(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106940660 B
(45)授权公告日 2020.08.14

(21)申请号 201610006181.8

(22)申请日 2016.01.05

(65)同一申请的已公布的文献号
申请公布号 CN 106940660 A

(43)申请公布日 2017.07.11

(73)专利权人 阿里巴巴集团控股有限公司
地址 英属开曼群岛大开曼资本大厦一座四层847号邮箱

(72)发明人 樊宏伟 李兆贵 卢嘉喜 李少翀 王义龙 蒋超

(74)专利代理机构 北京博思佳知识产权代理有限公司 11415
代理人 林祥

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/54(2006.01)

(56)对比文件
CN 104866339 A,2015.08.26
CN 104346285 A,2015.02.11
CN 102156720 A,2011.08.17
CN 102097084 A,2011.06.15
CN 104065568 A,2014.09.24

审查员 王丽娜

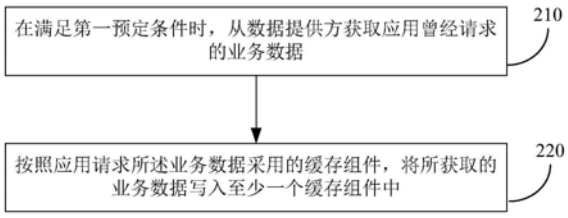
权利要求书2页 说明书7页 附图4页

(54)发明名称

缓存的实现的方法和装置

(57)摘要

本申请提供一种缓存的实现方法，所述缓存包括至少两个缓存组件，所述方法包括：在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据；按照应用请求所述业务数据采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中。通过本申请的技术方案，避免了因数据不一致导致的业务处理错误，极大的减少了并发障碍发生的可能性。



CN 106940660 B

1.一种缓存的实现方法，其特征在于，所述缓存包括至少两个不同缓存技术的缓存组件，所述方法包括：

当缓存更新周期到时，合并所有缓存组件中需要进行更新的业务数据，并记录每个业务数据所在的至少一个缓存组件；

从数据提供方获取合并后的每个业务数据；

按照记录的每个业务数据所在的缓存组件，利用获取的所述业务数据对所述缓存组件中的所述业务数据进行更新。

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：在收到应用采用某个缓存组件访问业务数据的请求，且所述缓存组件中不存在被请求的业务数据时，从数据提供方获取应用曾经请求的业务数据。

3.根据权利要求2所述的方法，其特征在于，所述方法还包括：按照应用所采用的缓存组件，将所获取的业务数据返回给应用。

4.根据权利要求1所述的方法，其特征在于，所述方法还包括：当符合第二预定条件的业务数据更新完成后，通知应用进行与所述业务数据相关的业务操作。

5.根据权利要求1至4任意一项所述的方法，其特征在于，所述方法还包括：

在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之前，开启对所述缓存组件或所述缓存组件中的所述业务数据的同步锁；

在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之后，释放对所述缓存组件或所述缓存组件中的所述业务数据的同步锁。

6.根据权利要求1所述的方法，其特征在于，所述方法还包括：收到应用采用某个缓存组件访问业务数据的请求后，将所述缓存组件中的被请求的业务数据返回给应用。

7.根据权利要求1所述的方法，其特征在于，所述缓存组件包括：ConcurrentMap缓存组件、OSCache缓存组件和/或EHCache缓存组件。

8.一种缓存的实现装置，其特征在于，所述缓存包括至少两个不同缓存技术的缓存组件，所述装置包括：

更新数据合并单元，用于当缓存更新周期到时，合并所有缓存组件中需要进行更新的业务数据，并记录每个业务数据所在的至少一个缓存组件；

业务数据获取单元，用于从数据提供方获取合并后的每个业务数据；

缓存组件写入单元，用于按照记录的每个业务数据所在的缓存组件，利用获取的所述业务数据对所述缓存组件中的所述业务数据进行更新。

9.根据权利要求8所述的装置，其特征在于，所述业务数据获取单元还用于：在收到应用采用某个缓存组件访问业务数据的请求，且所述缓存组件中不存在被请求的业务数据时，从数据提供方获取应用曾经请求的业务数据。

10.根据权利要求9所述的装置，其特征在于，所述装置还包括：业务数据返回单元，用于按照应用所采用的缓存组件，将所获取的业务数据返回给应用。

11.根据权利要求8所述的装置，其特征在于，所述装置还包括：缓存变化监听单元，用于当符合第二预定条件的业务数据更新完成后，通知应用进行与所述业务数据相关的业务操作。

12.根据权利要求8至11任意一项所述的装置，其特征在于，所述装置还包括：

同步锁开启单元，用于在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之前，开启对所述缓存组件或所述缓存组件中的所述业务数据的同步锁；

同步锁释放单元，用于在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之后，释放对所述缓存组件或所述缓存组件中的所述业务数据的同步锁。

13.根据权利要求8所述的装置，其特征在于，所述装置还包括：缓存组件查询单元，用于收到应用采用某个缓存组件访问业务数据的请求后，将所述缓存组件中的被请求的业务数据返回给应用。

14.根据权利要求8所述的装置，其特征在于，所述缓存组件包括：ConcurrentMap缓存组件、OSCache缓存组件和/或EHCache缓存组件。

缓存的实现的方法和装置

技术领域

[0001] 本申请涉及网络通信技术领域，尤其涉及一种缓存的实现方法和装置。

背景技术

[0002] 缓存是基于互联网的各种应用普遍使用的一种数据访问技术。基于互联网的应用通常需要通过网络获取各种业务数据，并利用这些业务数据来向用户提供相应的服务。对一些被应用频繁使用的业务数据，可以将这些数据暂时保存在本地，供应用多次读取，从而提高应用的响应速度并降低数据提供方的访问压力。

[0003] Java为开发人员提供了多种不同的缓存技术，每种缓存技术都具有各自的特点和适用场景。在一些较为复杂的应用中，经常采用几个对应于不同缓存技术的缓存组件，来满足应用对业务数据的不同处理需求。同样的业务数据可能存放在不同的缓存组件中，由于每个缓存组件会按照自身的更新机制来更新所缓存的业务数据，可能发生不同缓存组件中相同的业务数据不一致的情况，造成应用的业务处理错误；此外，每个缓存组件在更新相同的业务数据时会分别向数据提供方发起数据获取请求，容易导致对相同数据的多个并发访问请求，造成并发障碍。

发明内容

[0004] 有鉴于此，本申请提供一种缓存的实现方法，所述缓存包括至少两个缓存组件，所述方法包括：

[0005] 在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据；

[0006] 按照应用请求所述业务数据采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中。

[0007] 本申请还提供了一种缓存的实现装置，所述缓存包括至少两个缓存组件，所述装置包括：

[0008] 业务数据获取单元，用于在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据；

[0009] 缓存组件写入单元，用于按照应用请求所述业务数据采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中。

[0010] 由以上技术方案可见，本申请的实施例中，对两个及以上的缓存组件进行统一管理，在满足第一预定条件时向数据提供方获取业务数据，并写入到应用访问该业务数据所采用的一个到多个缓存组件中，使得不同缓存组件中同样的业务数据能够保持一致性，避免了因数据不一致导致的业务处理错误；各个缓存组件中相同的业务数据只需一次读取过程，极大的减少了并发障碍发生的可能性。

附图说明

[0011] 图1是本申请实施例中缓存管理器与应用、缓存组件之间的逻辑关系示意图；

[0012] 图2是本申请实施例中一种缓存的实现方法的流程图；

[0013] 图3是本申请应用示例中缓存管理器的一种逻辑结构图；

[0014] 图4是本申请应用示例中缓存管理器的组成部分在缓存更新过程中的交互流程图；

[0015] 图5是本申请实施例所应用的设备的一种硬件结构图；

[0016] 图6是本申请实施例中一种缓存的实现装置的逻辑结构图。

具体实施方式

[0017] 本申请的实施例提出一种新的缓存的实现方法，在应用与缓存组件之间增加用来对两个或两个以上缓存组件进行统一管理的缓存管理器，应用向缓存管理器请求业务数据，缓存管理器负责向数据提供方获取业务数据，写入到应用指定的至少一个缓存组件中；缓存管理器还可以对各个缓存组件中已缓存业务数据的更新进行统一调度；从而维护各个缓存组件中业务数据的一致性，并且避免各个缓存组件分别向数据提供方请求同样的业务数据，降低了并发障碍发生的可能性，以解决现有技术中存在的问题。

[0018] 需要说明的是，本申请实施例中所说的缓存管理器是在应用和缓存组件之间的一个逻辑部件或者一种逻辑上的层次结构，如图1所示，为描述方便，将其称为缓存管理器。对应用而言，缓存管理器是多个缓存组件的统一接口，应用通过缓存管理器来访问、和/或调用各个缓存组件。在具体实现时，缓存管理器可以独立于应用来实现，作为一个相对独立的组成部分包含在应用中。

[0019] 本申请实施例中，应用可以是任何需要从网络上获取数据的数据需求方，例如，可以是浏览器、应用程序等客户端(其数据提供方通常是作为服务端的应用服务器)，也可以是各种应用服务器等服务端(其数据提供方通常是各种数据库服务器，以及其他的应用服务器)。缓存组件可以是各种已有的实现数据缓存功能的功能单元，也可以是由开发人员自行实现的缓存功能单元。例如在Java环境中，现有技术中的各种Java缓存框架都可以作为缓存组件，如OSCache(一种广泛采用的高性能的Java平台缓存框架)、ConcurrentMap(Java中一种线程安全的缓存数据集)、EHCache(一种Java开源缓存框架)、JSC(Java Caching System，Java缓存系统)、SwarmCache(一种分布式缓存框架)等等。

[0020] 本申请实施例中，缓存管理器所在的设备与数据提供方通过网络相互可访问。其中，缓存管理器所在的设备可以是手机、平板电脑、PC(Personal Computer，个人电脑)、笔记本、虚拟机、物理或逻辑的服务器等。本申请实施例对缓存管理器所在设备的种类，以及该设备与数据提供方之间通信网络的类型、协议等均不做限定。

[0021] 本申请实施例中，采用至少两个缓存组件用来缓存应用所需的业务数据，相同的业务数据可以缓存在数个不同的缓存组件中。缓存的实现方法的流程如图2所示。

[0022] 步骤210，在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据。

[0023] 现有技术中，应用在获取可能会频繁使用的业务数据时，先查询所采用的某个缓存组件中是否保存有所需的业务数据，如果命中则直接使用该缓存组件中的业务数据，否则由应用向该业务数据的数据提供方请求该业务数据。

[0024] 本申请的实施例中，应用向缓存管理器请求需进行缓存的业务数据。哪些业务数据需要进行缓存可以根据实际场景确定，并参照现有技术实现，不再赘述。根据要使用业务

数据的具体方式，应用可能需要采用一个到多个不同的缓存组件来对所需的业务数据进行缓存。应用可以在向缓存管理器请求业务数据的时候指定所采用的一个或者一个以上缓存组件；具体实现中，可以在缓存管理器和应用之间自定义通用于所有缓存组件的业务数据请求接口(将应用采用的缓存组件作为其中的参数)，由缓存管理器将应用的业务数据请求转换为各个缓存组件的数据查询接口；也可以由应用直接使用各个缓存组件的数据查询接口，缓存管理器来根据应用使用的数据查询接口来识别出应用所采用的缓存组件；本申请的实施例不做限定。

[0025] 当缓存管理器收到应用采用某个缓存组件访问业务数据的请求后，在应用指定的缓存组件中查找是否存在被请求的业务数据，如果查找该缓存组件命中，则缓存管理器从该缓存组件中获得被请求的业务数据，并将其返回给应用；如果该缓存组件中不存在被请求的业务数据，则缓存管理器向数据提供方发起请求，来获取被应用请求的业务数据。在从数据提供方得到所请求的业务数据后，缓存管理器按照应用所采用的缓存组件，将所获取的业务数据返回给应用。

[0026] 在一些具体场景中，缓存组件以一定周期，对已缓存的业务数据进行更新，以避免因缓存的业务数据与数据提供方的业务数据不一致导致的问题。本申请实施例中可以由缓存管理器来统一对所有缓存组件中的业务数据进行更新，为所有缓存组件中的业务数据设置缓存更新周期，在缓存更新周期到时，缓存管理器从数据提供方获取需要更新的业务数据。统一更新缓存数据的方式可以根据具体场景的需求来确定，本申请的实施例不做限定。

[0027] 在一种实现方式中，可以由缓存管理器在本地设置一个统一缓存区，用来保存所有缓存组件中已缓存的业务数据以及每个业务数据所在的至少一个缓存组件，这些业务数据具有相同或不同的更新周期。缓存管理器按照业务数据的更新周期来更新统一缓存区中的业务数据，再按照统一缓存区中更新后的业务数据来刷新所有的缓存组件。这种方式中，缓存管理器要维护统一缓存区中业务数据的有效状态(缓存中的数据通常在一段时间未被访问后失效)，如果某个缓存组件中该业务数据为有效状态，则统一缓存区中该业务数据即为有效。

[0028] 在另一种实现方式中，可以为所有的业务数据设置相同的更新周期，当缓存更新周期到时，缓存管理器合并所有缓存组件中需要进行更新的业务数据(即读取所有缓存组件中已缓存的业务数据，并将相同的业务数据进行去重)，记录每个业务数据所在的一个到多个缓存组件(对被两个及以上缓存组件缓存的业务数据，所在的缓存组件超过一个)；然后缓存管理器向数据提供方发起请求，获取每个业务数据的更新值。这种方式中，业务数据的有效状态将由各个缓存组件自行维护，而无需由缓存管理器来维护，实现起来更为简单方便。

[0029] 这样，业务数据的获取，包括各个缓存组件中已缓存业务数据的更新，都由缓存管理器来进行，不会因多个缓存组件向数据提供方请求相同的业务数据而导致并发障碍，提高了应用和数据提供方的性能。

[0030] 步骤220，按照应用请求业务数据所采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中。

[0031] 对应用请求业务数据时采用的缓存组件未命中的情况，缓存管理器从数据提供方得到业务数据后，将得到的业务数据写入到应用请求该业务数据时采用的缓存组件中。

[0032] 对更新各个缓存组件中已缓存业务数据的情况，缓存管理器从数据提供方得到业务数据后，根据所获取的业务数据，对应用访问该业务数据采用的至少一个缓存组件中的该业务数据进行更新。针对上述两种更新的是实现方式而言，缓存管理器可以按照保存的或记录的某个业务数据所在的缓存组件(即应用曾经用来请求该业务数据的缓存组件)，利用获取的业务数据对每个所在缓存组件中的该业务数据进行更新。

[0033] 换言之，本步骤中的写入操作不仅包括在缓存组件中增加业务数据，也包括更新缓存组件中已有的业务数据。

[0034] 在一些场景中，出于性能考虑，会采用多个线程来并发将获取的业务数据写入各个缓存组件中。当这些线程同时访问同一个缓存组件或同一个缓存组件中的同一个业务数据时，容易造成本地的并发障碍。缓存管理器可以在在将所获取的业务数据写入应用访问该业务数据采用的至少一个缓存组件中之前，开启对该缓存组件或该缓存组件中的该业务数据的同步锁；在将所获取的业务数据写入应用访问该业务数据采用的至少一个缓存组件中之后，释放对该缓存组件或该缓存组件中的该业务数据的同步锁。这样，当某个负责写入操作的线程发现其要写入的目的缓存组件中的目的记录已经加了同步锁，表示本次的写入操作已经有其他线程在执行，则不再执行本次的写入操作，从而极大的降低本地并发障碍发生的可能性。

[0035] 在一些场景中，有些业务数据的更新会触发相关的业务过程。因此，本申请实施例中，在符合第二预定条件的业务数据更新完成后，缓存管理器可以通知应用进行与该业务数据相关的业务操作。第二预定条件用来筛选哪些业务数据的更新需要触发业务过程，可以根据实际场景的具体需求确定，例如，可以将更新后需要应用有所响应的业务数据的某个预定标志位置位，则第二预定条件为该预定标志位置位的业务数据。缓存管理器可以通过调用各个缓存组件的监听功能来实现触发应用的相关业务操作，不再赘述。

[0036] 可见，本申请的实施例中，应用不再直接访问和控制多个缓存组件，也不再自行向数据提供方请求需缓存的业务数据，而是向缓存管理器来请求业务数据，由缓存管理器来向数据提供方获取业务数据并统一管理所有的缓存组件，这样各个缓存组件中相同的业务数据只需一次读取过程，极大的减少了并发障碍发生的可能性，同时各个缓存组件中的业务数据都来自于缓存管理器，不同缓存组件中相同的业务数据能够保持一致，避免了因数据不一致导致的业务处理错误。

[0037] 在本申请的一个应用示例中，应用采用了三种Java缓存组件，ConcurrentMap缓存组件、OSCache缓存组件和EHCache缓存组件；该应用的数据提供方为数据库服务器。

[0038] 缓存管理器的结构如图3所示，包括访问与调度器、读取器、更新器和监听器四个组成部分。其中，访问与调度器用来接收应用的业务数据请求，查询缓存组件以及对读取器、更新器和监听器进行调度；读取器用来从数据库服务器获取业务数据；更新器用来根据读取器得到的业务数据更新对应的缓存组件；监听器用来在当某些业务数据的更新应触发后续业务过程时，将这些业务数据的更新通知给应用。

[0039] 应用采用某个缓存组件的数据查询格式，向缓存管理器请求需要缓存的业务数据。访问与调度器收到应用的请求后，通过数据查询格式识别出应用采用的是哪一个缓存组件，并在该缓存组件中查找是否有被请求的业务数据。如果查询该缓存组件命中，则访问与调度器按照该缓存组件的数据响应格式，将应用请求的业务数据返回给应用；如果查询

该缓存组件未命中，则访问与调度器将要获取的业务数据以及对应的缓存组件发送给读取器。

[0040] 读取器从数据库服务器获取该业务数据，并返回给访问与调度器。访问与调度器按照应用采用的缓存组件的数据响应格式，将业务数据返回给应用，并将该业务数据写入到该缓存组件中。

[0041] 在访问与调度器中，为三个缓存组件中缓存的业务数据设置了相同的缓存更新周期。当缓存更新周期到时，缓存管理器的四个组成部分之间的交互流程如图4所示。

[0042] 当缓存更新周期到时，访问与调度器指令读取器启动缓存数据读取。读取器分别采用三个缓存组件提供的数据查询接口，从三个缓存组件中读取所有缓存的业务数据，将其合并为所有需要更新的业务数据清单，并在清单中记录每个业务数据是从哪个或哪几个缓存组件中读取出来的(即该业务数据所在的缓存组件)。读取器向数据库服务器请求清单中每个业务数据的当前值，在获得全部清单中业务数据的当前值后，向访问与调度器返回缓存数据读取完成的消息。

[0043] 访问与调度器开启某个缓存组件的同步锁，指令更新器对该缓存组件中的业务数据进行更新；更新器采用该缓存组件提供的数据更新接口，将读取器获得业务数据的当前值更新到该缓存组件中，更新器可以从读取器生成的业务数据清单中获知某个业务数据应当更新到哪个或哪几个缓存组件中。在更新器更新完该缓存组件中的所有业务数据后，向访问与调度器返回更新完成的消息。访问与调度器释放该缓存组件的同步锁。访问与调度器、更新器重复本过程直到三个缓存组件全部更新完成。

[0044] 访问与调度器将缓存更新完毕的消息发送给监听器。监听器调用各个缓存组件提供的缓存监听接口，以发现各个符合预定条件的业务数据是否有变化，如果有变化则将该业务数据发生变化的消息通知给应用，以便应用启动与该业务数据相关的业务过程。监听器向访问与调度器返回监听执行完毕的消息。至此，本周期的缓存更新完成。

[0045] 与上述流程实现对应，本申请的实施例还提供了一种缓存的实现装置。该装置可以通过软件实现，也可以通过硬件或者软硬件结合的方式实现。以软件实现为例，作为逻辑意义上的装置，是通过该装置所在设备的CPU(Central Process Unit，中央处理器)将对应的计算机程序指令读取到内存中运行形成的。从硬件层面而言，除了图5所示的CPU、内存以及非易失性存储器之外，该装置所在的设备通常还包括用于进行无线信号收发的芯片等其他硬件，或者还包括用于实现网络通信功能的板卡等其他硬件。

[0046] 图6所示为本申请实施例提供的一种缓存的实现装置，该缓存包括至少两个缓存组件，所述装置包括业务数据获取单元和缓存组件写入单元，其中：业务数据获取单元用于在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据；缓存组件写入单元用于按照应用请求所述业务数据采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中。

[0047] 可选的，所述第一预定条件包括：收到应用采用某个缓存组件访问业务数据的请求，且所述缓存组件中不存在被请求的业务数据。

[0048] 可选的，所述装置还包括业务数据返回单元，用于按照应用所采用的缓存组件，将所获取的业务数据返回给应用。

[0049] 一个例子中，所述第一预定条件包括：缓存更新周期到；所述缓存组件写入单元具

体用于根据所获取的业务数据，对应用访问所述业务数据采用的至少一个缓存组件中的所述业务数据进行更新。

[0050] 上述例子中，所述装置还可以包括更新数据合并单元，用于当缓存更新周期到时，合并所有缓存组件中需要进行更新的业务数据，并记录每个业务数据所在的至少一个缓存组件；所述业务数据获取单元具体用于当缓存更新周期到时，从数据提供方获取合并后的每个业务数据；所述缓存组件写入单元具体用于按照记录的每个业务数据所在的缓存组件，利用获取的所述业务数据对所述缓存组件中的所述业务数据进行更新。

[0051] 上述例子中，所述装置还可以包括缓存变化监听单元，用于当符合第二预定条件的业务数据更新完成后，通知应用进行与所述业务数据相关的业务操作。

[0052] 可选的，其特征在于，所述装置还包括同步锁开启单元和同步锁释放单元，其中：同步锁开启单元用于在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之前，开启对所述缓存组件或所述缓存组件中的所述业务数据的同步锁；同步锁释放单元用于在将所获取的业务数据写入应用访问所述业务数据采用的至少一个缓存组件中之后，释放对所述缓存组件或所述缓存组件中的所述业务数据的同步锁。

[0053] 可选的，所述装置还包括缓存组件查询单元，用于收到应用采用某个缓存组件访问业务数据的请求后，将所述缓存组件中的被请求的业务数据返回给应用。

[0054] 可选的，所述缓存组件包括：ConcurrentMap缓存组件、OSCache缓存组件和/或EHCache缓存组件。

[0055] 以上所述仅为本申请的较佳实施例而已，并不用以限制本申请，凡在本申请的精神和原则之内，所做的任何修改、等同替换、改进等，均应包含在本申请保护的范围之内。

[0056] 在一个典型的配置中，计算设备包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

[0057] 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(RAM)和/或非易失性内存等形式，如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

[0058] 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，但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(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(DVD)或其他光学存储、磁盒式磁带，磁带磁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。

[0059] 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。

[0060] 本领域技术人员应明白，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用存储介质(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。

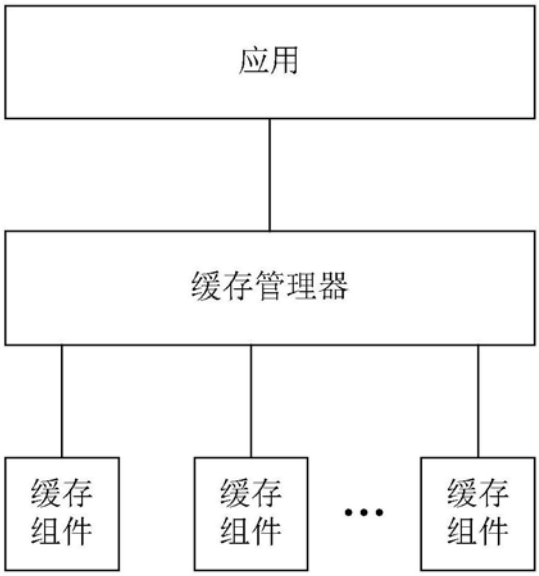

应用
缓存管理器
缓存组件
缓存组件
...
缓存组件

图1

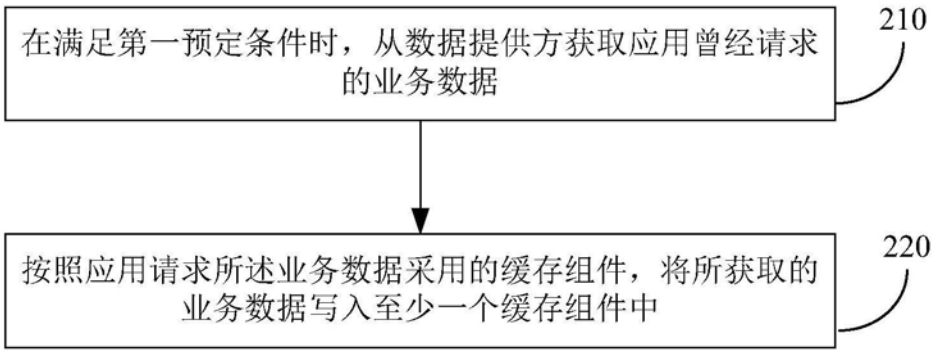

在满足第一预定条件时，从数据提供方获取应用曾经请求的业务数据
210
按照应用请求所述业务数据采用的缓存组件，将所获取的业务数据写入至少一个缓存组件中
220

图2

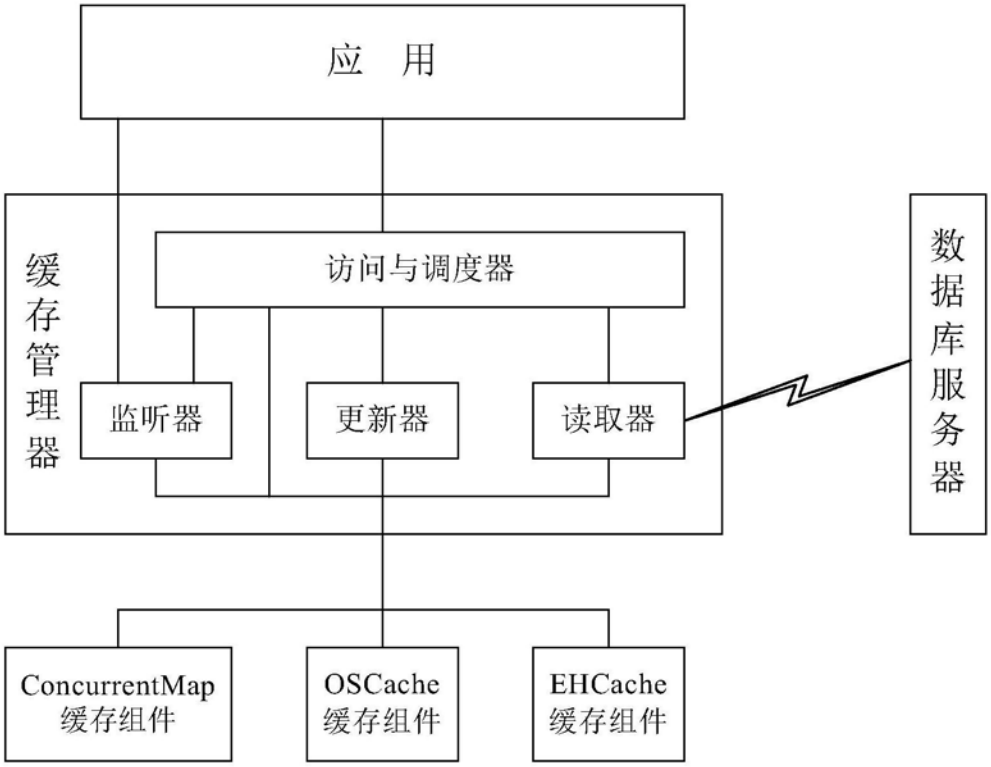

应 用
缓存管理器
访问与调度器
监听器
更新器
读取器
数据库服务器
ConcurrentMap 缓存组件
OSCache 缓存组件
EHCache 缓存组件

图3

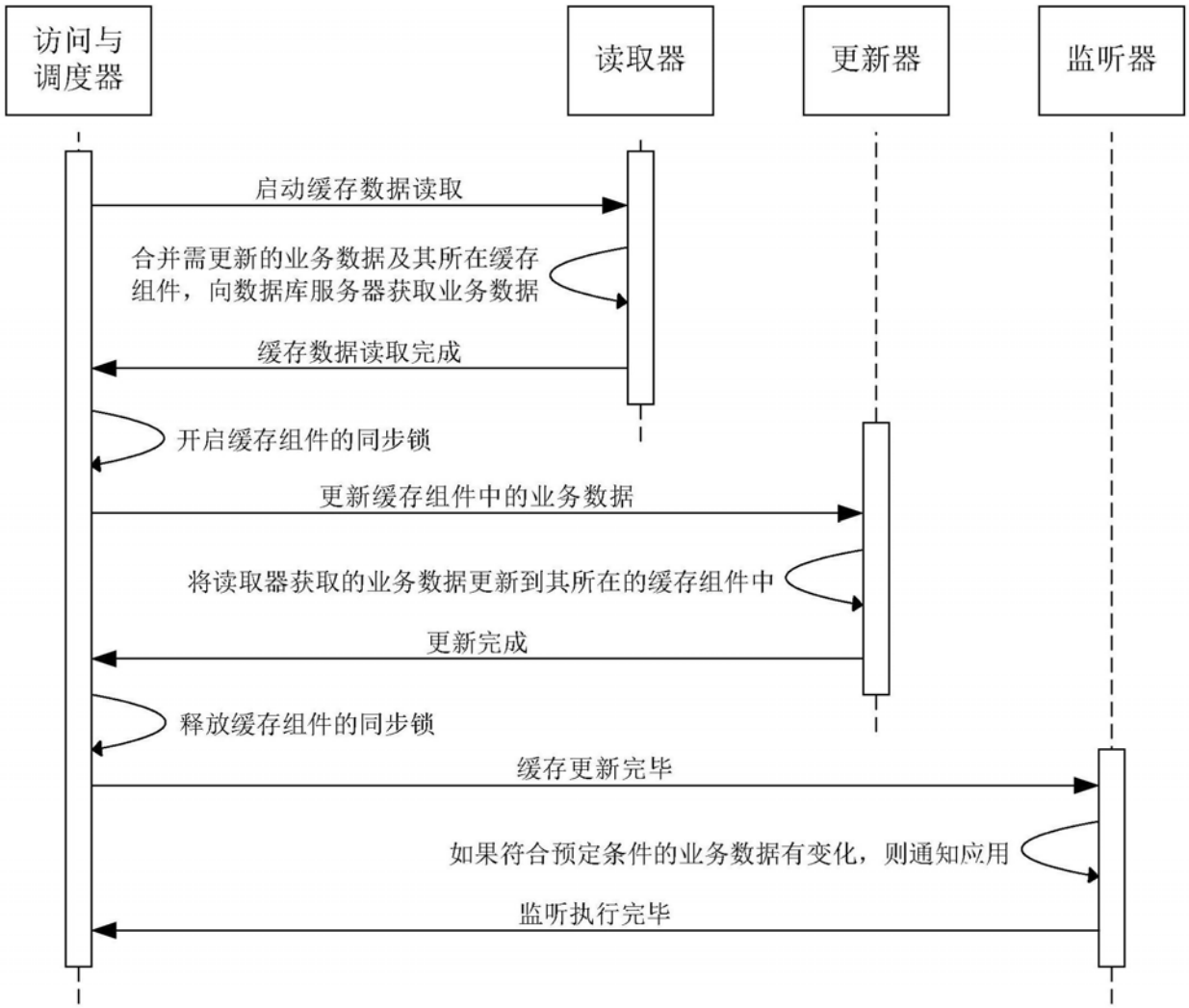

访问与
调度器
读取器
更新器
监听器
启动缓存数据读取
合并需更新的业务数据及其所在缓存
组件，向数据库服务器获取业务数据
缓存数据读取完成
开启缓存组件的同步锁
更新缓存组件中的业务数据
将读取器获取的业务数据更新到其所在的缓存组件中
更新完成
释放缓存组件的同步锁
缓存更新完毕
如果符合预定条件的业务数据有变化，则通知应用
监听执行完毕

图4

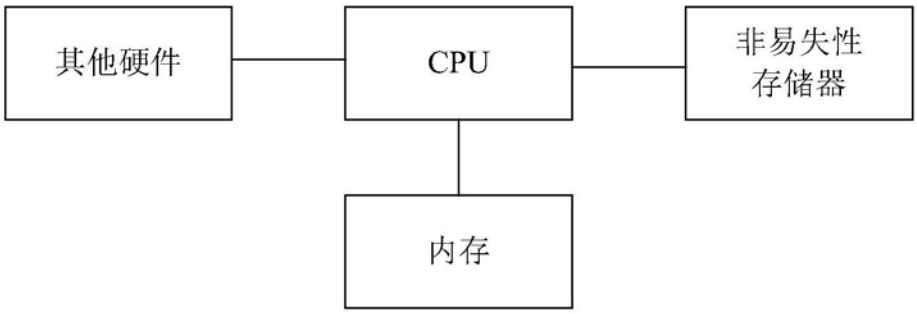

其他硬件
CPU
非易失性
存储器
内存

图5

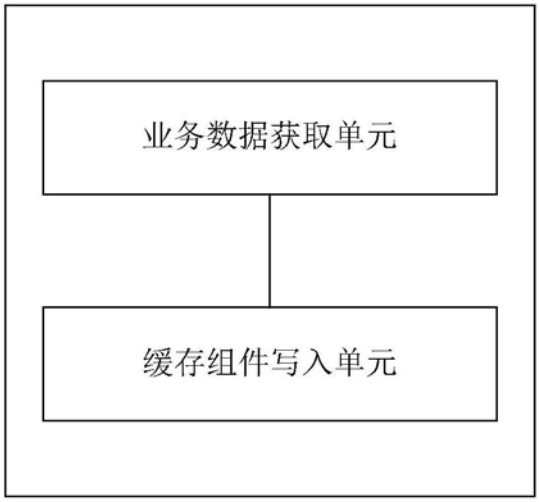


图6