡模块和第二计算节点中的第一网卡模块返回给第二计算节点中的第一处理模块,由第二计算节点中的第一处理模块将第二页面数据写入至第二计算节点中的本地内存缓存池。

20 对第二计算节点中的第一计算单元来说,在运行上层应用过程中,可以根据运行到的指令代码,查询本地内存缓存池,以判断从本地内存缓存池中是否存在其所需的页面数据。在本实施例中,对本地内存缓存池以及共享内存缓存池中的页面数据,可以维护多个状态,例如包括但不限于:已修改(Modified)状态、独占(Exclusive)状态、共享(Shared)状态、已失效(Invalidated)状态以及加写锁和加读锁的状态,这些页面数据的状态存储在本地内存缓存池的元数据和共享内存缓存池的元数据中。其中,已修改状态表示页面数据在当前计算节点的本地内存缓存池中被修改过了,但并未同步回共享内存缓存池中;独占状态表示页面数据被当前计算节点独占,其它计算节点的本地内存缓存池中的同一个页面数据处于失效状态;共享状态表示页面数据可被多个计算节点共享且都是有效状态;已失效状态表示页面数据是过时的,无法被使用的状态;加写锁状态与独占状态类似,表示页面数据被当前计算节点独占,且当前计算节点可对该页面数据进行修改操作,其它计算节点的本地内存缓存池中的同一个页面数据处于失效状态;加读锁状态表示页面数据可被当前计算节点进行读取操作,并且不允许其它计算节点对该页面数据进行修改操作。

35 例如,在运行上层应用过程中,当需要加载第二页面数据时,第二计算节点中的第一计算单元可以查询本地内存缓存池,判断是否存在可读取的第二页面数据;若不存在,向第二计算节点中的第一处理模块发送页面读指令,以从第二内存节点的共享内存缓存池中读取第二页面数据。或者,在运行上层应用过程中,当需要对第二页面数据进行改写时,40 第二计算节点中的第一计算单元可以查询本地内存缓存池,判断是否存在可改写的第二页面数据;若存在,则直接对本地内存缓存池中的第二页面数据进行改写,如果不存在,则向第二计算节点中的第一处理模块发送页面读指令,以从第二内存节点的共享内存缓存池中读取第二页面数据,进而对从共享内存缓存池中读取的第二页面数据进行改写操作。

在本实施例中,本地内存缓存池中不存在可读取的第二页面数据以及不存在可改写的

第二页面数据的情况进行了示例性说明，但并不限于下述列举的情况。例如，本地内存缓存池中不存在可读取的第二页面数据包括以下任一情况：本地内存缓存池中不存在第二页面数据；本地内存缓存池中存在被其它计算节点添加写锁的第二页面数据；本地内存缓存池中存在被标记为已失效状态的第二页面数据。相应地，本地内存缓存池中不存在可改写的第二页面数据包括以下任一情况：本地内存缓存池中不存在第二页面数据；本地内存缓存池中存在未被第二计算节点添加写锁的第二页面数据，也就是说，在该示例中，为了保持数据一致性，本地内存缓存池中的第二页面数据只有被第二计算节点添加写锁才能被第二计算节点中的第一计算单元进行改写（或修改）操作。

例如，在上述过程中，在第二计算节点中的第一计算单元需要对第二页面进行改写的情况下，若本地内存缓存池中不存在第二页面数据，第一计算单元可以根据本次改写操作的参数，确定本次改写操作涉及的第二页面数据是否属于已有页面数据；在第二页面数据是已有页面数据的情况下，向第二计算节点中的第一处理模块发送页面读指令，以从第二内存节点的共享内存缓存池中读取第二页面数据。在第二计算节点中的第一计算单元需要改写的第二页面数据不是已有页面数据的情况下，即第二计算节点中的第一计算单元需要向本地内存缓存池中的第二页面数据是一个新的页面数据，则可以直接向本地内存缓存池中写入第二页面数据。在一实施例中，第二计算节点中的第一计算单元还可以确定第二内存节点中的共享内存缓存池与本地内存缓存池之间是包含关系，在第二内存节点中的共享内存缓存池与本地内存缓存池之间是全部包含关系的情况下，向第二计算节点的第一处理模块发送内存分配请求，以供该第一处理模块通过第二计算节点的第一网卡模块向第二内存节点中的第二处理模块发送内存分配请求，以使第二内存节点中的第二处理模块请求第二内存节点中的第二计算单元从共享内存缓存池中为该新的第二页面数据分配内存空间，用以缓存后续被同步过来的第二页面数据。如果第二内存节点中的共享内存缓存池与本地内存缓存池之间是部分包含关系，则第二内存节点中的第二计算单元可以在接收到从第二计算节点同步过来的第二页面数据的情况下，实时为第二页面数据分配内存空间。其中，全部包含关系是指在各个计算节点的本地内存缓存池中的页面数据全部存储在至少一个内存节点的共享内存缓存池中，部分包含关系是指各个计算节点的本地内存缓存池中的页面数据并不一定全部存储在共享内存缓存池中。其中，共享内存缓存池和本地内存缓存池之间的包含关系可预先配置或确定。

针对页面数据加读锁处理的情况 3，由第二计算节点中的第一处理模块与第二内存节点中的第二处理模块的相互配合，为共享内存缓存池中待读取的第二页面数据添加读锁。

示例性的，为了提高对页面数据加读锁处理的加锁性能，第二计算节点中的第一处理模块，用于：在接收第二页面数据之前，向第二内存节点中的第二处理模块发送加读锁的请求，以使第二内存节点中的第二处理模块为第二页面数据添加读锁。

例如，第二内存节点中的第二处理模块，用于：在从共享内存缓存池中读取第二页面数据之前，通过第二内存节点中的第二网卡模块接收第二计算节点中的第一网卡模块发送的加读锁的请求，为第二页面数据添加读锁。

示例性的，第二内存节点中的第二处理模块为第二页面数据添加读锁的过程包括：第二内存节点中的第二处理模块通过共享内存缓存池的元数据，判断第二页面数据是否被其它计算节点添加了写锁；在第二页面数据未被其它计算节点添加写锁的情况下，可以直接为第二页面数据添加读锁，并将第二页面数据添加读锁的相关信息记录到共享内存缓存池的元数据中。其中，添加读锁的相关信息包括第二页面数据被哪个计算节点添加了读锁、添加读锁的时间等信息。在第二页面数据有被其它计算节点添加写锁的情况下，第二内存节点中的第二处理模块通过第二网卡模块向对第二页面数据添加写锁的计算节点发起解锁请求；对第二页面数据添加写锁的计算节点通过其第一网卡模块接收该解锁请求，并由

其第一处理模块判断是否可以对第二页面数据进行解锁，在确定可以对第二页面数据进行解锁的情况下，经第一网卡模块向第二内存节点的第二网卡模块发送解锁确认信息，第二内存节点的第二处理模块收到第二网卡模块提供的解锁确认信息后，为第二页面数据进行解锁，即将第二页面数据上的写锁删除并同步更新共享内存缓存池的元数据，之后重新为第二页面数据添加读锁，并在共享内存缓存池的元数据中记录第二页面数据添加读锁的相关信息。

示例性的，针对页面数据加写锁和作废处理的情况 4 和 5，为了保证数据一致性，在第三计算节点中的第一计算单元改写本地内存缓存池中的第三页面数据之前，可以通过第三计算节点中的第一处理模块和第三内存节点中的第二处理模块相互配合，完成共享内存缓存池中的第三页面数据加写锁和作废操作。其中，对第三页面数据加写锁是指为第三页面数据添加写锁的过程，对第三页面数据进行作废是指通知除第三计算节点之外的其它使用第三页面数据的计算节点将本地内存缓存池中的第三页面数据作废，以等待最新第三页面数据的过程。

示例性的，为了提高对页面数据加写锁和作废处理的性能，第三计算节点中的第一处理模块，用于：通过第三计算节点中的第一网卡模块向第三内存节点中的第二处理模块发送对第三页面数据进行加写锁和作废的请求，以使第三内存节点中的第二处理模块对共享内存缓存池中的第三页面数据添加写锁并通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理。

例如，第三内存节点中的第二处理模块，用于：通过第三内存节点中的第二网卡模块接收第三计算节点中的第一处理模块发送的请求对共享内存缓存池中的第三页面数据进行加写锁和作废的请求，对第三页面数据添加写锁并通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理。

在一实施例中，第三计算节点中的第一处理模块可以一并发送加写锁和作废的请求，相应地，第三内存节点中的第二处理模块可以依次对第三页面数据进行加写锁处理和作废处理。或者，在一实施例中，第三计算节点中的第一处理模块先向第三内存节点中的第二处理模块发送加写锁的请求，第三内存节点中的第二处理模块对第三页面数据进行加写锁处理，并在完成加写锁处理后通知第三计算节点中的第一处理模块；然后，第三计算节点中的第一处理模块向第三内存节点中的第二处理模块发送作废的请求，第三内存节点中的第二处理模块通过共享内存缓存池中的元数据确定本地内存缓存池中包含第三页面数据的其它计算节点，向其它计算节点的第一处理模块发送作废通知，以通知其它计算节点的第一处理模块对本地内存缓存池中的第三页面数据进行作废处理。其中，对本地内存缓存池中的第三页面数据进行作废处理具体可以是将本地内存缓存池中的第三页面数据标记为无效状态，当然也可以采用其它作废方式，例如删除第三页面数据。

参见图 3，以计算节点和内存节点中集成 CPU 和 CIPU 为例对涉及多个计算节点的数据一致性维护过程进行说明。在图 3 中仅示出了 CIPU，未对 CPU 进行图示。在图 3 中，在计算节点和内存节点的 CIPU 中集成了 DSA 模块，DSA 模块上运行本地缓存池管理模块或远程缓存池管理模块。当然，CIPU 上还可以集成网卡模块（如 NIC），网卡模块具有接收（Rx）和发送（Tx）两个接口，未图示。DSA 模块提供可编程能力，借助该可编程能力，DSA 模块在功能实现上更具灵活性。相对而言，第一计算单元（如 CPU）属于计算（CPU）域；CIPU 属于网络域，而且可以通过缓存（Cache）一致性通道访问计算（CPU）域中的本地内存缓存池。这样，NIC 在接收到网络传输的页面数据时，可以不通过与 CPU 之间的输入输出（Input/Output，IO）通道上报给 CPU，而是就由 DSA 模块上运行的本地缓存池管理模块直接进行处理，同样，DSA 模块处理过后需要发送的页面数据也可以直接通过 NIC 发送，不需要通过与 CPU 的通道上传到 CPU 上去进行处理。Cache 一致性通道

为 DSA 模块访问本地内存缓存池提供了比 IO 通道更为高效的通道，特别是对于像哈希表和链表等需要间接地址访问和离散地址访问的数据结构而言。由于 DSA 和 NIC 都位于 CIPU 上，且紧密耦合，因此，DSA 可以通过调整 NIC 的控制信息来实时改变 NIC 的行为。

在本实施例中，整个一致性的维护操作由 CIPU 完成，不需要跨 IO 通道上报 CPU，
5 整个一致性维护过程省略了 IO 通道的延迟。例如，当计算节点进行事务确认的时候，如果该计算节点发现该事务所涉及到的页面数据并非该计算节点独有，那么需要对其他计算节点进行状态同步。参见图 3 中的①，计算节点 A 的本地缓存池管理器需要发送一个事务请求给远端的内存节点 D 的远端缓存池管理器，该事务请求包括事务所涉及到的页面数据的元数据。参见图 3 中的②所示，远端的内存节点的远端缓存池管理器向事务涉及的其他
10 计算节点 B 和 C 发送事务通知，其他计算节点 B 和 C 接收到事务通知后，需要对涉及到的页面数据的状态进行更新。参见图 3 中的③所示，其他计算节点 B 和 C 向远端的内存节点 D 的远端缓存池管理器发送确认信息。参见图 3 中的④所示，远端的内存节点 D 的远端缓存池管理器在接收到所有其他计算节点 B 和 C 的确认信息后，向最初发出请求的计算节点 A 上的本地缓存池管理器发送确认信息。参见图 3 中的⑤所示，发出请求的计算节点 A
15 的本地缓存池管理器收到确认信息后，执行事务确认操作。

其中，上述涉及多个计算节点的数据一致性维护事务可以是但不限于：对页面数据进行作废的事务。以对页面数据进行作废为例，图 3 所示事务流程具体为：计算节点 A 的 CIPU 针对需要作废的页面数据向内存节点 D 发送作废请求，内存节点 D 是指共享内存缓存池中存储有待作废的页面数据的内存节点，计算节点 A 可以根据本地内存缓存池的元数据
20 以及其它相关信息，例如所维护的其与各内存节点之间的对应关系以及各内存节点的共享内存缓存池中存储的页面数据，确定待作废的页面数据所在的内存节点 D。内存节点 D 的 CIPU 接收到作废请求后，根据共享内存缓存池的元数据以及其它相关信息确定本地内存缓存池中存在待作废的页面数据的其它计算节点，即计算节点 B 和 C；然后，向计算节点 B 和 C 分别发送作废通知，并在收到计算节点 B 和 C 返回的作废确认信息后，向计算
25 节点 A 发送作废确认信息；计算节点 A 收到内存节点 D 发送的作废确认信息后，通知计算节点 A 的 CPU 执行针对页面数据的改写操作，以完成作废事务的过程。

在此说明，本申请实施例并不限定计算节点内的第一处理单元（如 CIPU）和内存节点中的第二处理单元（如 CIPU）在接收到请求或页面数据后的内部处理逻辑，凡是不需要跨 IO 通道上报 CPU 的数据处理方式均适用于本申请实施例。下面结合图 4，对计算节点内的第一处理单元或内存节点中的第二处理单元（如 CIPU）内部接收到页面数据的内部处理过程进行示例性的说明。
30

在一实施方式中，图 4 示出了某个计算节点接收页面状态更新请求后的处理流程。首先，通过 NIC 接口接收网络包，涉及页面状态更新的元数据可能被划分为一个或多个网络包进行传输，页面元数据被切分的网络包的数量与 NIC 的接收能力以及页面元数据的大小
35 有关。在此说明，在计算节点接收页面数据的情况下，页面数据也可以被划分为多个网络包进行传输。在本申请实施例中，DAS 模块具有可编程能力，能够支持各种大小的页面数据，可灵活适配上层应用。在图 4 中，以页面元数据被且切分为多个网络包为例，故在 DAS 模块上可由后台接收线程对网络包进行组装，得到整个页面元数据后，将该页面元数据进行压栈处理，先将其写入日志元数据队列中，并通知缓存池维护线程，缓存池维护线程按照先进先出的方式从日志元数据队列中读取页面元数据，直接通过缓存一致性通道对 CPU
40 域的本地内存缓存池的元数据进行更新，与此同时更新 WAL 日志数据，在完成页面状态更新后，缓存池维护线程向后台接收线程返回确认消息。需要说明的是，在需要向对端返回确认信息的情况下，也可以通过 NIC 接口对外发送包含确认信息的网络包。上述一系列操作都在 DAS 模块上执行，并不需要通过 IO 通道上报 CPU，即大多数操作都在 CIPU 域

完成,更新后的本地内存缓存池的元数据对于 CPU 域来说是即时可见的。而且,由于 DSA 模块和 NIC 接口是在 CIPU 上紧耦合的,网络包不再通过 IO 通道上报至 CPU 进行处理,消除了 IO 通道传输和拷贝的延迟。

5 在一些实施例中,每个计算节点中的上层应用为至少一个,每个上层应用对应一个本地内存缓存池,在本实施例中,将数据一致性维护任务下推到第一处理单元,意味着需要写入各个上层应用对应的本地内存缓存池的页面数据或需要给到各个上层应用的其它数据(例如通知消息)都会通过第一处理单元中的第一网卡模块进行接收。基于此,为了提高第一网卡模块接收数据的及时性和效率,计算节点中的第一处理模块可以对第一网卡模块的接收队列进行轮询;在接收队列中存在待处理数据的情况下,从接收队列中读取待处理数据,确定待处理数据所属的上层应用,并将待处理数据写入所属的上层应用对应的本地内存缓存池或本地内存缓存池的元数据中,以供第一计算单元对待处理数据进行处理,待处理数据包括页面数据或通知消息。

参见图 5,假设一个计算节点上部署有多个虚拟机(Virtual Machine, VM)实例,在传统模式下,每个虚拟机维护自己的接收队列并各自采用池化(pooling)模式各自进行网卡报文的收发,pooling 模式是指采用轮询方式进行网卡报文的收发,避免采用中断方式造成的响应延迟,以提升网卡收发性能。对于时延敏感应用,pooling 操作需要独立的 CPU 核来完成,这会造成 CPU 资源的浪费。在本申请实施例中,将相关的缓存池一致性维护逻辑下推至第一处理单元或第二处理单元(如 CIPU)之后,根据网络通信的带宽,各 VM 实例的 pooling 任务也实现了下推,可由第一处理单元或第二处理单元(例如 CIPU 中的 DSA 模块)指定的 CPU 核进行。此时,DSA 上执行 pooling 任务所需要的 CPU 核只要满足带宽需求即可,不再需要多个 CPU 核,可以进一步提升 pooling 任务能效比,节约 CPU 资源。

25 实际应用中,当计算节点的本地内存缓存池中的脏页面数据较多时,需要及时将本地内存缓存池中的脏页面数据回写到远端内存节点的共享内存缓存池中,此时脏页的刷新速度严重影响系统性能。其中,向共享内存缓存池中回写脏页面数据的情况是向共享内存缓存池中写入页面数据的一种具体应用场景,即上文中的第一页面数据可以是需要回写的脏页面数据。

基于上述,如果上文中的第一页面数据是需要回写的脏页面数据,则第一计算机节点中的第一处理模块可以将第一页面数据写入本地双写缓存区中,通过第一计算机节点中的第一网卡模块将本地双写缓存区(Doublewrite Buffer)中的第一页面数据发送给第一内存节点上的第二处理模块,以供第一内存节点中的第二处理模块将第一页面数据写入第一内存节点上的双写缓存区中并将第一内存节点的共享内存缓存池中指向第一页面数据的地址指针指向第一内存节点的双写缓存区。相应地,对第一内存节点中的第二处理模块还用于:在接收到第一页面数据的情况下,将所述第一页面数据写入本地双写缓存区中,并将指向共享内存缓存池中的第一页面数据的地址指针指向本地双写缓存区,以从本地双写缓存区中读取最新的第一页面数据。通过交换地址指针的方式可以减少一次写操作,有利于提高脏页面数据的回写效率。

其中,图 6 中左边所示为传统回写过程,在传统回写过程中,为了保证页面数据的原子性,需要采用 double write(双写)机制,即参见步骤①,将计算节点的本地内存缓存池中的脏页面数据写入本地 Doublewrite Buffer;接着如步骤②,将计算节点的本地 Doublewrite Buffer 中的脏页面数据写入远端内存节点上的 Doublewrite Buffer;之后,如步骤③,将计算节点的本地 Doublewrite Buffer 中的脏页面数据写入远端内存节点上的共享内存缓存池中。这样,一次脏页面数据的回写需要向远端内存节点执行两次写操作,即步骤②和③,极大增加了延迟。

与此类比, 在本实施例中, 在计算节点和内存节点中增加处理单元 (如 CIPU) 之后, 可以高效地消除传统 Double write 机制存在的问题。在图 6 中右边所示为本实施例改进后的回写过程, 参见步骤①, 将计算节点的本地内存缓存池上的脏页面数据写入本地的 Doublewrite Buffer; 接着参见步骤②, 将计算节点本地的 Doublewrite Buffer 中的脏页面数据写入远端内存节点上的 Doublewrite Buffer; 接着如步骤③, 在共享内存缓存池的元数据和网卡模块 (如 NIC) 的控制信息中修改指向该脏页面数据的地址信息, 在共享内存缓存池的元数据中将指向旧页面数据的地址信息 drr1 修改为本地 Doublewrite Buffer 中刚接收到的脏页面数据的地址信息 drr2, 以采用回写完成的脏页面数据代替之前共享内存缓存池中存储的旧的页面数据, 同时, 在网卡模块 (如 NIC) 的控制信息中携带共享内存缓存池中存储的旧页面数据的地址信息 drr1, 以将本地 Doublewrite Buffer 中回写完成的脏页面数据的地址信息 drr2 修改为共享内存缓存池中之前存储的旧页面数据的地址信息 drr1, 这两个操作之间具有原子性。这样, 在下次进行脏页面回写时, 计算节点在向远端内存节点的 Doublewrite Buffer 中同样的位置 drr2 进行回写时, 实际写入的是进行修改映射后的共享内存缓存池中的地址信息 drr1。另外, 在计算节点对远端内存节点的共享内存缓存池中同样的位置 drr1 进行读取时, 实际读取的是进行修改映射后的远端内存节点的 Doublewrite Buffer 中的位置 drr2, 确保读取最新的页面数据。由于在内存节点的第二处理单元上第二处理模块 (如 DSA) 和第二网卡模块 (如 NIC) 紧密耦合, 因此可以通过本地高效地地址修改映射来代替额外的一次网络数据传输, 有利于提高回写效率。

本申请实施例除了提供上述存算分离系统之外, 还提供有基于存算分离系统实现的内存缓存池维护方法。本申请实施例还一种内存缓存池维护方法。该方法应用于计算节点, 计算节点包括第一计算单元和第一处理单元, 第一处理单元与第一计算单元互联, 以及与至少一个内存节点中的第二处理单元网络互联, 该方法可以包括: 第一处理单元与至少一个内存节点中的第二处理单元配合, 在第一计算单元为上层应用提供的本地内存缓存池与至少一个内存节点为上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步, 以实现数据一致性; 其中, 本地内存缓存池能够被第一处理单元直接访问。

在一实施方式中, 在第一计算单元为上层应用提供的本地内存缓存池与至少一个内存节点为上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步, 包括: 从本地内存缓存池中读取第一页面数据, 并将第一页面数据发送给第一内存节点中的第二处理单元, 以供第一内存节点中的第二处理单元将第一页面数据写入第一内存节点中的共享内存缓存池中; 和/或, 接收第二内存节点中的第二处理单元发送的第二页面数据, 将第二页面数据写入本地内存缓存池中, 其中, 第二内存节点中的第二处理单元从第二内存节点的共享内存缓存池中读取第二页面数据并发送给第一处理单元; 和/或, 在接收第二页面数据之前, 向第二内存节点中的第二处理单元发送加读锁的请求, 以使第二内存节点中的第二处理单元为第二页面数据添加读锁; 和/或, 在第一计算单元改写本地内存缓存池中的第三页面数据之前, 向第三内存节点中的第二处理单元发送对第三页面数据进行加写锁和作废的请求, 以使第三内存节点中的第二处理单元对共享内存缓存池中的第三页面数据添加写锁并通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理。

在一些实施例中, 上述方法还包括: 响应第一计算单元发送的页面读指令, 向第二内存节点中的第二处理单元发送页面读请求, 以使第二内存节点中的第二处理单元从第二内存节点的共享内存缓存池中读取第二页面数据并返回。

在一实施方式中, 从本地内存缓存池中读取第一页面数据, 包括: 响应第一计算单元发送的脏页面回写指令, 从本地内存缓存池中读取第一页面数据, 脏页面回写指令是第一计算单元根据页面替换机制确定存在需要淘汰且处于已修改状态的第一页面数据时发送

的；或者，根据页面替换机制确定本地内存缓存池中存在需要淘汰且处于已修改状态的第一页面数据，从本地内存缓存池中读取第一页面数据。

在一实施例中，将第一页面数据发送给第一内存节点中的第二处理单元，包括：将第一页面数据写入本地双写缓存区中，将本地双写缓存区中的第一页面数据发送给第一内存节点上的第二处理单元，以供第二处理单元将第一页面数据写入第一内存节点上的双写缓存区中并将第一内存节点的共享内存缓存池中指向第一页面数据的地址指针指向第一内存节点的双写缓存区。

在一实施方式中，上层应用为至少一个，一个上层应用对应一个本地内存缓存池；该方法还包括：对第一处理单元中第一网卡模块的接收队列进行轮询；在接收队列中存在待处理数据的情况下，从接收队列中读取待处理数据；确定待处理数据所属的上层应用，并将其写入其所属的上层应用对应的本地内存缓存池中，以供第一计算单元对待处理数据进行处理，待处理数据包括页面数据或通知消息。

本申请实施例还一种内存缓存池维护方法的流程图，应用于内存节点，内存节点包括第二计算单元和第二处理单元，第二处理单元与第二计算单元互联，以及与至少一个计算节点中的第一处理单元网络互联，该方法包括：第二处理单元与至少一个计算节点中的第一处理单元配合，在内存节点为上层应用提供的共享内存缓存池与至少一个计算节点为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步，以实现数据一致性；其中，共享内存缓存池能够被第二处理单元直接访问。

在一些实施方式中，在内存节点为上层应用提供的共享内存缓存池与至少一个计算节点为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步，包括：接收第一计算节点中的第一处理单元发送的第一页面数据，将第一页面数据写入共享内存缓存池中；和/或，从共享内存缓存池中读取第二页面数据，并将第二页面数据发送给第二计算节点中的第一处理单元，以供第二计算节点中的第一处理单元将第二页面数据写入第二计算节点的本地内存缓存池中；和/或，在读取第二页面数据之前，接收第二计算节点中的第一处理单元发送的加读锁的请求，为第二页面数据添加读锁；和/或，接收第三计算节点中的第一处理单元发送的请求对共享内存缓存池中的第三页面数据进行加写锁和作废的请求，对第三页面数据添加写锁并通知本地内存缓存池中包含第三页面数据的其它计算节点对第三页面数据进行作废处理。

在一实施例中，上述方法还包括：接收第二计算节点中的第一处理单元发送的页面读请求，根据页面读请求，查询共享内存缓存池中是否存在可读取的第二页面数据，若不存在，控制存在可读取的第二页面数据的其它计算节点将第二页面数据回写到共享内存缓存池中。

在一实施例中，上述方法还包括：判断第二页面数据是否添加有写锁；若添加有写锁，向为第二页面数据添加写锁的计算节点发送解锁请求；在为第二页面数据添加写锁的计算节点返回解锁确认信息后，删除第二页面数据的写锁，并为第二页面数据添加读锁。

在一实施例中，上述方法还包括：在回写第一页面数据的情况下，将第一页面数据写入本地双写缓存区中；将指向共享内存缓存池中的第一页面数据的地址指针指向本地双写缓存区，以从本地双写缓存区中读取回写的第一页面数据。

为了更好地理解，下面介绍计算节点和内存节点之间的信令交互过程。

图7为本申请实施例提供的示例性的一种内存缓存池维护方法的信令交互图。参见图7，该方法可以包括S71-S73。

S71、第一计算节点中的第一处理单元从本地内存缓存池中读取第一页面数据。

其中，本地内存缓存池处于第一计算节点的第一计算单元中。

S72、第一计算节点中的第一处理单元将第一页面数据发送给第一内存节点中的第二

处理单元。

S73、第一内存节点中的第二处理单元将第一页面数据写入共享内存缓存池中。

其中，共享内存缓存池处于第一内存节点的第二计算单元中。

图 8 为本申请实例性的另一种内存缓存池维护方法的信令交互图。参见图 8，该方法

5 可以包括 S81-S85。

S81、第二内存节点中的第二处理单元从共享内存缓存池中读取第二页面数据。

S82、第二计算节点向第二内存节点中的第二处理单元发送加读锁的请求。

S83、第二内存节点中的第二处理单元为第二页面数据添加读锁。

10 S84、第二内存节点中的第二处理单元向第二计算节点中的第一处理单元发送第二页面数据。

S85、第二计算节点中的第一处理单元将第二页面数据写入本地内存缓存池。

其中，本地内存缓存池处于第二计算节点的第一计算单元中。

图 9 为本申请实例性的另一种内存缓存池维护方法的信令交互图。参见图 9，该方法可以包括 S91-S93。

15 S91、在第一计算单元改写本地内存缓存池中的第三页面数据之前，第三计算节点中的第一处理单元向第三内存节点中的第二处理单元发送对第三页面数据进行加写锁和作废的请求。

S92、第三内存节点中的第二处理单元对共享内存缓存池中的第三页面数据添加写锁。

20 S93、第三内存节点中的第二处理单元通知本地内存缓存池中包含第三页面数据的其它第三计算节点对所述第三页面数据进行作废处理。

关于上述实施例方法中各步骤的详细实施方式以及有益效果已经在前述实施例中进行了详细描述，此处将不做详细阐述说明。

25 图 10 为本申请实施例提供的一种处理单元的结构示意图。该处理单元可以作为前述实施例中计算节点中的第一处理单元实现，也可以作为前述实施例中内存节点中的第二处理单元实现。如图 10 所示，该处理单元包括：处理模块 100、网卡模块 110 和存储模块 120。

存储模块 120，用于存储计算机程序，并可被配置为存储其它各种数据以支持在处理单元上的操作。这些数据的示例包括用于在处理单元上操作的任何应用程序或方法的指令，消息，图片，视频等。

30 处理模块 100 可以由任何类型的易失性或非易失性存储设备或者它们的组合实现，如静态随机存取存储器（Static Random-Access Memory, SRAM），电可擦除可编程只读存储器（Electrically Erasable Programmable read only memory, EEPROM），可擦除可编程只读存储器（Erasable Programmable Read Only Memory, EPROM），可编程只读存储器（Programmable read-only memory, PROM），只读存储器（Read-Only Memory, ROM），磁存储器，快闪存储器，磁盘或光盘。

35 处理模块 100，与存储模块 120 耦合，用于执行存储模块 120 中的计算机程序，以用于：执行上述实施例中可由第一处理单元或第二处理单元执行的内存缓存池维护方法中的步骤。

40 在一实施例中，如图 10 所示，该处理单元还包括：通信组件 130、显示器 140、电源组件 150、音频组件 160 等其它组件。图 10 中仅示意性给出部分组件，并不意味着处理单元只包括图 10 所示组件。另外，图 10 中虚线框内的组件为可选组件，而非必选组件，具体可视处理单元的产品形态而定。

关于处理模块执行各动作的详细实施过程可参见前述方法实施例或设备实施例中的相关描述，在此不再赘述。

相应地，本申请实施例还提供一种存储有计算机程序的计算机可读存储介质，计算机

程序被执行时能够实现上述方法实施例中可由第一处理单元或第二处理单元执行的各步骤。

相应地，本申请实施例还提供一种计算机程序产品，包括计算机程序/指令，当计算机程序/指令被处理器执行时，致使处理器能够实现上述方法实施例中可由第一处理单元或第二处理单元执行的各步骤。

在本申请实施例中，针对存算分离架构，在计算节点和内存节点中增设与计算单元互连的处理单元，将内存缓存数据的一致性维护任务下推到处理单元上执行，而不是在计算单元上完成，由于处理单元具有对本地内存缓存池或共享内存缓存池直接访问的能力，使得内存缓存数据一致性维护过程中的数据传输过程大都在处理单元内部完成，无需上传至计算单元，能够消除计算单元和网卡之间进行数据传输的延迟，达到在维护缓存数据一致性的过程中降低数据传输延迟的目的。

上述通信组件被配置为便于通信组件所在设备和其他设备之间有线或无线方式的通信。通信组件所在设备可以接入基于通信标准的无线网络，如 WiFi (Wireless Fidelity, 无线保真)、2G (2Generation, 2 代)、3G (3Generation, 3 代)、4G (4Generation, 4 代)/LTE (long Term Evolution, 长期演进)、5G (5Generation, 5 代) 等移动通信网络，或它们的组合。在一个示例性实施例中，通信组件经由广播信道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中，所述通信组件还包括近场通信 (Near Field Communication, NFC) 模块，以促进短程通信。例如，在 NFC 模块可基于射频识别 (Radio Frequency Identification, RFID) 技术，红外数据协会 (The Infrared Data Association, IrDA) 技术，超宽带 (Ultra Wide Band, UWB) 技术，蓝牙 (Bluetooth, BT) 技术和其他技术来实现。

上述显示器包括屏幕，其屏幕可以包括液晶显示器 (Liquid Crystal Display, LCD) 和触摸面板 (Touch Panel, TP)。如果屏幕包括触摸面板，屏幕可以被实现为触摸屏，以接收来自用户的输入信号。触摸面板包括一个或多个触摸传感器以感测触摸、滑动和触摸面板上的手势。所述触摸传感器可以不仅感测触摸或滑动动作的边界，而且还检测与所述触摸或滑动操作相关的持续时间和压力。

上述电源组件，为电源组件所在设备的各种组件提供电力。电源组件可以包括电源管理系统，一个或多个电源，及其他与为电源组件所在设备生成、管理和分配电力相关联的组件。

上述音频组件，可被配置为输出和/或输入音频信号。例如，音频组件包括一个麦克风 (microphone, MIC)，当音频组件所在设备处于操作模式，如呼叫模式、记录模式和语音识别模式时，麦克风被配置为接收外部音频信号。所接收的音频信号可以被进一步存储在存储器或经由通信组件发送。在一些实施例中，音频组件还包括一个扬声器，用于输出音频信号。

本领域内的技术人员应明白，本申请的实施例可提供为方法、系统、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可读存储介质 (包括但不限于磁盘存储器、CD-ROM、光学存储器等) 上实施的计算机程序产品的形式。

本申请是参照根据本申请实施例的方法、设备 (系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用

于实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能。

这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和 / 或方框图一个方框或多个方框中指定的功能的步骤。

在一个典型的配置中，计算设备包括一个或多个处理器(central processing unit, CPU)、输入/输出接口、网络接口和内存。

内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器 (Random Access Memory, RAM)和/或非易失性内存等形式，如只读存储器(Read Only Memory, ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变化内存(Phase Change RAM, PRAM)、静态随机存取存储器(Static Random-Access Memory, SRAM)、动态随机存取存储器(Dynamic Random Access Memory, DRAM)、其他类型的随机存取存储器(Random Access Memory, RAM)、只读存储器(Read Only Memory, ROM)、电可擦除可编程只读存储器(Electrically-Erasable Programmable Read-Only Memory, EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(Digital versatile disc, DVD)或其他光学存储、磁盒式磁带，磁带磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体(transitory media)，如调制的数据信号和载波。

还需要说明的是，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

以上所述仅为本申请的实施例而已，并不用于限制本申请。对于本领域技术人员来说，本申请可以有各种更改和变化。凡在本申请的精神和原理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权利要求范围之内。

权 利 要 求

1. 一种计算节点，包括：第一计算单元和第一处理单元，所述第一计算单元与所述第一处理单元互联；
- 5 所述第一计算单元，用于运行上层应用，并为所述上层应用提供用于缓存页面数据的本地内存缓存池，所述本地内存缓存池能够被所述第一处理单元直接访问；
- 所述第一处理单元，还与至少一个内存节点中的第二处理单元网络互联，用于与所述第二处理单元进行数据交互，以在本地内存缓存池与所述至少一个内存节点为所述上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步。
- 10 2. 根据权利要求1所述的计算节点，其中，所述第一处理单元包括：第一处理模块和第一网卡模块，所述第一处理模块通过互联通道与所述第一计算单元互联，所述第一网卡模块与所述第二处理单元网络互联。
3. 根据权利要求2所述的计算节点，其中，所述第一处理模块，用于：
- 从本地内存缓存池中读取第一页面数据，并通过所述第一网卡模块发送给第一内存节点中的第二处理单元，以供所述第一内存节点中的第二处理单元将所述第一页面数据写入
- 15 所述第一内存节点中的共享内存缓存池中；
- 和/或，
- 通过所述第一网卡模块接收第二内存节点中的第二处理单元发送的第二页面数据，将所述第二页面数据写入本地内存缓存池中，其中，所述第二内存节点中的第二处理单元从所述第二内存节点的共享内存缓存池中读取所述第二页面数据并发送给所述第一网卡模块；
- 20 和/或，
- 在接收所述第二页面数据之前，向所述第二内存节点中的第二处理单元发送加读锁的请求，以使所述第二内存节点中的第二处理单元为所述第二页面数据添加读锁；
- 和/或，
- 25 在所述第一计算单元改写本地内存缓存池中的第三页面数据之前，通过所述第一网卡模块向第三内存节点中的第二处理单元发送对第三页面数据进行加写锁和作废的请求，以使所述第三内存节点中的第二处理单元对共享内存缓存池中的第三页面数据添加写锁并通知本地内存缓存池中包含所述第三页面数据的其它计算节点对所述第三页面数据进行作废处理。
- 30 4. 一种内存节点，包括：第二计算单元和第二处理单元，所述第二计算单元与所述第二处理单元互联；
- 所述第二计算单元，用于为至少一个计算节点上运行的上层应用提供共享内存缓存池，所述共享内存缓存池能够被所述第二处理单元直接访问；
- 所述第二处理单元，还与至少一个计算节点中的第一处理单元网络互联，用于与所述
- 35 第一处理单元进行数据交互，以在共享内存缓存池与所述至少一个计算节点为所述上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步。
5. 一种内存缓存池维护方法，应用于计算节点，所述计算节点包括第一计算单元和第一处理单元，所述第一处理单元与所述第一计算单元互联，以及与至少一个内存节点中的

第二处理单元网络互联，所述方法包括：

所述第一处理单元与所述至少一个内存节点中的第二处理单元进行数据交互，以在所述第一计算单元为上层应用提供的本地内存缓存池与所述至少一个内存节点为所述上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步；

5 其中，所述本地内存缓存池能够被所述第一处理单元直接访问。

6. 根据权利要求5所述的方法，其中，所述第一处理单元与所述至少一个内存节点中的第二处理单元进行数据交互，以在所述第一计算单元为上层应用提供的本地内存缓存池与所述至少一个内存节点为所述上层应用提供的共享内存缓存池之间进行页面数据和/或页面状态的同步，包括：

10 从本地内存缓存池中读取第一页面数据，并将所述第一页面数据发送给第一内存节点中的第二处理单元，以供所述第一内存节点中的第二处理单元将所述第一页面数据写入所述第一内存节点中的共享内存缓存池中；

和/或，

15 接收第二内存节点中的第二处理单元发送的第二页面数据，将所述第二页面数据写入本地内存缓存池中，其中，所述第二内存节点中的第二处理单元从所述第二内存节点的共享内存缓存池中读取所述第二页面数据并发送给所述第一处理单元；

和/或，

在接收所述第二页面数据之前，向所述第二内存节点中的第二处理单元发送加读锁的请求，以使所述第二内存节点中的第二处理单元为所述第二页面数据添加读锁；

20 和/或，

在所述第一计算单元改写本地内存缓存池中的第三页面数据之前，向第三内存节点中的第二处理单元发送对第三页面数据进行加写锁和作废的请求，以使所述第三内存节点中的第二处理单元对共享内存缓存池中的第三页面数据添加写锁并通知本地内存缓存池中包含所述第三页面数据的其它计算节点对所述第三页面数据进行作废处理。

25 7. 根据权利要求6所述的方法，其中，从本地内存缓存池中读取第一页面数据，包括：响应所述第一计算单元发送的脏页面回写指令，从本地内存缓存池中读取第一页面数据，所述脏页面回写指令是所述第一计算单元根据页面替换机制确定存在需要淘汰且处于已修改状态的第一页面数据时发送的；

或者，

30 根据页面替换机制确定本地内存缓存池中存在需要淘汰且处于已修改状态的页面数据为第一页面数据，从本地内存缓存池中读取所述第一页面数据。

8. 根据权利要求6所述的方法，其中，将所述第一页面数据发送给第一内存节点中的第二处理单元，包括：

35 将所述第一页面数据写入本地双写缓存区中，将所述本地双写缓存区中的第一页面数据发送给所述第一内存节点上的第二处理单元，以供所述第二处理单元将所述第一页面数据写入所述第一内存节点上的双写缓存区中，并将所述第一内存节点的共享内存缓存池中指向第一页面数据的地址指针指向所述双写缓存区。

9. 根据权利要求6-8任一项所述的方法，其中，所述上层应用为至少一个，一个上层

应用对应一个本地内存缓存池；所述方法还包括：

对所述第一处理单元中第一网卡模块的接收队列进行轮询；在所述接收队列中存在待处理数据的情况下，从所述接收队列中读取所述待处理数据；

5 确定所述待处理数据所属的上层应用，并将所述待处理数据写入所属的上层应用对应的本地内存缓存池中，以供所述第一计算单元对所述待处理数据进行处理，所述待处理数据包括页面数据或通知消息。

10. 一种内存缓存池维护方法，应用于内存节点，所述内存节点包括第二计算单元和第二处理单元，所述第二处理单元与所述第二计算单元互联，以及与至少一个计算节点中的第一处理单元网络互联，所述方法包括：

10 所述第二处理单元与所述至少一个计算节点中的第一处理单元进行数据交互，以在所述内存节点为上层应用提供的共享内存缓存池与所述至少一个计算节点为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步；

其中，所述共享内存缓存池能够被所述第二处理单元直接访问。

15 11. 根据权利要求10所述的方法，其中，所述第二处理单元与所述至少一个计算节点中的第一处理单元进行数据交互，以在所述内存节点为所述上层应用提供的共享内存缓存池与所述至少一个计算节点为上层应用提供的本地内存缓存池之间进行页面数据和/或页面状态的同步，包括：

接收第一计算节点中的第一处理单元发送的第一页面数据，将所述第一页面数据写入所述共享内存缓存池中；

20 和/或，

从所述共享内存缓存池中读取第二页面数据，并将所述第二页面数据发送给第二计算节点中的第一处理单元，以供所述第二计算节点中的第一处理单元将所述第二页面数据写入所述第二计算节点的本地内存缓存池中；

和/或，

25 在读取所述第二页面数据之前，接收所述第二计算节点中的第一处理单元发送的加读锁的请求，为所述第二页面数据添加读锁；

和/或，

30 接收第三计算节点中的第一处理单元发送的请求对共享内存缓存池中的第三页面数据进行加写锁和作废的请求，对所述第三页面数据添加写锁并通知本地内存缓存池中包含所述第三页面数据的其它计算节点对所述第三页面数据进行作废处理。

12. 根据权利要求11所述的方法，其中，还包括：

在回写所述第一页面数据的情况下，将所述第一页面数据写入本地双写缓存区中；

将指向所述共享内存缓存池中的第一页面数据的地址指针指向本地双写缓存区，以从本地双写缓存区中读取回写的第一页面数据。

35 13. 一种存算分离系统，包括：多个如权利要求1-3任一项所述的计算节点、至少一个如权利要求4所述的内存节点以及至少一个存储节点；

其中，所述至少一个存储节点，用于持久化存储所述共享内存缓存池和/或本地内存缓存池中的页面数据。

14. 一种处理单元，所述处理单元包括：处理模块、网卡模块和存储模块，所述存储模块中存储有计算机程序，所述处理模块执行所述存储模块中存储的计算机程序，以用于实现权利要求 5-9 以及权利要求 10-12 任一项所述方法中的步骤。

15. 一种存储有计算机程序的计算机可读存储介质，当所述计算机程序被处理器执行
5 时，致使所述处理器能够实现权利要求 5-9 以及权利要求 10-12 任一项所述方法中的步骤。