(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 109347906 B
(45)授权公告日 2021.04.20

(21)申请号 201811008654.3
(22)申请日 2018.08.30
(65)同一申请的已公布的文献号
申请公布号 CN 109347906 A
(43)申请公布日 2019.02.15
(73)专利权人 腾讯科技(深圳)有限公司
地址 518057 广东省深圳市南山区高新区科技中一路腾讯大厦35层
(72)发明人 燕皓阳 赵森 苏仙科 曹宝山
(74)专利代理机构 广州三环专利商标代理有限公司 44202
代理人 郝传鑫 贾允
(51)Int.Cl.
H04L 29/08(2006.01)
H04L 12/26(2006.01)
H04L 12/24(2006.01)

(56)对比文件
CN 105512266 A,2016.04.20
CN 106201739 A,2016.12.07
CN 103036961 A,2013.04.10
CN 106789095 A,2017.05.31
CN 105122727 A,2015.12.02
US 2016042043 A1,2016.02.11

审查员 文庆

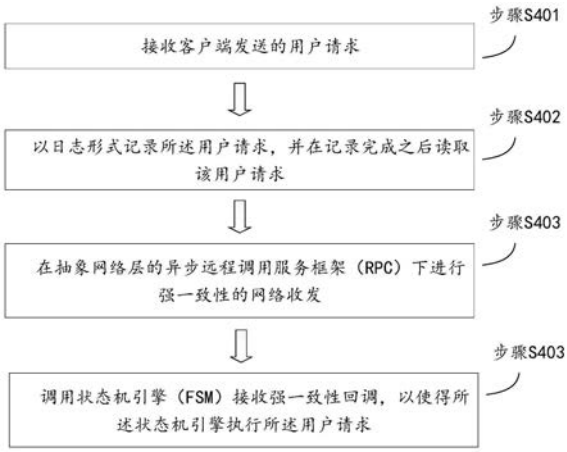
权利要求书2页 说明书9页 附图8页

(54)发明名称

一种数据传输方法、装置、与服务器

(57)摘要

本发明提出一种数据处理方法、装置与服务器，所述方法包括如下步骤：接收客户端发送的用户请求；以日志形式记录所述用户请求，并在记录完成之后读取该日志；在抽象网络层的异步远程调用服务框架下进行强一致性的网络收发；调用状态机引擎接收强一致性回调，以使得所述状态机引擎执行所述用户请求。



CN 109347906 B

1.一种数据处理方法,其特征在于,应用于公共强一致性算法模块,所述公共强一致性算法模块包括主节点和从节点,所述方法包括如下步骤:

接收客户端发送的用户请求;

以日志形式记录所述用户请求,并在记录完成之后读取该日志;

将基于异步远程调用服务框架的异步磁盘IO、网络IO及状态机引擎的异步实现封装成同步接口;

在抽象网络层的异步远程调用服务框架(RPC)下,通过回调所述同步接口进行强一致性的网络收发;

当主节点接收到大多数从节点的成功落盘通知时,调用状态机引擎(FSM)接收强一致性回调,以使得所述状态机引擎执行所述用户请求。

2.根据权利要求1所述的方法,其特征在于,所述接收客户端发送的用户请求包括:由主节点接收来自客户端的用户请求,当从节点接收到来自客户端的请求时,拒绝所述请求,并向客户端返回所述主节点的网络地址。

3.根据权利要求1所述的方法,其特征在于,以日志形式记录所述用户请求,并在记录完成之后读取该日志包括:

接收用户请求;

将所述用户请求序列化为请求日志;

向磁盘读写逻辑发送日志落盘请求;

接收磁盘读写逻辑返回的信息,当落盘完成时,读出该落盘日志。

4.根据权利要求1所述的方法,其特征在于,在抽象网络层的异步远程调用服务框架(RPC)下进行强一致性的网络收发包括:

主节点将读出的日志同步给从节点;

所述从节点对所述日志进行落盘操作,并在成功落盘时,通知所述主节点。

5.根据权利要求4所述的方法,其特征在于,所述公共强一致性算法模块还包括候选节点,所述主节点将读出的日志同步给从节点步骤包括:

主节点向从节点发送心跳包;当没有需要同步的日志时,心跳包的内容为空;当需要同步日志时,则带上需要同步的日志;

主节点判断从节点的超时情况以进行容灾操作;

从节点根据收到心跳包的间隔判断主节点是否超时;当主节点超时,从节点则变为候选节点,并发起选举申请。

6.根据权利要求1所述的方法,其特征在于,所述调用状态机引擎(FSM)接收强一致性回调,以使得所述状态机引擎执行所述用户请求包括:

当主节点接收到大多数从节点的成功写入日志文件通知时,调用状态机引擎(FSM),并通知所述状态机引擎(FSM)对所述日志进行写操作,然后将结果返回给客户端。

7.根据权利要求1所述的方法,其特征在于,所述状态机引擎执行所述用户请求之后,将执行结果返回所述客户端。

8.根据权利要求1所述的方法,其特征在于,在抽象网络层的异步远程调用服务框架(RPC)下进行强一致性的网络收发过程中,等待单个线程返回的参数状态,当参数状态变为完成后,继续执行异步远程调用服务框架(RPC)的其他线程。

9.一种数据处理装置,其特征在于,应用于公共强一致性算法模块,所述公共强一致性算法模块包括主节点和从节点,所述方法包括如下步骤:

接收模块,用于接收客户端发送的用户请求;

日志读取模块,用于以日志形式记录所述用户请求,并在记录完成之后读取该日志;

封装模块,用于将基于异步远程调用服务框架的异步磁盘IO、网络IO及状态机引擎的异步实现封装成同步接口;

强一致收发模块,用于在抽象网络层的异步远程调用服务框架(RPC)下,通过回调所述同步接口进行强一致性的网络收发;

调用模块,用于当主节点接收到大多数从节点的成功落盘通知时,调用状态机引擎(FSM)接收强一致性回调,以使得所述状态机引擎执行所述用户请求。

10.一种服务器,其特征在于,所述服务器包含权利要求9所述的装置。

11.一种存储介质,其特征在于,所述存储介质中存储有至少一条指令、至少一段程序、代码集或指令集,所述至少一条指令、所述至少一段程序、所述代码集或指令集由处理器加载并执行以实现所述权利要求1-8任一的数据处理方法。

一种数据传输方法、装置、与服务器

技术领域

[0001] 本发明涉及数据库技术领域，特别涉及一种数据处理方法、装置与服务器。

背景技术

[0002] Redis的主从复制采用异步复制方法，从服务器周期性向主服务器报告复制流的处理进度。

[0003] Redis是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set--有序集合)和hash(哈希类型)。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是人Redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了Master-slave(主从)同步。

[0004] Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

[0005] 然而，Redis没有提供强一致同步模式，无法覆盖银行、保险等金融级应用场景中。

发明内容

[0006] 为了解决现有技术中存在的技术问题，本发明实施例提供了一种数据处理方法、装置、服务器与存储介质。所述技术方案如下：

[0007] 一方面，提供一种数据处理方法，包括：接收客户端发送的用户请求；以日志形式记录所述用户请求，并在记录完成之后读取该日志；在抽象网络层的异步远程调用服务框架下进行强一致性的网络收发；调用状态机引擎接收强一致性回调，以使得所述状态机引擎执行所述用户请求。

[0008] 一方面，提供一种数据处理装置，包括：接收模块，用于接收客户端发送的用户请求；日志读取模块，用于以日志形式记录所述用户请求，并在记录完成之后读取该日志；强一致收发模块，用于在抽象网络层的异步远程调用服务框架下进行强一致性的网络收发；调用模块，用于调用状态机引擎接收强一致性回调，以使得所述状态机引擎执行所述用户请求。

[0009] 另一方面，提供一种服务器，所述服务器包含前述的装置。

[0010] 另一方面，提供一种存储介质，所述存储介质中存储有至少一条指令、至少一段程序、代码集或指令集，所述至少一条指令、所述至少一段程序、所述代码集或指令集由处理器加载并执行以实现前述的数据处理方法。

[0011] 本发明实施例提供的技术方案带来的有益效果包括:通过强一致性收发可以保证数据处理的高安全性和高可靠性,保证数据处理过程可以用于银行、保险等金融级应用场景,以及其它需要严格安全等级的场景。

附图说明

[0012] 为了更清楚地说明本发明实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,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。

[0013] 图1是本发明实施例提供的一种分布式数据管理单元示意图;

[0014] 图2是本发明实施例提供的采用公共强一致性算法和日志库的网络构架示意图;

[0015] 图3是本发明实施例提供的与采用公共强一致性算法和日志库的网络构架示意图对应的分布式数据管理单元示意图;

[0016] 图4是本发明实施例提供的数据处理方法步骤流程示意图;

[0017] 图5是本发明实施例提供的数据处理处理方法的交互过程示意图;

[0018] 图6是本发明实施例提供的心跳包发送路径示意图;

[0019] 图7是本发明实施例提供的心跳包发送交互过程示意图;

[0020] 图8是本发明实施例提供的节点生命周期示意图;

[0021] 图9是本发明实施例提供的数据处理装置原理框图;

[0022] 图10是本发明实施例提供的接收模块原理框图;

[0023] 图11是本发明实施例提供的日志读取模块原理框图;

[0024] 图12是本发明实施例提供的强一致收发模块原理框图;

[0025] 图13是本发明实施例提供的主从同步模块原理框图;

[0026] 图14是本发明实施例提供的调用模块的原理框图;

[0027] 图15是本发明实施例提供的包含结果返回模块的数据处理装置原理框图;

[0028] 图16是本发明实施例提供的强一致收发模块的其他可选子模块原理框图;

[0029] 图17是本发明实施例提供的服务器结构示意图。

具体实施方式

[0030] 为使本发明的目的、技术方案和优点更加清楚,下面将结合附图对本发明实施方式作进一步地详细描述。

[0031] 在本发明的一个实施例中,如图1所示,提出一种分布式数据管理单元,每个数据管理单元称为分片(shard),图1展示的是一台机器上的分片情况。每个业务ID(Bid)包含一个副本集(replicat set),一个副本集中又包含若干个分片(shard),每个分片管理一定大小的共享内存块,底层引擎把这些内存快进行组织管理已实现Redis各接口的功能。由分片ID(shard id)可以计算出分片所属服务所在的CPU ID。每个副本集的分片原则上分布于不同的机器,即若此时整个副本集中有3台机器A、B、C,创建一个Bid时,分别在A、B、C各创建一个分片,并指定主备角色后,便可以接收用户请求并服务了。

[0032] 根据以上的描述,图1中不同的分片第一分片(shard-0)、第二分片(shard-1)、第

三分片(shard-2)、第四分片(shard-3)分属于不同的业务,而用户请求是随机发送到任意CPU的,这时接收用户请求的CPU所属的服务若通过查询路由发现自己并不是目标分片,则把请求通过递交函数(submit)发送到目标CPU,由目标CPU上的服务进行处理。每个分片还对应各自的共享内存第一共享内存(shm-0)、第二共享内存(shm-1)、第三共享内存(shm-2)、第四共享内存(shm-3),共享内存用于存储关键内容(key value)。以下是转发的请求代码片段:

[0033] Return submit(ctx->CPU(),[this,ctx]){

[0034] Process(ctx);

[0035] })

[0036] 在一些金融业务场景中,对于数据各副本集的一致性要求很高,图1所示的构架还无法完全保持副本之间的强一致特性。

[0037] 在一个实施例中,如图2所示,为了保证兼容性,没有直接基于异步远程调用服务框架(RPC)来实现日志复制和一致性算法,而是采用内构公共强一致性算法和日志库(libraft)来实现日志复制和一致性算法协议中的基本逻辑和接口调用。公共强一致性算法和日志库(libraft)提供常规的同步接口,由于异步远程调用服务框架无法直接调用阻塞式的磁盘IO和网络IO,所以强一致逻辑的实现难点主要在于异步框架和同步接口的结合。因此,在强一致的逻辑中,基于异步远程调用服务框架(RPC)的异步磁盘IO、网络IO及状态机引擎,并把这些异步实现封装成同步接口供公共强一致性算法和日志库(libraft)回调,这样可以实现主从分片的强一致复制。

[0038] 与图2所对应,图3展示了一种分布式数据管理单元,结合图2与图3,在分片的相同级别增加一个强一致性分片对象。强一致性分片对象可以被视为一个强一致性传输协议中的节点(Node),具有强一致性传输协议中单个节点的完整功能,在每个强一致性分片包中包含一个公共强一致性算法和日志库(libraft),包含强一致性基本逻辑的同步接口、磁盘和网络IO逻辑、状态机引擎(FSM)实现、供公共强一致性算法和日志库(libraft)回调的接口。每个强一致性传输协议中的节点(Node)具有自己的角色,处于主节点、候选节点、从节点中的一个。主节点接受用户读写请求,从节点只接受读请求,从在收到写请求后会把请求通过异步远程调用服务框架(RPC)的方式发送给主节点。主节点收到读请求后,先把请求序列化成一条日志进行落盘操作,接着会把已经落盘的日志从磁盘中读出,通过网络IO把这条日志同步给其他从节点,其他从节点收到日志同步的请求后会进行落盘,当这条日志已经成功落盘,从节点会通知主节点。最后,主节点若发现这条日志已经被多数节点成功写入日志文件,就会调用底层引擎,同时也会通知从节点调用底层引擎,应用这条日志所对应的写操作,然后将结果返回给用户。

[0039] 在上述过程中,用户请求首先到达强一致性传输协议中的节点(Node),在强一致性节点之间进行流转,确认日志已经到达大多节点数后,才调用底层引擎。

[0040] 用户请求在单个强一致性分片中的请求路径如图2所示:

[0041] 客户端(Client),从网络上收到请求后,封包发送至强一致性模块;

[0042] 定时器(Timer),定期驱动公共强一致性模块,以响应超时事件;

[0043] 抽象网络层(Network),提供异步远程调用服务框架(RPC)的网络收发功能,并驱动强一致性驱动模块;

[0044] 状态机引擎(FSM),接受来自强一致性的回调,将用户请求应用到实际引擎上,并返回用户结果;

[0045] 公共强一致性算法和日志库(libraft),实现了一致性算法和日志库协议的同步接口,由远程调用服务框架驱动;

[0046] 磁盘读写逻辑(DiskLog),处理来自公共强一致性算法和日志库的日志读写请求。

[0047] 用户请求与回复流程如图4所示,图4中的抽象网络层(Network)、状态机引擎(FSM)、磁盘读写逻辑(DiskLog)都是把异步远程调用服务框架(RPC)的异步接口封装成同步接口提供给公共强一致性算法和日志库(libraft)调用,当异步远程调用服务框架(RPC)需要调用公共强一致性算法和日志库(libraft)的同步接口时,比如当有用户请求进来,就需要在异步远程调用服务框架(RPC)中调用公共强一致性算法和日志库(libraft)中的特定接口来接收用户请求,例如用户数据接收接口(RecvUserData)。具体地,图4描述了如下数据处理步骤:

[0048] 步骤S401,接收客户端发送的用户请求。

[0049] 客户端接收用户请求,将所述请求封包后发送至公共强一致性算法模块。公共强一致性算法模块包含三种角色的节点:主节点、从节点和候选节点,由主节点接收来自客户端的封包。

[0050] 步骤S402,以日志形式记录所述用户请求,并在记录完成之后读取该用户请求;

[0051] 公共强一致性算法模块中的主节点收到读请求后,先把请求序列化成一条日志,然后向磁盘读写逻辑发送日志落盘请求,在落盘完成之后,主节点将该落盘的日志读出。

[0052] 步骤S403,在抽象网络层的异步远程调用服务框架(RPC)下进行强一致性的网络收发。

[0053] 公共强一致性算法模块通过网络IO将读出的日志同步给其他从节点,其他从节点在接收到日志同步请求之后,会对请求进行落盘操作,如果日志在成功落盘,从节点会通知主节点。

[0054] 步骤S404,调用状态机引擎(FSM)接收强一致性回调,以使得所述状态机引擎执行所述用户请求。

[0055] 当主节点发现日志已经被大多数节点成功写入日志文件,就会调用底层的状态机引擎,同时也会通知从节点调用底层引擎,应用这条日志所对应的写操作,然后将结果返回给用户。

[0056] 综上所述,强一致性可以带来高安全性和高可靠性,保证数据处理过程可以用于银行、保险等金融级应用场景,以及其它需要严格安全等级的场景。图5描述了上述步骤的交互流程。

[0057] 在异步远程调用服务框架(RPC)中,唯一可以调用同步接口的地方是在RPC框架线程中,否则会导致程序异常终止(coredump)。所以在接收用户请求的场景中,启动一个异步远程调用服务框架(RPC)线程,调用公共强一致性算法和日志库(libraft)中的特定接口,例如RecvUserData()。各异步远程调用服务框架(RPC)线程在创建它的CPU上运行,并且不共享栈,所以并不需要锁来保证线程安全。类似以上的异步远程调用服务框架(RPC)线程在强一致的实现过程中有若干个,无论是异步远程调用服务框架(RPC)调用公共强一致性算法和日志库(libraft)的同步接口还是公共强一致性算法和日志库(libraft)的回调都是

在其中进行的。例如,可以将future接口封装成同步接口,以满足异步接口的同步封装:

[0058] Future<>DisklogImp_::DisklogImpl_::AppendEntry(const Entry&entry)

[0059] Return gate(_gate,[this,entry]){

[0060] Return lock(_lock,for_write(),[this,entry]){

[0061] Void DisklogImp_::DisklogImpl_::AppendEntry(const Entry&entry){

[0062] _impl.AppendEntry(entry).get();

[0063] }

[0064] 以上是把一个异步方法封装成同步接口,这里的get()函数等待异步的AppendEntry()返回的future变成完成状态。当future状态变为完成后,这个CPU上的其他RPC框架线程才能够继续执行。可见,经过上述步骤,可以将异步执行框架的接口进行同步封装。

[0065] 强一致性公共库对象(libraft)中,服务器被赋予三种角色:主节点、从节点和候选节点。在这三种节点中,主节点的地位是核心的,其用于接收来自客户端的请求,客户端不会将请求发送到从节点,即使客户端将请求发送到从节点,从节点会拒绝客户端的请求,并将主节点的IP地址返回给客户端。主节点和从节点之间的数据是单向流动的,数据以心跳包的形式由主节点流向从节点,心跳包中包含有日志文件。从节点接收来自主节点的心跳包,若一个从节点在规定的时间内没有收到主节点的心跳包,则该节点会转变为候选节点,并向其他从节点发起投票请求,当大多数从节点投票给该候选节点时,该候选节点转变为主节点。作为候选节点,不接受主节点发送的心跳包。

[0066] 在计时器(Timer)的驱动下,主节点会给各从节点发送心跳包,当没有需要同步的日志时,心跳包的内容为空,反之则带上需要同步的日志。主节点会判断从节点的超时情况来做一些容灾操作,从节点则会根据收到心跳包的间隔判断主节点是否超时,若主节点超时则自己变为候选节点发起选举。心跳包发送路径如图6所示:

[0067] 步骤S601,主节点向从节点发送心跳包;当没有需要同步的日志时,心跳包的内容为空;当需要同步日志时,则带上需要同步的日志。

[0068] 步骤S602,主节点判断从节点的超时情况以进行容灾操作。

[0069] 步骤S603,从节点根据收到心跳包的间隔判断主节点是否超时;当主节点时,从节点则变为候选节点发起选举申请。

[0070] 图7则展示了心跳包发送的交互过程。

[0071] 在一个具体的实施例中,从节点的定时器周期性地调用运行时间模块(run_periodic),该模块会检查上次主节点发送过来的心跳时间,若长时间未收到主节点的心跳(超过时间阈值),从节点就会将自己变成候选节点并发起选举。候选节点通过当前写入期限模块和当前投票因素模块来回调本地磁盘逻辑中的持久化选举元信息。候选节点可以通过读取最后目录信息模块和读取期限模块回调磁盘逻辑以获得自身的期限和索引。期限和索引用于选举请求,选举过程中其他节点根据选举节点的期限和索引决定是否同意投票。候选节点通过发送指令回调抽象网络层向集群中的其他节点发起选举,候选节点通过接收函数回调抽象网络接收选举请求的回包。如图8所示,候选节点是有时间限制的,这个实现限制反映为候选节点自身的期限。候选节点的生命周期包括选举期、分裂投票期和一般操作期。选举期内,候选节点向从节点发起邀票请求;分裂投票期间,候选节点接收来自从节

点的投票;一般操作期则是根据索引和节点期限判断该候选节点是否成为主节点,此外,如果同时有两个候选节点发起投票,那么哪个候选节点作为主节点由二者竞争实现。两个候选节点在一段timeout比如300ms等待以后,因为双方得到的票数是一样的,那么在300ms以后,再由这两个候选者发出邀票,这时同时得票的概率大大降低,那么首先发出邀票的候选者得到了大多数同意,成为主节点,而另外一个候选节点后来发出邀票时,那些从节点已经投票给第一个候选节点,不能再投票给它,它就成为落选者了,最后这个落选者也成为普通从节点一员了。

[0072] 在一个可能的实施例中,就用户数据接收接口而言,考虑下面的场景,服务器向客户端发送数据“_META_DATA_\r\n_USER_DATA”,要求"\r\n"之前的数据_META_DATA_在第一次请求中接收,剩下的请求调用读取_USER_DATA_部分的数据。因为tcp协议是流协议(Stream),并且_META_DATA_数据不是定长的,所以没有办法保证一次请求调用不读到_USER_DATA_部分的数据,除非一次读取一个字符。此时,可以考虑请求函数中的数据峰参数(MSG_PEEK),请求的原型是ssize_t Recv(int s,void*buf,size_t len,int flags);通常flags都设置为0,此时请求函数读取tcp缓存中的数据到数据缓存中,并从tcp缓存中移除已读取的数据。把flags设置为数据峰参数(MSG_PEEK),仅把tcp缓存中的数据读取到缓存中,并不把已读取的数据从tcp缓存中移除,再次调用Recv仍然可以读到刚才读到的数据。针对上面的场景,Recv(fd,buf,nbuf,MSG_PEEK)查看数据,查看"\r\n"的位置pos,再Recv(fd,buf,pos+2,0)读取(并移除)数据。当然,这样的情况比较极端,很多时候,就算在一次Recv中读到了_USER_DATA_部分的数据,仍可以先把这部分数据保存起来,添加到后续的Recv数据之前,但是如果不同的Recv跨越很多函数,保存数据带来了额外的复杂度。

[0073] 在一个可能的实施例中,还可以考虑另外的场景,同一个端口支持文本协议和二进制协议,利用数据峰参数(MSG_PEEK)看一下头几个字符,先判断是文本协议还是二进制协议,再做请求分发,也是不错的选择。当然,真正的Recv之前调用使用数据峰参数(MSG_PEEK)导致额外的一次函数调用。

[0074] 在一个可能的实施例中,在异步远程调用服务框架(RPC)中,唯一可以调用同步接口的地方是在框架的线程中,否则会导致程序异常终止(coredump)。各异步远程调用服务框架(RPC)线程在创建它的CPU上运行,并且不共享栈,所以并不需要锁来保证线程安全。由于异步远程调用服务框架(RPC)中的各个线程之间并不共享栈,各个线程之间构成了类似通道的(channel)关系。在一般模式下,如果串行执行两次循环函数(loop)会产生连续的两次输出,例如:

[0075] func loop(){

[0076] for i:=0;i<10;i++{

[0077] fmt.Printf("%d",i)

[0078] }}

[0079] func main(){

[0080] loop()

[0081] loop()

[0082] }

[0083] 那么执行结果是01234567890123456789。

[0084] 而在异步远程调用服务框架(RPC)中,如果串行执行两次循环函数(loop),则只会输出一次结果0123456789。这是因为在异步框架下,第二次循环还没来得及执行,主函数就已经退出。因此,在异步框架下,为了防止主函数过早的退出,采用等待的方法,即在loop()函数中增加等待时间:

[0085] func main(){

[0086] loop();

[0087] loop();

[0088] time.sleep(time.second);

[0089] }

[0090] 但是增加等待时间的方式无疑会增加整个的执行时间消费,所以可以在线程间增加通知函数,而为另外一个线程增加阻塞:

[0091] for thread in threads;

[0092] thread.join();

[0093] 只有当信道执行完成之后,才会通知消除阻塞,而这种方式有带来死锁的可能。因此,将异步远程调用服务框架(RPC)作为唯一可以调用同步接口的地方,可以减少由于其他调用造成的异常终止的可能。

[0094] 在本发明的一个实施例中,如图9所示,提供一种数据处理装置的原理框图,所述装置包括:接收模块,用于接收客户端发送的用户请求;日志读取模块,用于以日志形式记录所述用户请求,并在记录完成之后读取该日志;强一致收发模块,用于在抽象网络层的异步远程调用服务框架(RPC)下进行强一致性的网络收发;调用模块,用于调用状态机引擎(FSM)接收强一致性回调,以使得所述状态机引擎执行所述用户请求。

[0095] 在一个可选的实施例中,如图10所示,提供接收模块的原理框图,包括:接收子模块,用于主节点接收到来自客户端的用户请求时,接收来自客户端的用户请求;拒收子模块,用于从节点接收到来自客户端的请求时,拒绝所述请求,并向客户端返回所述主节点的网络地址。

[0096] 在一个可选的实施例中,如图11所示,提供日志读取模块的原理框图,包括:用户请求接收子模块,用于接收用户请求;序列化子模块,用于将所述用户请求序列化为请求日志;落盘请求子模块,用于向磁盘读写逻辑发送日志落盘请求;读出子模块,用于接收磁盘读写逻辑返回的信息,当落盘完成时,读出该落盘日志。

[0097] 在一个可选的实施例中,如图12所示,提供强一致收发模块的原理框图,包括:主从同步子模块,用于将主节点将读出的日志同步给从节点;落盘通知子模块,用于所述从节点对所述日志进行落盘操作,并在成功落盘时,通知所述主节点。

[0098] 在一个可选的实施例中,如图13所示,提供主从同步子模块的原理框图,包括:心跳包发送子模块,用于主节点向从节点发送心跳包;当没有需要同步的日志时,心跳包的内容为空;当需要同步日志时,则带上需要同步的日志;主节点超时判断子模块,用于主节点判断从节点的超时情况以进行容灾操作;从节点超时判断子模块,从节点根据收到心跳包的间隔判断主节点是否超时;当主节点超时,从节点则变为候选节点,并发起选举申请。

[0099] 在一个可选的实施例中,如图14所示,提供调用模块的原理框图,包括:调用子模块,用于当主节点接收到大多数从节点的成功写入日志文件通知时,调用状态机引擎

(FSM)，并通知所述状态机引擎(FSM)对所述日志进行写操作；结果返回子模块，用于将结果返回给客户端。

[0100] 在一个可选的实施例中，如图15所示，提供包含结果返回模块的数据处理装置原理框图：结果返回模块用于在所述状态机引擎执行所述用户请求之后，将执行结果返回所述客户端。

[0101] 在一个可选的实施例中，如图16所示，提供强一致收发模块的其他可选子模块，包括：线程等待模块，用于在抽象网络层的异步远程调用服务框架(RPC)下进行强一致性的网络收发过程中，等待单个线程返回的参数状态；参数状态监控模块，用于监控参数状态；同步封装模块，用于当参数状态变为完成后，继续执行异步远程调用服务框架(RPC)的其他线程。通过以上过程，可以将异步框架中的接口封装为同步

[0102] 请参考图17，其示出了本发明一个实施例提供的服务器的结构示意图。该服务器用于实施上述实施例中提供的服务器侧的数据处理方法。具体来讲：

[0103] 所述服务器1200包括中央处理单元(CPU)1201、包括随机存取存储器(RAM)1202和只读存储器(ROM)1203的系统存储器1204，以及连接系统存储器1204和中央处理单元1201的系统总线1205。所述服务器1200还包括帮助计算机内的各个器件之间传输信息的基本输入/输出系统(I/O系统)1206，和用于存储操作系统1213、应用程序1214和其他程序模块1215的大容量存储设备1207。

[0104] 所述基本输入/输出系统1206包括有用于显示信息的显示器1208和用于用户输入信息的诸如鼠标、键盘之类的输入设备1209。其中所述显示器1208和输入设备1209都通过连接到系统总线1205的输入输出控制器1210连接到中央处理单元1201。所述基本输入/输出系统1206还可以包括输入输出控制器1210以用于接收和处理来自键盘、鼠标、或电子触控笔等多个其他设备的输入。类似地，输入输出控制器1210还提供输出到显示屏、打印机或其他类型的输出设备。

[0105] 所述大容量存储设备1207通过连接到系统总线1205的大容量存储控制器(未示出)连接到中央处理单元1201。所述大容量存储设备1207及其相关联的计算机可读介质为服务器1200提供非易失性存储。也就是说，所述大容量存储设备1207可以包括诸如硬盘或者CD-ROM驱动器之类的计算机可读介质(未示出)。

[0106] 不失一般性，所述计算机可读介质可以包括计算机存储介质和通信介质。计算机存储介质包括以用于存储诸如计算机可读指令、数据结构、程序模块或其他数据等信息的任何方法或技术实现的易失性和非易失性、可移动和不可移动介质。计算机存储介质包括RAM、ROM、EPROM、EEPROM、闪存或其他固态存储其技术，CD-ROM、DVD或其他光学存储、磁带盒、磁带、磁盘存储或其他磁性存储设备。当然，本领域技术人员可知所述计算机存储介质不局限于上述几种。上述的系统存储器1204和大容量存储设备1207可以统称为存储器。

[0107] 根据本发明的各种实施例，所述服务器1200还可以通过诸如因特网等网络连接到网络上的远程计算机运行。也即服务器1200可以通过连接在所述系统总线1205上的网络接口单元1211连接到网络1212，或者说，也可以使用网络接口单元1211来连接到其他类型的网络或远程计算机系统(未示出)。

[0108] 所述存储器还包括一个或者一个以上的程序，所述一个或者一个以上程序存储于存储器中，且经配置以由一个或者一个以上处理器执行。上述一个或者一个以上程序包含

用于执行上述后台服务器侧的方法的指令。

[0109] 在示例性实施例中，还提供了一种包括指令的非临时性计算机可读存储介质，例如包括指令的存储器，上述指令可由终端的处理器执行以完成上述方法实施例中发送方客户端或接收方客户端侧的各个步骤，或者上述指令由服务器的处理器执行以完成上述方法实施例中后台服务器侧的各个步骤。例如，所述非临时性计算机可读存储介质可以是ROM、随机存取存储器(RAM)、CD-ROM、磁带、软盘和光数据存储设备等。

[0110] 应当理解的是，在本文中提及的“多个”是指两个或两个以上。“和/或”，描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。字符“/”一般表示前后关联对象是一种“或”的关系。

[0111] 上述本发明实施例序号仅仅为了描述，不代表实施例的优劣。

[0112] 本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成，也可以通过程序来指令相关的硬件完成，所述的程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，磁盘或光盘等。

[0113] 以上所述仅为本发明的较佳实施例，并不用以限制本发明，凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

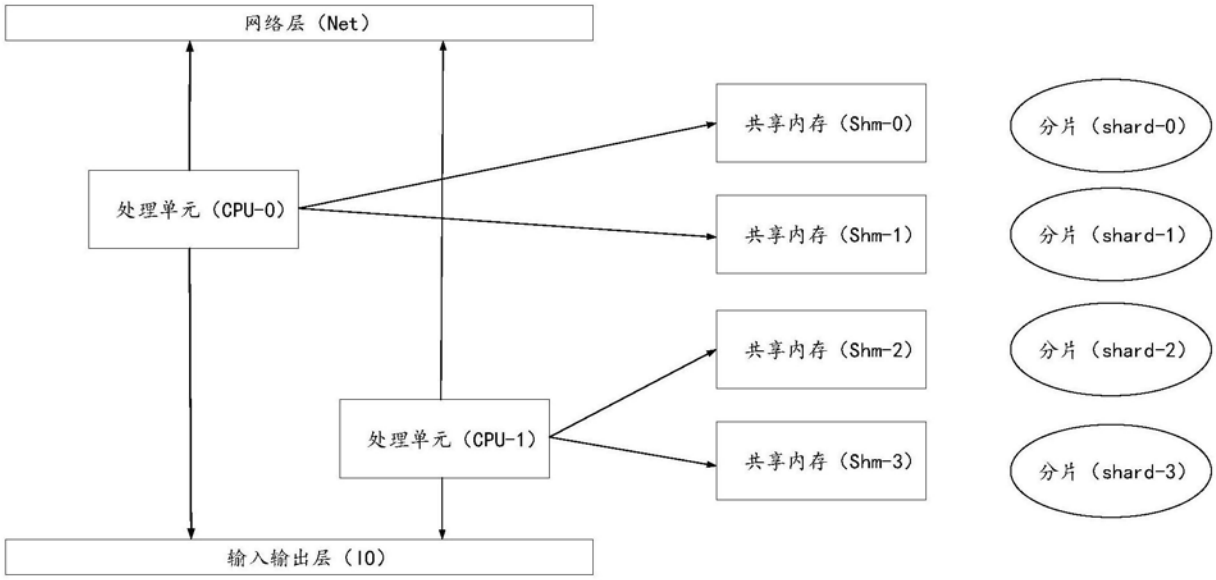

网络层（Net）
处理单元（CPU-0）
处理单元（CPU-1）
输入输出层（IO）
共享内存（Shm-0）
共享内存（Shm-1）
共享内存（Shm-2）
共享内存（Shm-3）
分片（shard-0）
分片（shard-1）
分片（shard-2）
分片（shard-3）

图1

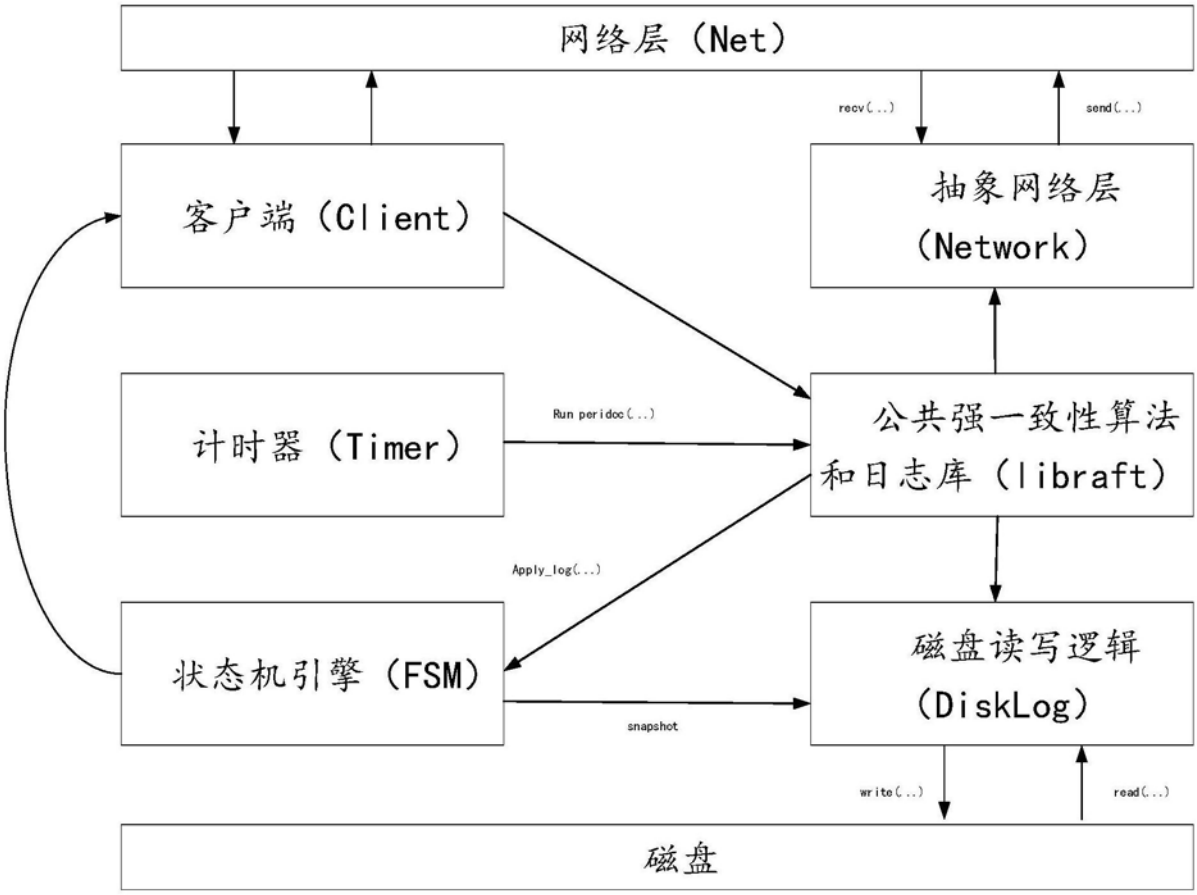

网络层（Net）
recv(...)
send(...)
客户端（Client）
抽象网络层
（Network）
计时器（Timer）
Run peridoc(...)
公共强一致性算法
和日志库（libraft）
Apply_log(...)
状态机引擎（FSM）
磁盘读写逻辑
（DiskLog）
snapshot
write(...)
read(...)
磁盘

图2

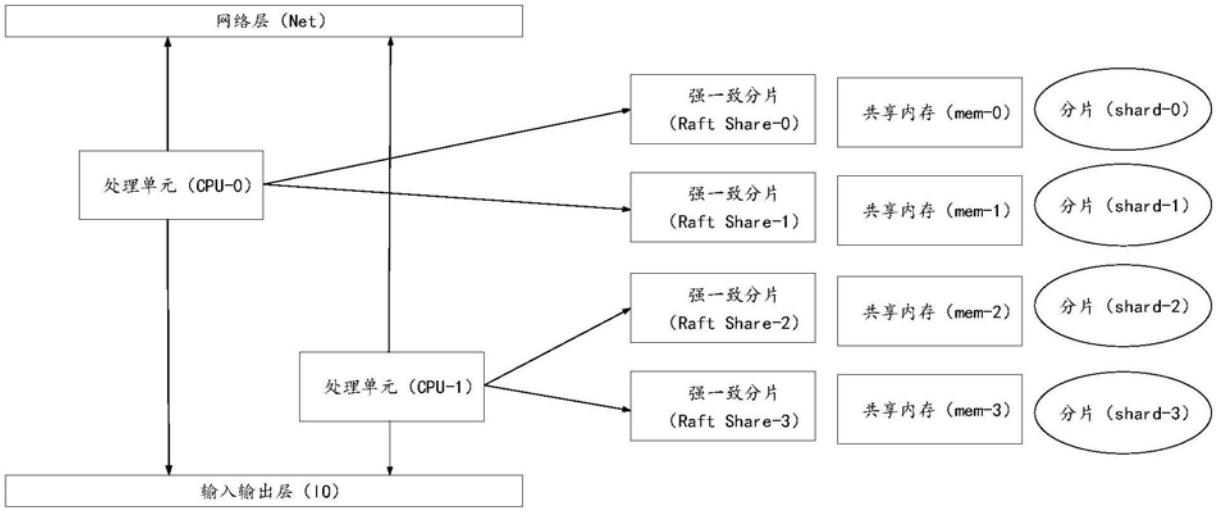

网络层（Net）
处理单元（CPU-0）
处理单元（CPU-1）
输入输出层（IO）
强一致分片（Raft Share-0）
强一致分片（Raft Share-1）
强一致分片（Raft Share-2）
强一致分片（Raft Share-3）
共享内存（mem-0）
共享内存（mem-1）
共享内存（mem-2）
共享内存（mem-3）
分片（shard-0）
分片（shard-1）
分片（shard-2）
分片（shard-3）

图3

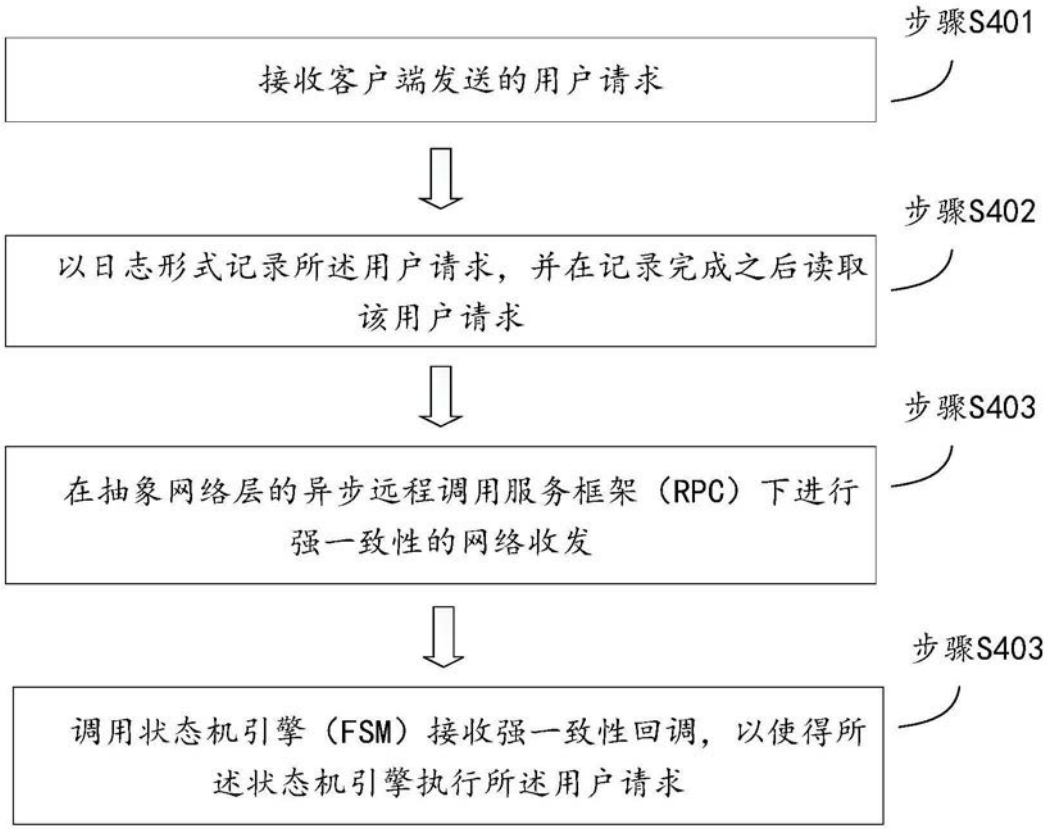

步骤S401
接收客户端发送的用户请求
步骤S402
以日志形式记录所述用户请求，并在记录完成之后读取该用户请求
步骤S403
在抽象网络层的异步远程调用服务框架（RPC）下进行强一致性的网络收发
步骤S403
调用状态机引擎（FSM）接收强一致性回调，以使得所述状态机引擎执行所述用户请求

图4

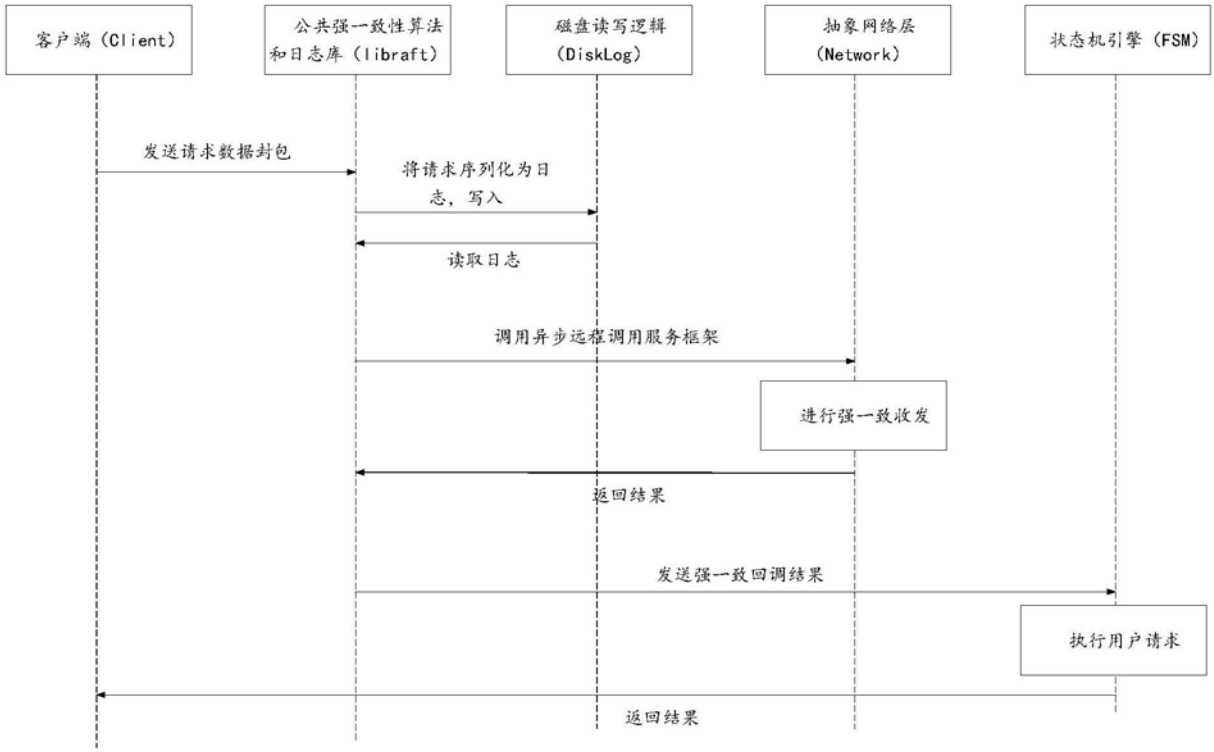

客户端（Client）
公共强一致性算法和日志库（libraft）
磁盘读写逻辑（DiskLog）
抽象网络层（Network）
状态机引擎（FSM）
发送请求数据封包
将请求序列化为日志，写入
读取日志
调用异步远程调用服务框架
进行强一致收发
返回结果
发送强一致回调结果
执行用户请求
返回结果

图5

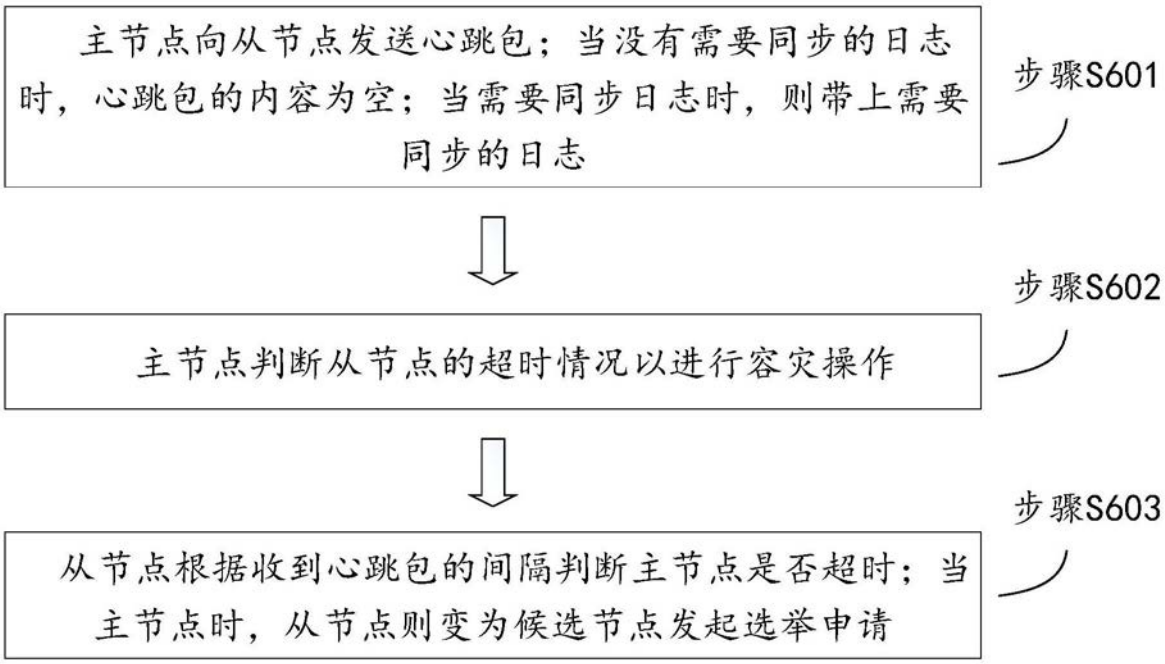

主节点向从节点发送心跳包；当没有需要同步的日志时，心跳包的内容为空；当需要同步日志时，则带上需要同步的日志
步骤S601
主节点判断从节点的超时情况以进行容灾操作
步骤S602
从节点根据收到心跳包的间隔判断主节点是否超时；当主节点时，从节点则变为候选节点发起选举申请
步骤S603

图6

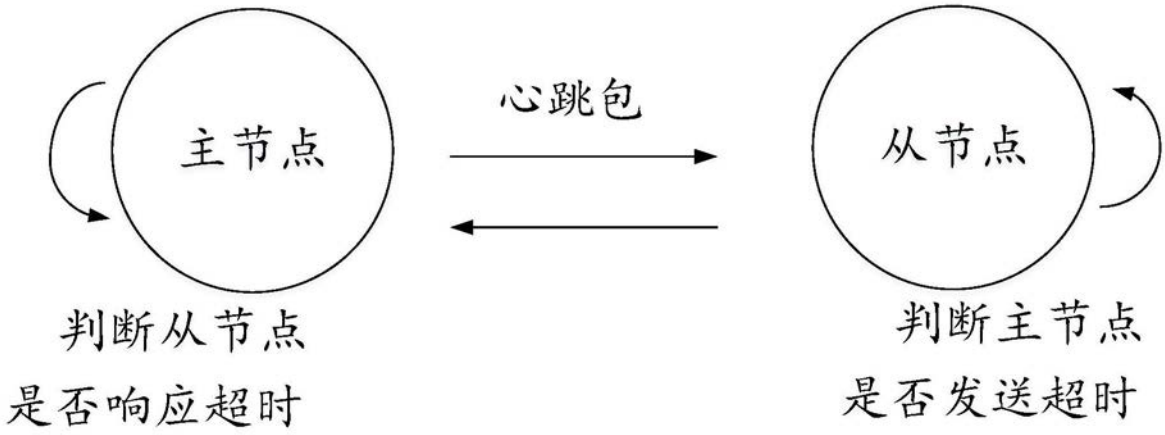

主节点
心跳包
从节点
判断从节点
是否响应超时
判断主节点
是否发送超时

图7

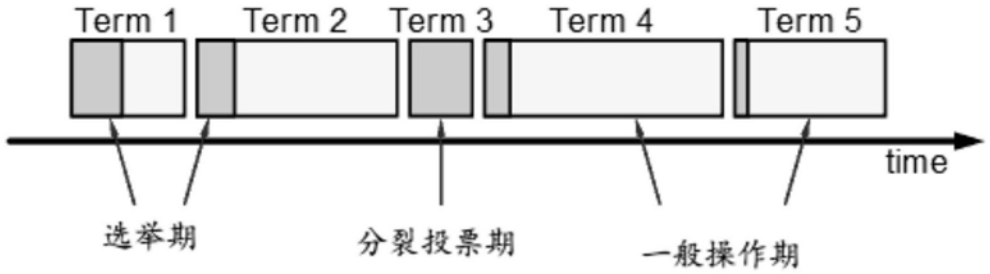

Term 1
Term 2
Term 3
Term 4
Term 5
time
选举期
分裂投票期
一般操作期

图8

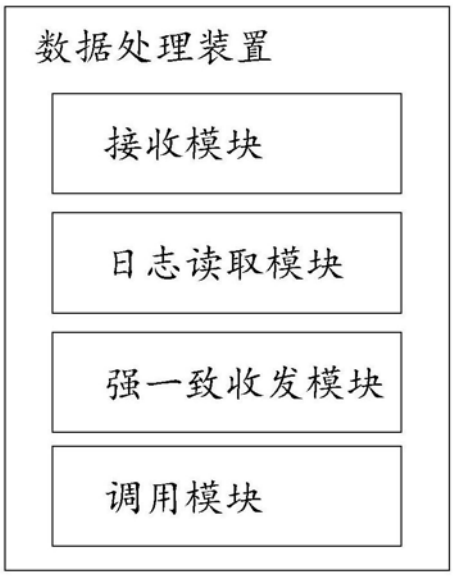

数据处理装置
接收模块
日志读取模块
强一致收发模块
调用模块

图9

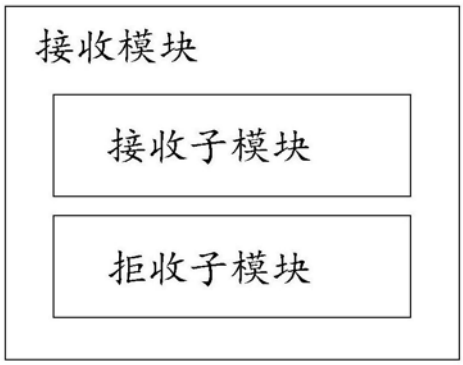


图10

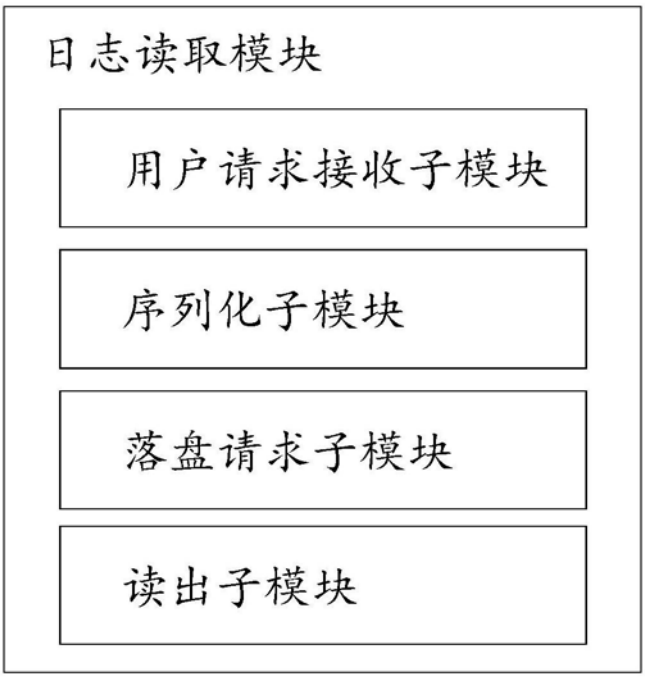


图11


强一致收发模块
主从同步子模块
落盘通知子模块

图12

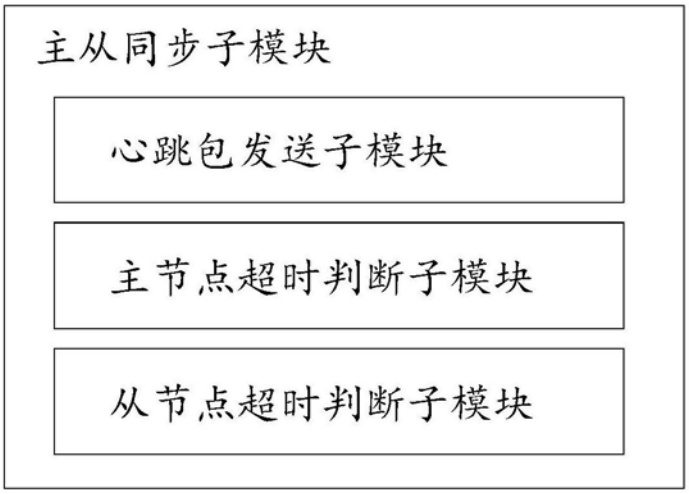

主从同步子模块
心跳包发送子模块
主节点超时判断子模块
从节点超时判断子模块

图13


调用模块
调用子模块
结果返回模块

图14

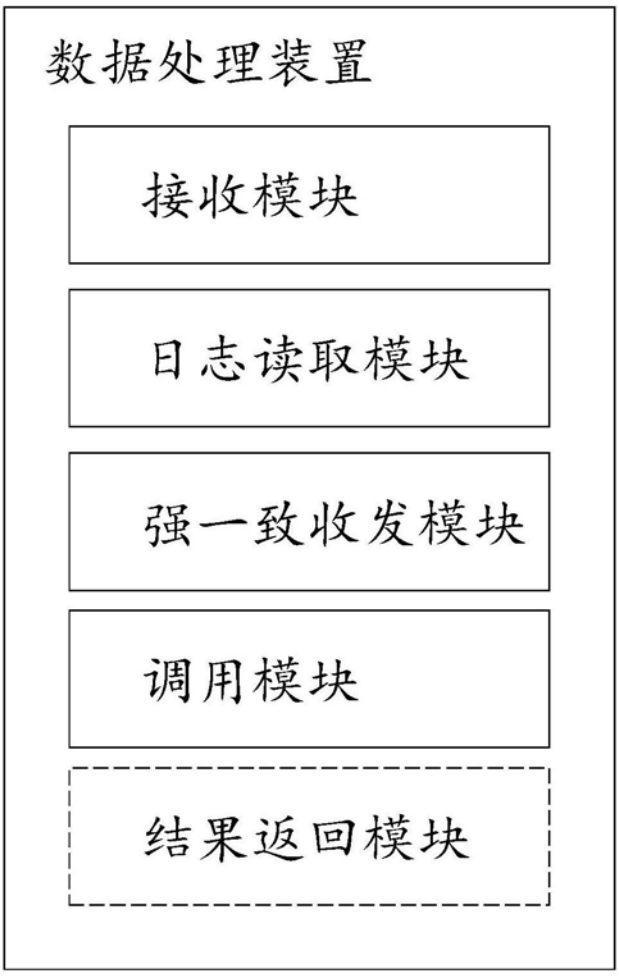

数据处理装置
接收模块
日志读取模块
强一致收发模块
调用模块
结果返回模块

图15

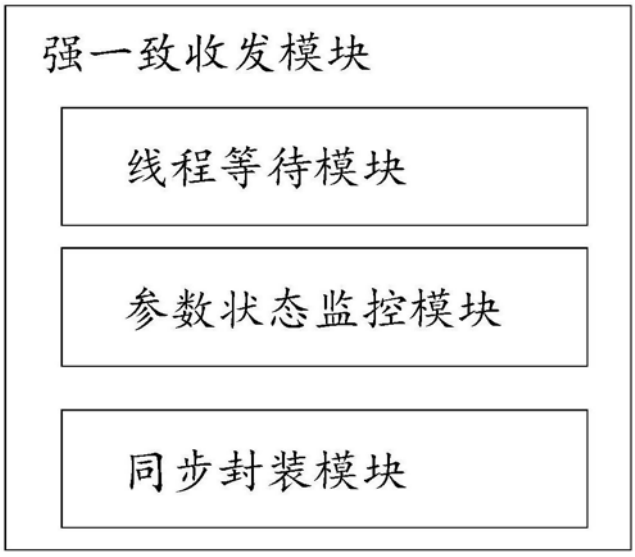

强一致收发模块
线程等待模块
参数状态监控模块
同步封装模块

图16

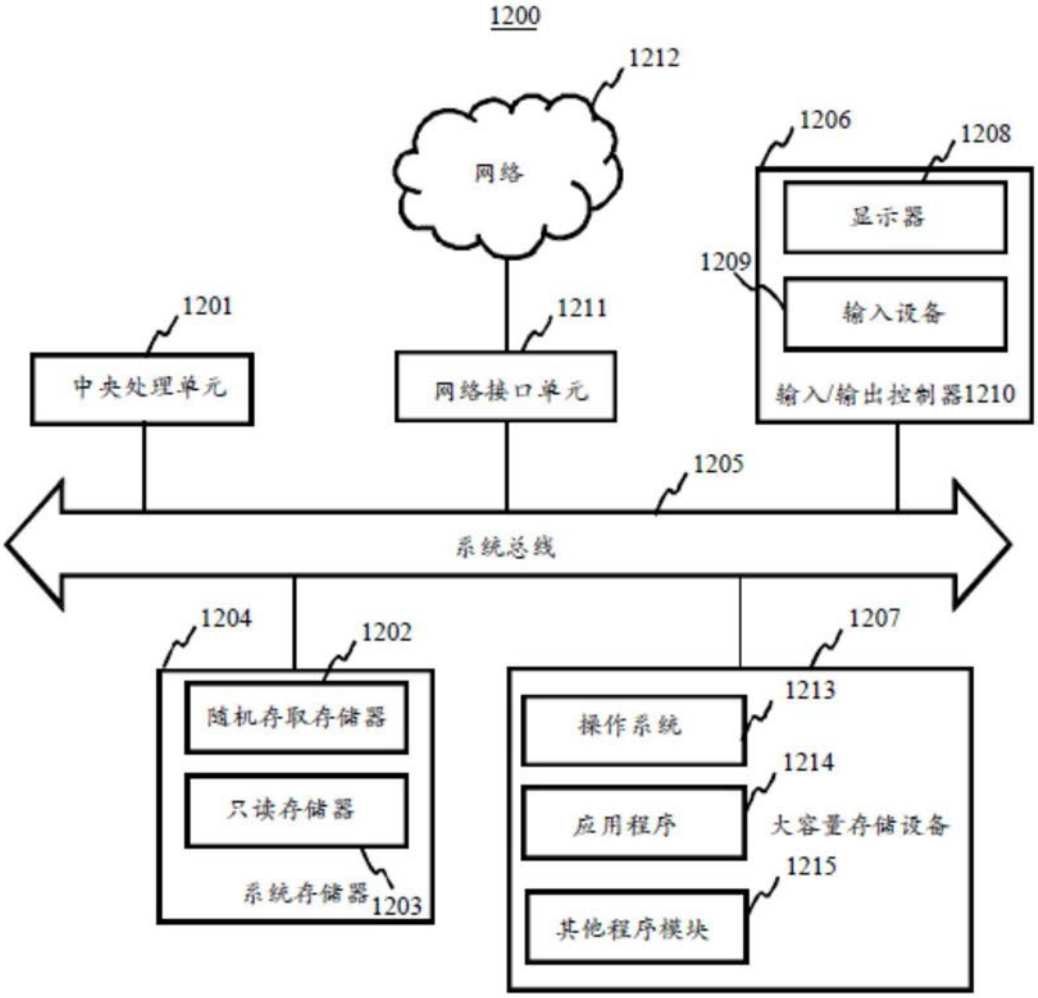

1200
1212
网络
1206
1208
显示器
1209
输入设备
输入/输出控制器1210
1201
中央处理单元
1211
网络接口单元
1205
系统总线
1204
1202
随机存取存储器
只读存储器
系统存储器
1203
1207
1213
操作系统
1214
应用程序
大容量存储设备
1215
其他程序模块

图17