(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 108347455 B
(45)授权公告日 2021.03.26

(21)申请号 201710053030.2
(22)申请日 2017.01.24
(65)同一申请的已公布的文献号
申请公布号 CN 108347455 A
(43)申请公布日 2018.07.31
(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 程霖 朱云锋 付鑫 安凯歌 唐治洋 陶云峰 卢毅军
(74)专利代理机构 上海百一领御专利代理事务所(普通合伙) 31243
代理人 陈贞健
(51)Int.Cl.
H04L 29/08(2006.01)

(56)对比文件
CN 105635278 A,2016.06.01
CN 105187556 A,2015.12.23
CN 101079902 A,2007.11.28
CN 104166600 A,2014.11.26
CN 103167026 A,2013.06.19
US 2015207880 A1,2015.07.23

审查员 郑骏

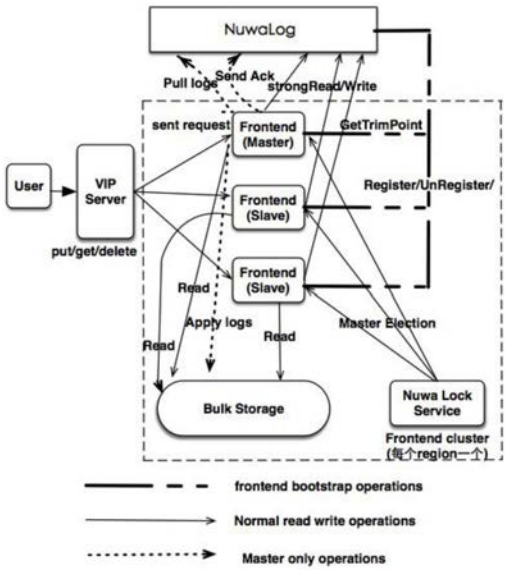
权利要求书5页 说明书23页 附图4页

(54)发明名称

元数据交互方法及系统

(57)摘要

本发明的目的是提供一种元数据交互方法及系统,通过本发明通过将多个后端机组成一个后端一致性总系统,将后端机分成多个分区,各个分区分别响应对应的请求,分担请求压力,避免由同一个后端机响应所有请求,导致压力过大的问题,实现了后端机的水平扩展,另外,通过为不同的用户分配对应的分区,能够给每个用户所写的元数据资源空间做Quota配额,而且用户之间的元数据得到分区与隔离,安全性也得到保证,也便于区分不同的用户进行Quota配额的计费。



CN 108347455 B

1.一种后端机上的元数据交互方法,其中,包括:

从前端机接收用户的更新对应分区的元数据请求,根据所述请求将对应的元数据更新入对应分区;

向所述前端机发送元数据,并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置,根据所述所在的分区中的位置删除所述前端机回复的多个所述位置中已经拉取的元数据的最小ID号的下一个ID号之前所有的元数据。

2.根据权利要求1所述的方法,其中,向所述前端机发送元数据之后,还包括:

从所述前端机获取用户的读取对应分区的元数据请求,根据所述读取元数据请求向所述前端机发送对应的元数据。

3.根据权利要求2所述的方法,其中,所述前端机采用主/备服务器架构。

4.根据权利要求3所述的方法,其中,从前端机接收用户的更新对应分区的元数据请求或从所述前端机获取用户的读取对应分区的元数据请求之前,还包括:

从所述主服务器和备服务器接收注册请求;

根据所述注册请求对所述主服务器和备服务器进行注册后,向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的元数据在所在的分区中的位置。

5.根据权利要求3所述的方法,其中,由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

6.根据权利要求5所述的方法,其中,由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时,还包括:

从下线的主服务器和/或备服务器接收注销请求,根据所述注销请求将下线的主服务器和/或备服务器进行注销后,向所述下线的主服务器和/或备服务器发送注销结果信息;

从新上线的主服务器和/或备服务器接收注册请求,根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后,向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的元数据在所在的分区中的位置。

7.根据权利要求1所述的方法,其中,将对应的元数据更新入对应分区的同时,还包括:

当所述元数据为日志时,根据所述分区中当前存储的元数据,生成对应快照并存储入所述分区;

根据所述所在的分区中位置删除所述分区中已经拉取的元数据的同时还包括:

从所述分区中删除对应的快照。

8.一种前端机上的元数据交互方法,其中,该方法包括:

接收用户的更新对应分区的元数据请求,通过向对应分区发送所述请求,将元数据更新入对应分区中;

从对应分区拉取元数据,并将拉取到的元数据存储至前端机所在地域的本地存储系统,并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置,以供所述分区根据所述所在的分区中的位置删除已经拉取的元数据的最小ID号的下一个ID号之前所有的元数据。

9.根据权利要求8所述的方法,其中,将拉取到的元数据存储至前端机所在地域的本地存储系统之后,还包括:

接收用户的读取对应分区的元数据请求;

根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，

若从所述本地存储系统获取到，则将获取到对应分区的元数据回复至所述用户。

10.根据权利要求9所述的方法，其中，从所述前端机所在地域的本地存储系统获取对应分区的元数据之后，还包括：

若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到对应分区的元数据回复至所述用户。

11.根据权利要求10所述的方法，其中，所述前端机采用主/备服务器架构。

12.根据权利要求11所述的方法，其中，接收用户的更新对应分区的元数据请求或接收用户的读取对应分区的元数据请求之前，包括：

所述主服务器和备服务器向后端一致性总系统发送注册请求；

所述主服务器和备服务器从后端一致性总系统接收注册成功信息后，所述主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

13.根据权利要求11所述的方法，其中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

14.根据权利要求13所述的方法，其中，分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时，还包括：

下线的主服务器和备服务器，向所述分布式锁服务系统发送注销请求，从后端一致性总系统接收注销结果信息；

新上线的主服务器和备服务器向后端一致性总系统发送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

15.根据权利要求11所述的方法，其中，接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中，包括：

所述主服务器或备服务器从一负载均衡服务器获取用户的更新对应分区的元数据请求，所述主服务器或备服务器通过向所述后端机发送所述用户的更新对应分区的元数据请求，将元数据更新入对应分区；

从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述后端机回复当前已经拉取的元数据在所在的分区中的位置，包括：

所述主服务器从所述对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复已经拉取的元数据在所在的分区中的位置。

16.根据权利要求15所述的方法，其中，接收用户的读取对应分区的元数据请求，根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户，包括：

所述主服务器或备服务器从所述负载均衡服务器接收用户的读取对应分区的元数据请求；

所述主服务器或备服务器根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户。

17.根据权利要求16所述的方法，其中，若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户，包括：

若未从所述本地存储系统获取到，所述主服务器或备服务器通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户。

18.一种后端机，其中，包括：

更新装置，用于从前端机接收用户的更新对应分区的元数据请求，根据所述请求将对应的元数据更新入对应分区；

发送装置，向所述前端机发送元数据，并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置，根据所述所在的分区中的位置删除所述前端机回复的多个所述位置中已经拉取的元数据的最小ID号的下一个ID号之前所有的元数据。

19.根据权利要求18所述的后端机，其中，所述发送装置，还用于从所述前端机获取用户的读取对应分区的元数据请求，根据所述读取元数据请求向所述前端机发送对应的元数据。

20.根据权利要求19所述的后端机，其中，所述前端机采用主/备服务器架构。

21.根据权利要求20所述的后端机，其中，还包括注册装置，用于：从所述主服务器和备服务器接收注册请求；根据所述注册请求对所述主服务器和备服务器进行注册后，向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的元数据在所在的分区中的位置。

22.根据权利要求20所述的后端机，其中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

23.根据权利要求22所述的后端机，其中，所述注册装置，还用于从下线的主服务器和/或备服务器接收注销请求，根据所述注销请求将下线的主服务器和/或备服务器进行注销后，向所述下线的主服务器和/或备服务器发送注销结果信息；

所述注册装置，还用于从新上线的主服务器和/或备服务器接收注册请求，根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后，向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的元数据在所在的分区中的位置。

24.根据权利要求18所述的后端机，其中，所述更新装置，还用于当所述元数据为日志时，将对应的元数据更新入对应分区的同时，根据所述分区中当前存储的元数据，生成对应快照并存储入所述分区；

所述发送装置，还用于根据所述所在的分区中位置删除所述分区中已经拉取的元数据的同时，从所述分区中删除对应的快照。

25.一种前端机，其中，该前端机包括：

发起更新装置，用于接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中；

拉取装置，从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除已经拉取的元数据的最小ID号的下一个ID号之

前所有的元数据。

26.根据权利要求25所述的前端机，其中，还包括读取装置，用于接收用户的读取对应分区的元数据请求；根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，

若从所述本地存储系统获取到，则将获取到对应分区的元数据回复至所述用户。

27.根据权利要求26所述的前端机，其中，所述读取装置，还用于若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到对应分区的元数据回复至所述用户。

28.根据权利要求27所述的前端机，其中，所述前端机采用主/备服务器架构。

29.根据权利要求28所述的前端机，其中，还包括发起注册装置，用于供所述主服务器和备服务器向后端一致性总系统发送注册请求；所述主服务器和备服务器从后端一致性总系统接收注册成功信息后，所述主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

30.根据权利要求28所述的前端机，其中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

31.根据权利要求30所述的前端机，其中，还包括发起注销装置，用于下线的主服务器和备服务器，向所述分布式锁服务系统发送注销请求，从后端一致性总系统接收注销结果信息；

所述发起注册装置，还用于新上线的主服务器和备服务器向后端一致性总系统发送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

32.根据权利要求28所述的前端机，其中，所述发起更新装置，用于供所述主服务器或备服务器从一负载均衡服务器获取用户的更新对应分区的元数据请求，所述主服务器或备服务器通过向所述后端机发送所述用户的更新对应分区的元数据请求，将元数据更新入对应分区；

所述拉取装置，用于所述主服务器从所述对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复已经拉取的元数据在所在的分区中的位置。

33.根据权利要求32所述的前端机，其中，所述读取装置用于供所述主服务器或备服务器从所述负载均衡服务器接收用户的读取对应分区的元数据请求；及所述主服务器或备服务器根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户。

34.根据权利要求33所述的前端机，其中，所述读取装置，还用于若未从所述本地存储系统获取到，所述主服务器或备服务器通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户。

35.一种基于计算的设备，包括：

处理器；以及

被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

从前端机接收用户的更新对应分区的元数据请求，根据所述请求将对应的元数据更新入对应分区；

向所述前端机发送元数据，并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置，根据所述所在的分区中的位置删除所述前端机回复的多个所述位置中已经拉取的元数据的最小ID号的下一个ID号之前所有的元数据。

36.一种基于计算的设备，包括：

处理器；以及

被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中；

从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除已经拉取的元数据的最小ID号的下一个ID号之前所有的元数据。

元数据交互方法及系统

技术领域

[0001] 本发明涉及计算机领域,尤其涉及一种元数据交互方法及系统。

背景技术

[0002] 现在越来越多的业务的数据存储需要突破数据中心的边界,实现跨地域服务的能力。基于跨地域的数据异步复制看似能解决这个问题,然而事实上这种异步复制会带来数据难以达到一致性的问题,甚至存在无法处理地域级别的容灾问题。在一些对数据一致性和可靠性要求极高的行业,例如互联网金融业,这种异步复制是完全不能胜任的。

[0003] 因此,在设计跨地域一致性元数据存储系统(Global Meta System)方案时,后端系统会始终离不开使用分布式一致性系统(Quorum)来解决数据的高可用,一致性等。Quorum组图如图2所示,Quorum组中有角色Leader,Follower,Observer。在分布式一致性系统中,把组成一致性系统的几台机器称为一个Quorum,这个Quorum里面的机器中的数据是一样的,其中有几个角色,Leader代表是该Quorum中的领导者,也就是所有请求到Quorum中的事务性请求都必须进过Leader来处理;Follower是接受非事务性请求,如果接收到事务性请求就需要转发给Leader。Leader和Follower之间需要对用户的事务性请求数据进行决策,同不同意这个请求数据。Observer在Quorum中只是个学习者,就从Leader中拉取数据,主要起到备份数据的作用。在此,事务性请求指的是写/更新等可以改变后端一致性系统(NuwaLog)中的log数据的请求;非事务性请求指的是读等不改变后端一致性系统(NuwaLog)log数据的请求。

[0004] 随着Quorum组中的机器增多,一个事务性请求请求需要proposer到Quorum组中N/2+1台机器得到确认(ack),其中,N代表该Quorum中的机器数量,一般在Quorum组中部署奇数个机器的Server,其中,机器之间的通信导致的网络延迟增大,由此可见,Quorum不能任意扩展,Quorum组之间通信的开销会导致事务性请求的性能下降。

[0005] 目前,业界有一些比较好的一致性系统,如:zookeeper,chubby等。后端一致性系统(Quorum)一般由奇数个服务器(server)组成。跨地域一致性元数据系统中的(Frontend)的所有前端事务性请求都需要达到后端中,所有的请求都会与后端建立TCP连接与会话session,这样会导致后端的压力,而理论上Quorum不能任意扩展,因为Quorum组之间通信的开销会导致事务性请求的性能下降。另外,在每一个Quorum组中,用户的数据都往这个组里面写,会经常出现后端压力过大,用户所写的数据会不均匀,有的用户写的数据多,有的用户写的数据少,不能给每个用户所写的资源空间做Quota配额,而且用户之间的数据得不到分区与隔离,安全性也得不到保证的问题。

[0006] 在全球化的场景下,单纯的利用一致性系统存储元数据如log(日志)和snapshot(快照),会造成单机存储性能的下降和存储容量的限制,数据存储站点数量受一致性协议Quorum节点数的限制,而且每个地域(region)读取数据性能低。其中,Log为事务性日志,也就是用户发送的请求数据,在后端一致性系统中统一称为Log;Snapshot为快照,也就是后端一致性系统中某一时刻内存中全量数据的快照。

[0007] 在设计跨地域一致性元数据存储系统架构中,最核心的是需要分布式一致性系统来保证全球元数据的一致性。业界通用的做法是利用zookeeper或者chubby,或者基于paxos实现的一致性系统,如图1所示,它们的通用架构是:

[0008] 每个地域(region)需要一个本地存储系统(storage system),可以是mysql或者分布式数据库等。这种架构的组件比较少,交互协议设计比较简单。分布式一致性系统会每隔一段时间将snapshot打到存储系统中,每个地域的client/user就会从存储系统中获取解析数据。但是,这种架构中,Quorum中的每个机器包括主服务器(Leader)和从服务器(Follower),还是需要存储所有的snapshot,主服务器(Leader)和从服务器(Follower)的内存中还是存储着全量的log数据。

[0009] 在此,可以明显看出上述这种一致性系统存在着明显的可扩展性,单机性能瓶颈,磁盘和内存数据容量受限的问题。而且本地存储系统(storage system)还是得需要从一致性系统中获取数据,这样还得需要Quorum系统中的Leader接入mysql/storage system,主动将snapshot数据发送到mysql/storage system,其性能不佳。因此,这样的架构协议虽然简单,一致性系统不需要任何修改,只要每隔一段时间将snapshot数据推送到mysql/storage system中就可以了,但是,这种架构只是解决了每个地域的用户client/user的读取请求而已,并没有解决全球化跨地域的一致性元数据访问的难点。

[0010] 如上所述,直接使用分布式一致性系统作为跨地域核心后端系统,协议设计简单,系统也不容易出错,但是面临着众多性能和机器存储容量的问题,也不具备可扩展性,这样的分布式系统没法在云计算这么复杂的场景下使用。

发明内容

[0011] 本发明的一个目的是提供一种元数据交互方法及系统,能够解决后端机的问题压力过大,不能给每个用户所写的资源空间做Quota配额,而且用户之间的数据得不到分区与隔离,安全性也得不到保证的问题。

[0012] 本发明提供一种后端机上的元数据交互方法,包括:

[0013] 从前端机接收用户的更新对应分区的元数据请求,根据所述请求将对应的元数据更新入对应分区;

[0014] 向所述前端机发送元数据,并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置,根据所述所在的分区中的位置删除对应分区中所述已经拉取的元数据。

[0015] 进一步的,上述方法中,向所述前端机发送元数据之后,还包括:

[0016] 从所述前端机获取用户的读取对应分区的元数据请求,根据所述读取元数据请求向所述前端机发送对应的元数据。

[0017] 进一步的,上述方法中,所述前端机采用主/备服务器架构。

[0018] 进一步的,上述方法中,从前端机接收用户的更新对应分区的元数据请求或从所述前端机获取用户的读取对应分区的元数据请求之前,还包括:

[0019] 从所述主服务器和备服务器接收注册请求;

[0020] 根据所述注册请求对所述主服务器和备服务器进行注册后,向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的元数据在所在的分区中的位置。

[0021] 进一步的，上述方法中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

[0022] 进一步的，上述方法中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时，还包括：

[0023] 从下线的主服务器和/或备服务器接收注销请求，根据所述注销请求将下线的主服务器和/或备服务器进行注销后，向所述下线的主服务器和/或备服务器发送注销结果信息；

[0024] 从新上线的主服务器和/或备服务器接收注册请求，根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后，向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的元数据在所在的分区中的位置。

[0025] 进一步的，上述方法中，将对应的元数据更新入对应分区的同时，还包括：

[0026] 当所述元数据为日志时，根据所述分区中当前存储的元数据，生成对应快照并存储入所述分区；

[0027] 根据所述所在的分区中位置删除所述分区中已经拉取的元数据的同时还包括：

[0028] 从所述分区中删除对应的快照。

[0029] 根据本发明的另一面，还提供一种前端机上的元数据交互方法，该方法包括：

[0030] 接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中；

[0031] 从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除其内部存储的所述已经拉取的元数据。

[0032] 进一步的，上述方法中，将拉取到的元数据存储至前端机所在地域的本地存储系统之后，还包括：

[0033] 接收用户的读取对应分区的元数据请求；

[0034] 根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，

[0035] 若从所述本地存储系统获取到，则将获取到对应分区的元数据回复至所述用户。

[0036] 进一步的，上述方法中，从所述前端机所在地域的本地存储系统获取对应分区的元数据之后，还包括：

[0037] 若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到对应分区的元数据回复至所述用户。

[0038] 进一步的，上述方法中，所述前端机采用主/备服务器架构。

[0039] 进一步的，上述方法中，接收用户的更新对应分区的元数据请求或接收用户的读取对应分区的元数据请求之前，包括：

[0040] 所述主服务器和备服务器向后端一致性总系统发送注册请求；

[0041] 所述主服务器和备服务器从后端一致性总系统接收注册成功信息后，所述主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

[0042] 进一步的，上述方法中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

[0043] 进一步的，上述方法中，分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时，还包括：

[0044] 下线的主服务器和备服务器，向所述分布式锁服务系统发送注销请求，从后端一致性总系统接收注销结果信息；

[0045] 新上线的主服务器和备服务器向后端一致性总系统发送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

[0046] 进一步的，上述方法中，接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中，包括：

[0047] 所述主服务器或备服务器从一负载均衡服务器获取用户的更新对应分区的元数据请求，所述主服务器或备服务器通过向所述后端机发送所述用户的更新对应分区的元数据请求，将元数据更新入对应分区；

[0048] 从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述后端机回复当前已经拉取的元数据在所在的分区中的位置，包括：

[0049] 所述主服务器从所述对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复已经拉取的元数据在所在的分区中的位置。

[0050] 进一步的，上述方法中，接收用户的读取对应分区的元数据请求，根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户，包括：

[0051] 所述主服务器或备服务器从所述负载均衡服务器接收用户的读取对应分区的元数据请求；

[0052] 所述主服务器或备服务器根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户。

[0053] 进一步的，上述方法中，若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户，包括：

[0054] 若未从所述本地存储系统获取到，所述主服务器或备服务器通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户。

[0055] 根据本发明的另一面，还提供一种后端机，包括：

[0056] 更新装置，用于从前端机接收用户的更新对应分区的元数据请求，根据所述请求将对应的元数据更新入对应分区；

[0057] 发送装置，向所述前端机发送元数据，并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置，根据所述所在的分区中的位置删除对应分区中所述已经拉取的元数据。

[0058] 进一步的，上述后端机中，所述发送装置，还用于从所述前端机获取用户的读取对应分区的元数据请求，根据所述读取元数据请求向所述前端机发送对应的元数据。

[0059] 进一步的，上述后端机中，所述前端机采用主/备服务器架构。

[0060] 进一步的，上述后端机中，还包括注册装置，用于：从所述主服务器和备服务器接收注册请求；根据所述注册请求对所述主服务器和备服务器进行注册后，向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的元数据在所在的分区中的位置。

[0061] 进一步的，上述后端机中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

[0062] 进一步的，上述后端机中，所述注册装置，还用于从下线的主服务器和/或备服务器接收注销请求，根据所述注销请求将下线的主服务器和/或备服务器进行注销后，向所述下线的主服务器和/或备服务器发送注销结果信息；

[0063] 所述注册装置，还用于从新上线的主服务器和/或备服务器接收注册请求，根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后，向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的元数据在所在的分区中的位置。

[0064] 进一步的，上述后端机中，所述更新装置，还用于当所述元数据为日志时，将对应的元数据更新入对应分区的同时，根据所述分区中当前存储的元数据，生成对应快照并存储入所述分区；

[0065] 所述发送装置，还用于根据所述所在的分区中位置删除所述分区中已经拉取的元数据的同时，从所述分区中删除对应的快照。

[0066] 根据本发明的另一面，还提供一种前端机，该前端机包括：

[0067] 发起更新装置，用于接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中；

[0068] 拉取装置，从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除其内部存储的所述已经拉取的元数据。

[0069] 进一步的，上述前端机中，还包括读取装置，用于接收用户的读取对应分区的元数据请求；根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，

[0070] 若从所述本地存储系统获取到，则将获取到对应分区的元数据回复至所述用户。

[0071] 进一步的，上述前端机中，所述读取装置，还用于若未从所述本地存储系统获取到，通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到对应分区的元数据回复至所述用户。

[0072] 进一步的，上述前端机中，所述前端机采用主/备服务器架构。

[0073] 进一步的，上述前端机中，还包括发起注册装置，用于供所述主服务器和备服务器向后端一致性总系统发送注册请求；所述主服务器和备服务器从后端一致性总系统接收注册成功信息后，所述主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

[0074] 进一步的，上述前端机中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。

[0075] 进一步的，上述前端机中，还包括发起注销装置，用于下线的主服务器和备服务

器，向所述分布式锁服务系统发送注销请求，从后端一致性总系统接收注销结果信息；

[0076] 所述发起注册装置，还用于新上线的主服务器和备服务器向后端一致性总系统发送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从对应分区获取当前已经拉取的元数据在所在的分区中的位置。

[0077] 进一步的，上述前端机中，所述发起更新装置，用于供所述主服务器或备服务器从一负载均衡服务器获取用户的更新对应分区的元数据请求，所述主服务器或备服务器通过向所述后端机发送所述用户的更新对应分区的元数据请求，将元数据更新入对应分区；

[0078] 所述拉取装置，用于所述主服务器从所述对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复已经拉取的元数据在所在的分区中的位置。

[0079] 进一步的，上述前端机中，所述读取装置用于供所述主服务器或备服务器从所述负载均衡服务器接收用户的读取对应分区的元数据请求；及所述主服务器或备服务器根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从所述本地存储系统获取到，则将获取到的元数据回复至所述用户。

[0080] 进一步的，上述前端机中，所述读取装置，还用于若未从所述本地存储系统获取到，所述主服务器或备服务器通过向对应分区发送所述读取元数据请求，从对应分区中获取元数据后，将获取到的元数据回复至所述用户。

[0081] 根据本发明的另一面，还提供一种基于计算的设备，包括：

[0082] 处理器；以及

[0083] 被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

[0084] 从前端机接收用户的更新对应分区的元数据请求，根据所述请求将对应的元数据更新入对应分区；

[0085] 向所述前端机发送元数据，并从所述前端机接收当前已经拉取的元数据在所在的分区中的位置，根据所述所在的分区中的位置删除对应分区中所述已经拉取的元数据

[0086] 根据本发明的另一面，还提供一种基于计算的设备，包括：

[0087] 处理器；以及

[0088] 被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

[0089] 接收用户的更新对应分区的元数据请求，通过向对应分区发送所述请求，将元数据更新入对应分区中，；

[0090] 从对应分区拉取元数据，并将拉取到的元数据存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的元数据所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除其内部存储的所述已经拉取的元数据。

[0091] 与现有技术相比，本发明通过将多个后端机组成一个后端一致性总系统，将后端机分成多个分区，各个分区分别响应对应的请求，分担请求压力，避免由同一个后端机响应所有请求，导致压力过大的问题，实现了后端机的水平扩展，另外，通过为不同的用户分配对应的分区，能够给每个用户所写的元数据资源空间做Quota配额，而且用户之间的元数据得到分区与隔离，安全性也得到保证，也便于区分不同的用户进行Quota配额的计费，此外，

通过前端机(Frontend)不断从某个后端机(NuwaLog)的对应分区上拉取日志,并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS),向对应分区回复当前已经拉取的日志的位置,所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志,这样对应分区不需要存储所有的日志,只需要存储还未拉取到本地存储系统(OTS),可以避免后端机(NuwaLog)的单机存储性能的下降和存储容量的限制,数据存储站点数量受一致性协议Quorum节点数的限制的问题,达到整个系统的数据一致性、正确性与健壮性。

附图说明

[0092] 通过阅读参照以下附图所作的对非限制性实施例所作的详细描述,本发明的其它特征、目的和优点将会变得更明显:

[0093] 图1示出现有的跨地域强一致性元数据存储系统的架构图;

[0094] 图2示出现有的Quorum组图;

[0095] 图3示出本发明一实施例的跨地域强一致性系统的简易组件图;

[0096] 图4示出本发明一实施例的跨地域强一致性系统协议交互通信图;

[0097] 图5示出本发明一实施例的简易组件图;

[0098] 图6示出本发明一实施例的跨地域强一致性系统的原理图。

具体实施方式

[0099] 下面结合附图对本发明作进一步详细描述。

[0100] 在本申请一个典型的配置中,终端、服务网络的设备和可信方均包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

[0101] 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。

[0102] 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非暂存电脑可读媒体(transitory media),如调制的数据信号和载波。

[0103] 下面以元数据是日志为例,对本发明进行详细介绍,本领域技术人员能够理解,所述元数据包括但不限于日志等数据,元数据可以是各种描述数据属性(property)的信息,用来支持如指示存储位置、历史数据、资源查找、文件记录等功能。

[0104] 首先,如图3所示,将多个后端机(Nuwalog)组成一个后端一致性总系统(Global Meta System),将后端机分成多个分区(Domain),即后端机可以设计成多Quorum形式,每个Quorum可以设计成多Domain的形式,各个分区分别响应对应的请求,分担请求压力,减轻前端系统Frontend对后端机Quorum的压力,避免由同一个后端机响应所有请求,导致压力过

大的问题,用户访问需要指定Global Meta System,quorum与domain信息去一层层访问需要的资源位置,采用这种NeuronId(后端一致性总系统编号)->QuorumId(后端机编号)->DomainId(分区编号)的多层设计结构,实现了后端机(Nuwalog)的水平扩展,另外,为不同的用户分配对应的分区,以便给每个用户所写的元数据资源空间做Quota配额,而且用户之间的元数据得到分区与隔离,安全性也得到保证,也便于区分不同的用户进行Quota配额的计费。在此,可在全球范围内部署多个后端一致性总系统(Global Meta System),图3中示出本发明以只在全球部署一个后端一致性总系统(Global Meta System)的实施例。

[0105] 如图3所示,用户(client)这边发送的put/get/delete请求,也就是写读删操作(requests)通过指定资源访问位置,在本发明的多层设计中,需要指定NeuronId,QuorumId,DomainId来发送请求如restful请求到Frontend端,Frontend端通过存储的映射关系去找到用户指定资源的位置的机器以及资源路径PATH,用户的资源存储在后端的NeuronId->QuorumId->DomainId中,多层次的组合能够准确定位用户访问资源的位置。

[0106] 根据图3的系统架构图可以看出,需要三层访问机制才可以到达后端NuwaLog系统。当然,全球目前配置一个NuwaLog作为全球后端机,有需求的话可以配置多套,每个NuwaLog有一个NeuronId作为标识,这里就直接标识成NeuronId=1。

[0107] 在每个NuwaLog中部署多套Quorum系统,减轻前端机对后端的压力。每套Quorum系统都以Quorum1,Quorum2,Quorum3....来标识。

[0108] 每个Quorum里面可以建立多个Domain作为用户自己的域,防止用户间的干扰,起到资源隔离的作用,每个Domain都有个标识,Domain1,Domain2,....等。DomainId在NuwaLog后端可以只是一个path(log路径的前缀),作为资源隔离访问。

[0109] 在此,client端需要传入NeuronId,QuorumId,DomainId,这三个元素共同来组成restful协议中的PATH字段,通过标准的restful协议发送到frontend端。关于NeuronId->QuorumId->DomainId与实际的服务器地址的映射关系可保存于一配置文件中。

[0110] 一种方案是可以将所述配置文件存储在frontend本地。frontend端需要根据用户请求中的NeuronId,QuorumId,DomainId来从本地的frontend配置文件中寻找对应的服务器(server)地址,也就是NuwaLog后端的服务器ip:port,所以在本地需要配置文件来制定NeuronId,QuorumId里面的机器ip:port地址,配置文件以Json格式来制定,其中DomainId表示用户的唯一ID号,写到后端NuwaLog上的path会带上这个DomainId。例如,如图4所示,Neuron1代表一个Neuron的ID号为1,其中Quorum1的三个server的ip地址以及端口号,Quorum2的三个server的ip地址以及端口号。用户传来的信息是Neuron1,Quorum1这样的ID号,那么数据就会传输到10.101.10.10,10.101.10.11,10.101.10.12上面,然后根据用户传来的DomainID号,来作为用户写到后端server上的path前缀,这里写到后端的路径前缀是/DomainID/。

[0111] 另一种方案是可以将所述配置文件存储在后端的NuwaLog中。利用后端NuwaLog的通知机制,QuorumId与DomainId的映射关系可以在后端的NuwaLog中存储,frontend可以去订阅该信息。QuorumId与DomainId的映射关系可以通过事先输入到后端NuwaLog系统中存储,frontend就会获取到订阅的信息,进而存储在内存中。这样frontend可以根据用户请求中的path去解析,找到需要访问的机器IP与资源位置。

[0112] 系统前端frontend与后端NuwaLog系统之间采用标准的Restful协议进行通信,之

所以采用标准的Restful协议更能结合资源定位，用户数据存储的位置都当作资源单位。当client发送来的请求是一个标准的restful协议，frontend这边根据传来的NeuronId，QuorumId，DomainId的信息，获取到对应的NuwaLog后端的服务的ip地址和端口号。这种设计可以定位数据归属地址，实现成功数据访问，其中，NeuronId作为首地址，起到资源准确定位到全球哪一个地域。

[0113] 其次，给出本发明设计的log和snapshot分离的跨地域强一致性系统(Global Meta System)的简易组件图，如图5所示，从图5的简易组件图可以看出，有三个server(服务)组件，分别为：

[0114] 分布式一致性系统(NuwaLog)，前端机(Frontend)和本地存储系统(bulk local storage)，其中，

[0115] 分布式一致性系统(NuwaLog)是后端Log一致性存储系统，其是一个Quorum，是一个事务性日志提交系统，前端机(Frontend)接收到的用户(client/user)的内容会组装成log写到后端NuwaLog，其能够针对用户的数据做到一致性并高可靠，具有failover(故障切换)情况下不丢数据的功能，所有的请求在NuwaLog里面都只是一条Log而已，不需要理解该Log的任何含义，其中，Log为事务性日志，也就是用户发送的请求数据，在NuwaLog后端系统中统一称为Log；

[0116] 前端机(Frontend)分别在每个地域部署，起前端中转用户请求的作用，以及与NuwaLog推拉数据的功能；

[0117] 本地存储系统(Bulk Storage System/Local Storage)为一云开放存储系统，分别在前端机所在的每个地域部署，作为各地域(region)内的大规模本地存储。在此bulk local storage可是本地的大规模表格存储系统，例如可以是OTS或HDFS系统，能够持久化存储用户的大量数据。

[0118] 当用户(client/user/user)向前端机(Frontend)发送向请求，整个跨地域强一致性系统(Global Meta System)的运行需要各个组件间交互协议的设计合理才能达到系统的正确性与健壮性。

[0119] 本发明在构建跨地域强一致性元数据系统(Global Meta System)时，需要对于后端分布式一致性系统(NuwaLog)进行log和snapshot的分离，每个地域的本地存储系统存储snapshot全量数据，后端机(NuwaLog)只存储log，而且log会随着前端机(Frontend)的读取不断删除(trim)掉。所以这里面临着通常业界分布式一致性系统的拆分。分布式一致性系统拆分后的服务组件之多，系统之间组件间通信协议比较复杂，而且需要保证系统数据的正确性，一致性，可靠性等要素。在此，所述后端机可以为一后端一致性系统。

[0120] 如图6所示，本发明一实施例中Frontend与Bulk Storage System以及NuwaLog交互协议API有：

[0121] (1)Register/UnRegister:frontend向NuwaLog注册或者注销frontendID号，如果有多个NuwaLog组成分布式一致性总系统，每个NuwaLog又分成多个分区时，frontend可以向分布式一致性总系统进行注册，以便后续与对应的分区交互；

[0122] (2)GetTrimPoint:frontend去NuwaLog获取要pull数据的Log NXID号，如果有多个NuwaLog组成分布式一致性总系统，每个NuwaLog又分成多个分区时，frontend去对应分区获取要pull数据的Log NXID号；

[0123] (3)Submit(也称为Write):frontend提交写请求事务性log日志到NuwaLog系统，如果有多个NuwaLog组成分布式一致性总系统，每个NuwaLog又分成多个分区时，frontend提交写请求事务性log日志到对应分区；

[0124] (4)Pull:frontend去NuwaLog上拉取数据，如果有多个NuwaLog组成分布式一致性总系统，每个NuwaLog又分成多个域时，frontend去对应分区上拉取数据；

[0125] (5)Ack:frontend向Bulk Storage写入pull来的数据，并向NuwaLog确认数据写入的最后一个Log的NXID号，如果有多个NuwaLog组成分布式一致性总系统，每个NuwaLog又分成多个分区时，frontend向对应分区确认数据写入的最后一个Log的NXID号。

[0126] 如图3和6所示，本发明提供一种后端机上的元数据交互方法，其中，包括：

[0127] 将多个后端机组成一个后端一致性总系统(Global Meta System)，将后端机分成多个分区(domain)，也就是说一个后端一致性总系统可由多个后端机(Quorum)组成，而每个后端机又由多个域组成；在此，这里的分区是用户之间的资源隔离域；

[0128] 为不同的用户分配对应的分区；

[0129] 如图6所示，从前端机(Frontend)接收用户的更新对应分区的日志请求，根据所述用户的更新日志请求将对应的日志更新入对应的后端一致性总系统中的某个后端机NuwaLog中的所述用户的分区；在此，所述更新包括写入和/或写入后的修改；

[0130] 所述分区不断向所述前端机(Frontend)发送日志，并从所述前端机接收当前已经拉取的日志在所在的分区中的位置，根据所述日志在所在的分区中的位置删除本分区中所述已经拉取的日志。

[0131] 在此，对应分区会接收所有前端机(Frontend)发出的请求，会存储Log。前端机(Frontend)可以启动两个线程，一个线程不断从某个NuwaLog中的对应分区上拉Log数据，另一个线程不断往OTS(Bulk Storage)里面写数据，并回复对应分区已经pull下来的位置，对应分区这边会根据接收到的位置删除掉(trim)所有前端机(Frontend)回复的多个位置中已经pull下来的log的最小ID号的下一个ID号即NXID号之前所有的log，例如目前有三台前端机A、B和C，另后端对应分区中原来有10条日志，各条日志对应为1～10的ID号，各前端机拉取日志均从最小的ID号1开始拉取，若当前端机A拉取到了ID号为3的日志后，向对应分区回复NXID号位置为4，此时前端机B拉取到了ID号为4的日志后，向对应分区回复NXID号位置为5，此时前端机C拉取到了ID号为5的日志后，向对应分区回复NXID号位置为6，那么后续对应分区仅会将ID号为4之前的ID号为1～3的已经拉取的日志从其存储空间内删除，因为ID号为4～5的日志有部分前端机没有拉取完，需要待所有前端机拉取完后才能删除。

[0132] 上述实施例通过将多个后端机组成一个后端一致性总系统，将后端机分成多个分区，各个分区分别响应对应的请求，分担请求压力，避免由同一个后端机响应所有请求，导致压力过大的问题，实现了后端机的水平扩展；另外，通过为不同的用户分配对应的分区，能够给每个用户所写的元数据资源空间做Quota配额，而且用户之间的元数据得到分区与隔离，安全性也得到保证，也便于区分不同的用户进行Quota配额的计费；此外，通过前端机(Frontend)不断从某个后端机(NuwaLog)的对应分区上拉取日志，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS)，向对应分区回复当前已经拉取的日志的位置，所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志，这样对应分区不需要存储所有的日志，只需要存储还未拉取到本地存储系统(OTS)，可以避免后端机

(NuwaLog)的单机存储性能的下降和存储容量的限制,数据存储站点数量受一致性协议Quorum节点数的限制的问题,达到整个系统的数据一致性、正确性与健壮性。

[0133] 本发明一实施例中,如图6所示,对应分区不断向所述前端机发送日志之后,还包括:

[0134] 对应分区从所述前端机(Frontend)获取用户的读取日志请求,根据所述读取日志请求向所述前端机发送对应的日志。本实施例是对所述前端机从其所在地域的本地存储系统获取对应的日志的方案的优化补充,避免当用户需要读取的日志尚未拉取到对应地域的本地存储系统中的情况发生时,也能及时响应用户的日志读取请求,直接从对应分区中读取即可。

[0135] 本发明一实施例中,所述前端机(Frontend)采用主/备服务器架构。在此,采用master/slave架构,可避免Frontend单点故障,导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新。

[0136] 本发明一实施例中,对应分区从前端机接收用户的更新日志请求或从所述前端机获取用户的读取日志请求之前,还包括:

[0137] 后端一致性总系统从所述主服务器和备服务器接收注册请求;

[0138] 后端一致性总系统根据所述注册请求对所述主服务器和备服务器进行注册后,向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的日志在所在的分区中的位置。在此,每一个地域的Frontend包括所述主服务器和备服务器都需要注册到后端一致性总系统上,这样后端一致性总系统中的分区才能够找到对应的Frontend发送请求结果,具体来说,如图6所示,每个Frontend启动的时候,可先调用register注册到NuwaLog,由于主服务器负责事务性操作,注册成功之后主服务器可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号,例如当前删除到ID号为3号的日志,则获取到当前已经拉取的日志的位置为4,即主服务器下一次从位置4开始拉取日志。

[0139] 本发明一实施例中,由一分布式锁服务系统(Nuwa Lock Service)对所述前端机中的主服务器和备服务器进行切换。在此,为了避免Frontend单点故障,导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新了,这里采取master/slave架构保证每次只有主服务器(master)进行事务性操作,当主服务器(master)故障之后能够通过Nuwa Lock Service重新进行选主,也就是选择master,Nuwa Lock Service可以提供分布式锁协议,能够适用于frontend进行master选主,所述分布式锁服务系统(Nuwa Lock Service)可以是一个全球化的分布式锁服务系统(Nuwa Global Lock Service),可以对全球master/slave架构进行管理。

[0140] 本发明一实施例中,由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时,还包括:

[0141] 如图6所示,后端一致性总系统从下线的主服务器和/或备服务器接收注销请求,根据所述注销请求将下线的主服务器和/或备服务器进行注销后,向所述下线的主服务器和/或备服务器发送注销结果信息;

[0142] 后端一致性总系统从新上线的主服务器和/或备服务器接收注册请求,根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后,向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的日志在所在

的分区中的位置。在此，每一个地域的Frontend都需要注册到NuwaLog上，包括新上线的主服务器和备服务器，这样后端一致性总系统才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个新上线的主服务器和备服务器启动的时候，可先调用register注册到后端一致性总系统，由于只有主服务器(master)进行事务性操作，所以注册成功之后主服务器(master)可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器(master)下一次从位置4开始pull日志。另外，当下线掉Frontend包括主服务器和备服务器的时候，需要调用unregister来从后端一致性总系统中下线掉Frontend。

[0143] 本发明一实施例中，如图6所示，将对应的日志更新入对应分区的同时，还包括：

[0144] 根据对应分区当前存储的日志，生成对应快照并存储入同一分区；

[0145] 对应分区根据所述已经拉取的日志在所在的分区中的位置删除本分区所述已经拉取的日志的同时还包括：

[0146] 从所述分区中删除对应的快照。在此，Snapshot为快照，也就是对应分区中某一时刻内存中全量数据的快照。对应分区会接收frontend的请求，会存储log和snapshot，这里存储snapshot是为了解决该分区发生故障切换(failover)的情况，而且log数据还没有传到本地的OTS中，该分区重新启动能够从snapshot中恢复还没有传输到OTS中的数据，该分区根据所述位置删除所述已经拉取的日志的同时，每隔一段时间删除对应磁盘上的snapshot文件，避免存储过期无用的snapshot。

[0147] 本发明一实施例中，前端(frontend)与后端(NuwaLog)之间可以采用avro序列化方式，利用Netty作为RPC通信。这里选择使用avro协议的原因是该序列化协议比较简单并高效。Netty作为业界通用的RPC协议，在JAVA领域是一个成熟度并高效的远程网络通信协议。

[0148] 如图6所示，本发明还提供一种前端机上的元数据交互方法，该方法包括：

[0149] 前端机(Frontend)接收用户的更新对应分区的日志请求，通过向对应分区发送所述用户的更新对应分区的日志请求，将对应的日志(Log)更新入对应分区中其中，不同的用户分配有对应的分区，后端机分成多个分区，多个后端机组成后端一致性总系统；在此，对应分区的会接收所有frontend的请求，会存储log；

[0150] 前端机(Frontend)不断从所述后端机(NuwaLog)上拉取(pull)日志，并不断将拉取到的日志存储(Apply)至前端机所在地域的本地存储系统(OTS)，并向所述对应分区回复(Send Ack)当前已经拉取的日志在所在的分区中的位置，以供所述对应分区根据所述位置删除其内部存储的所述已经拉取(pull)的日志。

[0151] 在此，对应分区会接收所有前端机(Frontend)发出的请求，会存储Log。前端机(Frontend)可以启动两个线程，一个线程不断从对应分区上拉Log数据，另一个线程不断往OTS里面写数据，并回复对应分区已经pull下来的位置，对应分区那边会根据接收到的位置删除掉(trim)所有前端机(Frontend)回复的多个位置中已经pull下来的log的最小ID号的下一个ID号即NXID号之前所有的log，例如目前有三台前端机A、B和C，另后端NuwaLog中原来有10条日志，各条日志对应为1～10的ID号，各前端机拉取日志均从最小的ID号1开始拉取，若当前端机A拉取到了ID号为3的日志后，向NuwaLog回复NXID号位置为4，此时前端机B拉取到了ID号为4的日志后，向NuwaLog回复NXID号位置为5，此时前端机C拉取到了ID号为5

的日志后，向NuwaLog回复NXID号位置为6，那么后续NuwaLog仅会将ID号为4之前的ID号为1～3的已经拉取的日志从其存储空间内删除，因为ID号为4～5的日志有部分前端机没有拉取完，需要待所有前端机拉取完后才能删除。

[0152] 本实施例通过将多个后端机组成一个后端一致性总系统，将后端机分成多个分区，各个分区分别响应对应的请求，分担请求压力，避免由同一个后端机响应所有请求，导致压力过大的问题，实现了后端机的水平扩展；另外，通过为不同的用户分配对应的分区，能够给每个用户所写的元数据资源空间做Quota配额，而且用户之间的元数据得到分区与隔离，安全性也得到保证，也便于区分不同的用户进行Quota配额的计费；此外，通过前端机(Frontend)不断从所述对应分区上拉取日志，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS)，向所述对应分区回复当前已经拉取的日志的位置，所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志，这样对应分区不需要存储所有的日志，只需要存储还未拉取到本地存储系统(OTS)，可以避免对应分区的单机存储性能的下降和存储容量的限制，数据存储站点数量受一致性协议Quorum节点数的限制的问题，达到整个系统的数据一致性、正确性与健壮性，另外，前端机(Frontend)和本地存储系统(OTS)可以实现按需要进行扩展。

[0153] 本发明一实施例中，如图6所示，前端机(Frontend)将拉取到的日志存储至前端机所在地域的本地存储系统之后还包括：

[0154] 前端机接收用户的读取(get)对应分区的日志请求；

[0155] 前端机(Frontend)根据所述读取对应分区的日志请求，从所述前端机所在地域的本地存储系统获取对应分区的日志，

[0156] 若从所述本地存储系统获取到，则前端机(Frontend)将获取到的日志回复至所述用户。从而可以避免所有的非事务性操作请求都发送到后端一致性总系统中的对应分区，以减轻后端一致性总系统的工作负担，另外，通过对应地域的前端机从前端机所在地域的本地存储系统获取对应的日志也可以提高用户的读取日志请求的响应速度，从而提高读取数据性能，例如在纽约的用户可以通过纽约的前端机(Frontend)从纽约的本地存储系统读取日志，在北京的用户可以通过纽约的前端机(Frontend)从北京的本地存储系统读取日志。

[0157] 本发明一实施例中，从所述前端机所在地域的本地存储系统获取对应分区的日志之后，还包括：

[0158] 若未从所述本地存储系统获取到，通过向对应分区发送所述读取日志请求，从对应分区获取(strong read)对应的日志后，将获取到对应分区的日志回复至所述用户，本实施例是对上一实施例所述前端机从其所在地域的本地存储系统获取对应的日志的方案的优化补充，避免当用户需要读取的日志尚未拉取到对应地域的本地存储系统中的情况发生时，也能及时响应用户的日志读取请求，直接从后端一致性总系统的对应分区中读取即可。

[0159] 本发明一实施例中，所述前端机采用主/备服务器(master/slave)架构。在此，采用master/slave架构，可避免Frontend单点故障，导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新。

[0160] 本发明一实施例中，接收用户的更新日志请求或接收用户的读取对应分区的日志请求之前，包括：

[0161] 所述主服务器和备服务器向后端一致性总系统发送注册请求；

[0162] 所述主服务器和备服务器从后端一致性总系统接收注册成功信息后，所述主服务器从所述后端机获取当前已经拉取的日志在所在的分区中的位置。在此，每一个地域的Frontend包括所述主服务器和备服务器都需要注册到后端一致性总系统上，这样后端一致性总系统中的对应分区才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个Frontend启动的时候，可先调用register注册到后端一致性总系统，由于主服务器负责事务性操作，注册成功之后主服务器可调用GetTrimPoint来获取对应分区当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器下一次从位置4开始拉取日志。

[0163] 本发明一实施例中，如图6所示，由一分布式锁服务系统(Nuwa Lock Service)对所述前端机中的主服务器和备服务器进行切换。在此，为了避免Frontend单点故障，导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新了，这里采取master/slave架构，保证每次只有主服务器(master)进行事务性操作，当主服务器(master)故障之后能够通过Nuwa Lock Service重新进行选主，也就是选择master，Nuwa Lock Service就是可以提供分布式锁协议，能够适用于frontend进行master选主。

[0164] 本发明一实施例中，分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换的同时，还包括：

[0165] 下线的主服务器和备服务器，向所述分布式锁服务系统发送注销请求，从后端一致性总系统接收注销结果信息；

[0166] 新上线的主服务器和备服务器向后端一致性总系统发送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从所述后端机获取当前已经拉取的日志在所在的分区中的位置。在此，每一个地域的Frontend都需要注册到后端一致性总系统上，包括新上线的主服务器和备服务器，这样后端一致性总系统中的对应分区才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个新上线的主服务器和备服务器启动的时候，可先调用register注册到后端一致性总系统，由于只有主服务器(master)进行事务性操作，所以注册成功之后主服务器(master)可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器(master)下一次从位置4开始pull日志。另外，当下线掉Frontend包括主服务器和备服务器的时候，需要调用unregister来从后端一致性总系统中下线掉Frontend。

[0167] 本发明一实施例中，如图6所示，从一负载均衡服务器(VIP Server)获取用户的更新日志请求，通过向所述后端机发送所述用户的更新日志请求，将对应的日志更新入后端机的工作，可由所述主服务器或备服务器完成；

[0168] 而不断从对应分区上拉取日志(Pull logs)，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(Apply logs)，并向所述对应分区回复已经拉取的日志在所在的分区中的位置(Send Ack)的事务性工作，只由所述主服务器来完成，从而保证整个系统的数据一致性。

[0169] 本发明一实施例中，接收用户的读取对应分区的元数据请求，根据所述读取对应分区的元数据请求，从所述前端机所在地域的本地存储系统获取对应分区的元数据，若从

所述本地存储系统获取到,则将获取到的元数据回复至所述用户,包括:

[0170] 如图6所示,所述主服务器(master)或备服务器(slave)从所述负载均衡服务器(VIP Server)接收用户的读取对应分区的日志请求;

[0171] 所述主服务器或备服务器根据所述读取对应分区的日志请求,从所述前端机所在地域的本地存储系统获取对应分区的日志,若从所述本地存储系统获取到,则将获取到的日志回复至所述用户。负载均衡服务器(VIP Server)起到负载均衡的作用,client/user只需要访问一个IP或者域名,也就是VIP server的IP地址或者域名地址,VIP Server可以设置挂载多个Frontend,配置上每个机器上的frontend的ip和port即可。用户client/user发来的请求会由VIP Server按照某种负载均衡的策略选择其中一台主服务器或备服务器并向其发送请求。

[0172] 本发明一实施例中,未从所述本地存储系统获取到,通过向对应分区发送所述读取元数据请求,从对应分区中获取元数据后,将获取到的元数据回复至所述用户,包括:

[0173] 若未从所述本地存储系统获取到,所述主服务器或备服务器通过向对应分区发送所述读取元数据请求,从对应分区中获取元数据后,将获取到的元数据回复至所述用户。在此,非事务性的从对应分区获取对应的日志,可所述负载均衡服务器(VIP Server)选择其中一台主服务器或备服务器完成,起到更好的负载均衡的效果。

[0174] 详细地,下面讲解下各个组成部件的工作数据流向:

[0175] 图6中,有三类工作数据流向,分别用不同的线型表示,其中,第一类是前端机启动操作(Frontend bootstrap operations),包括主服务器(master)和备服务器(slave)启动操作;第二类是正常的读写操作(Normal read write operations),可以由主服务器(master)或备服务器(slave)从负载均衡服务器接收对应的请求进行操作;第三类是只有主服务器进行的操作(Master only operations),包括Pull logs、Apply logs和Send Ack。

[0176] 从Frontend角度来说,工作数据流向如下:

[0177] 1.如图6所示,前端机启动操作(Frontend bootstrap operations),也就是启动的时候,可先调用register注册到后端一致性总系统,由于主服务器负责事务性操作,注册成功之后主服务器可调用GetTrimPoint来获取对应分区当前已经删除的日志下一个位置NXID号,例如当前删除到ID号为3号的日志,则获取到当前已经拉取的日志的位置为4,即主服务器下一次从位置4开始pull日志。当下线掉Frontend的时候,需要调用unregister来从后端一致性总系统中下线掉Frontend;

[0178] 2.Pull logs与Apply logs是frontend中的一个线程来做的事情,每隔一段时间从对应分区中拖取log,然后Apply到Bulk storage System中;

[0179] 3.Send Ack是frontend Apply logs到Bulk Storage System的那个NXID,由Frontend汇报给对应分区该Frontend已经拖取下来并成功写到Bulk Storage中的log的最后一个ID号的下一个ID号即NXID;

[0180] 4.Frontend接收client/user发来的put/get/delete请求,put/delete这些事务性请求需要发送到对应分区中,get是非事务性操作发往Bulk Storage中。

[0181] 从分区角度来说,工作数据流向如下:

[0182] 1.存储Frontend发送来的put请求,将数据以log的形式写到分区中;

[0183] 2.与Frontend进行pull与ack的交互,将log发给Frontend,以及回复现在Frontend发给对应分区的ack成功的NXID;

[0184] 3.当client/user发给Frontend的read请求,Frontend如果从Bulk Storage中没有获取到,那么就需要给对应分区发strong read请求,直接去对应分区获取数据。

[0185] 从Bulk Storage System角度来说,工作数据流向如下:

[0186] Bulk Storage System是永久性存储系统,这个存储全量的log数据,Frontend与Bulk Storage交互,传输log以及获取log。

[0187] 具体API协议制定为的规则包括:所有的Response返回信息中都会有字段rc,message,traceinfo,rc代表返回码,message代表返回的信息,traceinfo可以跟踪快慢请求的作用,具体来说,

[0188] 1.Register与UnRegister的Request与Response

[0189] Register与UnRegister的Request与Response进行avro序列化,Frontend启动的时候会向后端一致性总系统注册一个frontendID,因为每个region都会有多个frontend,所以NuwaLog后端才知道将信息发送给那个frontend。

[0190] 2.GetTrimPoint的Request与Response

[0191] GetTrimPoint的Request与Response进行avro序列化,主服务器启动的时候会去NuwaLog后端获取Trim Point的log的NXID,这个在frontend发生failover的时候,新上线的主服务器也会去获取Trim Point点。这个Trim Point点供主服务器的Pull线程使用,pull线程凭借此Trim Point点去后端获取Log信息。

[0192] 3.Submit的Request与Response

[0193] Submit的Request与Response进行avro序列化。对于Submit请求,request在frontend这端会组装成一个事务性Log请求,response返回返回码以及返回信息。

[0194] 4.Pull的Request与Response

[0195] Pull的Request与Response进行avro序列化。Frontend会启动一个pull线程定时去NuwaLog上拿log序列数组,request是当前的log的Nxid,控制流量的最大log数量和最大log字节。response是返回码以及返回信息,log数组信息(log里面包括该log的nxid,log的内容)。

[0196] 5.Ack的Request(请求)与Response(响应)

[0197] Ack的Request与Response进行avro序列化。Frontend也会启动一个ack线程定时从Pull线程拉取的log数据队列,从该队列中拉数据到本地OTS中,当commit(确认)成功之后,去向对应分区确认一个信息,request是commit到本地OTS成功的最大的nxid,response是返回码以及返回信息。

[0198] 上述五个API可以通过restful协议来传输,所以需要制定标准的http协议头与协议体。如下:

[0199] Request示例中,所使用的每个字段的含义是:

[0200] 1.URL:标明发往的服务端的ip和端口号port;

[0201] 2.REQUEST:标明这是个HTTP POST请求,并指明访问的资源路径path;

[0202] 3.Http Headers:是http头部,这里面需要标明协议是avro,其中NuwaNeuron-Action是标明这个http请求发起的行为,这里面几个行为是:Submit,Pull,Ack,Register,

UnRegister。NuwaNeuron-Timestamp标明这个http请求发起的时间戳,NuwaNeuron-API-Version标明这个http请求的版本号。NuwaNeuron-PackageId标明这个http请求发起的包的序号。NuwaNeuron-RequestId标明这个http请求发起的ID号。

[0203] Response示例中,所使用的每个字段的含义是:http请求返回的response,其中包括response header以及http body。header中包括Content-type,标明这个请求内容的类型,这里是avro。Content-length标明http返回请求内容的长度。NuwaNeuron-Timestamp标明这个http response的时间戳,NuwaNeuron-RequestId标明这个http请求的requestID号,同前述。

[0204] 根据本发明的另一面,还提供一种后端机,如图3和6所示,包括:

[0205] 分区装置,用于将多个后端机组成后端一致性总系统,将后端机分成多个分区,为不同的用户分配对应的分区;

[0206] 更新装置,用于从前端机接收用户的更新对应分区的日志请求,根据所述请求将对应的日志更新入对应分区;

[0207] 发送装置,向所述前端机发送日志,并从所述前端机接收当前已经拉取的日志在所在的分区中的位置,根据所述所在的分区中的位置删除对应分区中所述已经拉取的日志。

[0208] 在此,对应分区会接收所有前端机(Frontend)发出的请求,会存储Log。前端机(Frontend)可以启动两个线程,一个线程不断从某个NuwaLog中的对应分区上拉Log数据,另一个线程不断往OTS(Bulk Storage)里面写数据,并回复对应分区已经pull下来的位置,对应分区这边会根据接收到的位置删除掉(trim)所有前端机(Frontend)回复的多个位置中已经pull下来的log的最小ID号的下一个ID号即NXID号之前所有的log,例如目前有三台前端机A、B和C,另后端对应分区中原来有10条日志,各条日志对应为1～10的ID号,各前端机拉取日志均从最小的ID号1开始拉取,若当前端机A拉取到了ID号为3的日志后,向对应分区回复NXID号位置为4,此时前端机B拉取到了ID号为4的日志后,向对应分区回复NXID号位置为5,此时前端机C拉取到了ID号为5的日志后,向对应分区回复NXID号位置为6,那么后续对应分区仅会将ID号为4之前的ID号为1～3的已经拉取的日志从其存储空间内删除,因为ID号为4～5的日志有部分前端机没有拉取完,需要待所有前端机拉取完后才能删除。

[0209] 上述实施例通过将多个后端机组成一个后端一致性总系统,将后端机分成多个分区,各个分区分别响应对应的请求,分担请求压力,避免由同一个后端机响应所有请求,导致压力过大的问题,实现了后端机的水平扩展;另外,通过为不同的用户分配对应的分区,能够给每个用户所写的日志资源空间做Quota配额,而且用户之间的日志得到分区与隔离,安全性也得到保证,也便于区分不同的用户进行Quota配额的计费;此外,通过前端机(Frontend)不断从某个后端机(NuwaLog)的对应分区上拉取日志,并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS),向对应分区回复当前已经拉取的日志的位置,所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志,这样对应分区不需要存储所有的日志,只需要存储还未拉取到本地存储系统(OTS),可以避免后端机(NuwaLog)的单机存储性能的下降和存储容量的限制,数据存储站点数量受一致性协议Quorum节点数的限制的问题,达到整个系统的数据一致性、正确性与健壮性。

[0210] 本发明一实施例的后端机中,所述发送装置,还用于从所述前端机获取用户的读

取对应分区的日志请求，根据所述读取日志请求向所述前端机发送对应的日志。本实施例是对所述前端机从其所在地域的本地存储系统获取对应的日志的方案的优化补充，避免当用户需要读取的日志尚未拉取到对应地域的本地存储系统中的情况发生时，也能及时响应用户的日志读取请求，直接从对应分区中读取即可。

[0211] 本发明一实施例的后端机中，所述前端机采用主/备服务器架构。在此，采用master/slave架构，可避免Frontend单点故障，导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新。

[0212] 本发明一实施例的后端机中，还包括注册装置，用于：从所述主服务器和备服务器接收注册请求；根据所述注册请求对所述主服务器和备服务器进行注册后，向所述主服务器和备服务器发送注册成功信息和向主服务器发送当前已经拉取的日志在所在的分区中的位置。在此，每一个地域的Frontend包括所述主服务器和备服务器都需要注册到后端一致性总系统上，这样后端一致性总系统中的分区才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个Frontend启动的时候，可先调用register注册到NuwaLog，由于主服务器负责事务性操作，注册成功之后主服务器可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器下一次从位置4开始拉取日志。

[0213] 本发明一实施例的后端机中，由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。在此，为了避免Frontend单点故障，导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新了，这里采取master/slave架构保证每次只有主服务器(master)进行事务性操作，当主服务器(master)故障之后能够通过Nuwa Lock Service重新进行选主，也就是选择master，Nuwa Lock Service可以提供分布式锁协议，能够适用于frontend进行master选主，所述分布式锁服务系统(Nuwa Lock Service)可以是一个全球化的分布式锁服务系统(Nuwa Global Lock Service)，可以对全球master/slave架构进行管理。

[0214] 本发明一实施例的后端机中，如图6所示，所述注册装置，还用于从下线的主服务器和/或备服务器接收注销请求，根据所述注销请求将下线的主服务器和/或备服务器进行注销后，向所述下线的主服务器和/或备服务器发送注销结果信息；

[0215] 所述注册装置，还用于从新上线的主服务器和/或备服务器接收注册请求，根据所述注册请求将所述新上线的主服务器和/或备服务器进行注册后，向所述新上线的主服务器和/或备服务器发送注册成功信息和向新上线的主服务器发送当前已经拉取的日志在所在的分区中的位置。在此，每一个地域的Frontend都需要注册到NuwaLog上，包括新上线的主服务器和备服务器，这样后端一致性总系统才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个新上线的主服务器和备服务器启动的时候，可先调用register注册到后端一致性总系统，由于只有主服务器(master)进行事务性操作，所以注册成功之后主服务器(master)可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器(master)下一次从位置4开始pull日志。另外，当下线掉Frontend包括主服务器和备服务器的时候，需要调用unregister来从后端一致性总系统中下线掉Frontend。

[0216] 本发明一实施例的后端机中，所述更新装置，还用于将对应的日志更新入对应分

区的同时，根据所述分区中当前存储的日志，生成对应快照并存储入所述分区；

[0217] 所述发送装置，还用于根据所述所在的分区中位置删除所述分区中已经拉取的日志的同时，从所述分区中删除对应的快照。在此，Snapshot为快照，也就是对应分区中某一时刻内存中全量数据的快照。对应分区会接收frontend的请求，会存储log和snapshot，这里存储snapshot是为了解决该分区发生故障切换(failover)的情况，而且log数据还没有传到本地的OTS中，该分区重新启动能够从snapshot中恢复还没有传输到OTS中的数据，该分区根据所述位置删除所述已经拉取的日志的同时，每隔一段时间删除对应磁盘上的snapshot文件，避免存储过期无用的snapshot。

[0218] 本发明一实施例中，前端(frontend)与后端(NuwaLog)之间可以采用avro序列化方式，利用Netty作为RPC通信。这里选择使用avro协议的原因是该序列化协议比较简单并高效。Netty作为业界通用的RPC协议，在JAVA领域是一个成熟度并高效的远程网络通信协议。

[0219] 根据本发明的另一面，还提供一种前端机，该前端机包括：

[0220] 发起更新装置，用于接收用户的更新对应分区的日志请求，通过向对应分区发送所述请求，将日志更新入对应分区中，其中，不同的用户分配有对应的分区，后端机分成多个分区，多个后端机组成后端一致性总系统；

[0221] 拉取装置，从对应分区拉取日志，并将拉取到的日志存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的日志所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除其内部存储的所述已经拉取的日志。

[0222] 在此，对应分区会接收所有前端机(Frontend)发出的请求，会存储Log。前端机(Frontend)可以启动两个线程，一个线程不断从对应分区上拉Log数据，另一个线程不断往OTS里面写数据，并回复对应分区已经pull下来的位置，对应分区那边会根据接收到的位置删除掉(trim)所有前端机(Frontend)回复的多个位置中已经pull下来的log的最小ID号的下一个ID号即NXID号之前所有的log，例如目前有三台前端机A、B和C，另后端NuwaLog中原来有10条日志，各条日志对应为1～10的ID号，各前端机拉取日志均从最小的ID号1开始拉取，若当前端机A拉取到了ID号为3的日志后，向NuwaLog回复NXID号位置为4，此时前端机B拉取到了ID号为4的日志后，向NuwaLog回复NXID号位置为5，此时前端机C拉取到了ID号为5的日志后，向NuwaLog回复NXID号位置为6，那么后续NuwaLog仅会将ID号为4之前的ID号为1～3的已经拉取的日志从其存储空间内删除，因为ID号为4～5的日志有部分前端机没有拉取完，需要待所有前端机拉取完后才能删除。

[0223] 本实施例通过将多个后端机组成一个后端一致性总系统，将后端机分成多个分区，各个分区分别响应对应的请求，分担请求压力，避免由同一个后端机响应所有请求，导致压力过大的问题，实现了后端机的水平扩展；另外，通过为不同的用户分配对应的分区，能够给每个用户所写的元数据资源空间做Quota配额，而且用户之间的元数据得到分区与隔离，安全性也得到保证，也便于区分不同的用户进行Quota配额的计费；此外，通过前端机(Frontend)不断从所述对应分区上拉取日志，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS)，向所述对应分区回复当前已经拉取的日志的位置，所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志，这样对应分区不需要存储所有的日志，只需要存储还未拉取到本地存储系统(OTS)，可以避免对应分区的单机存储性能的下降

和存储容量的限制,数据存储站点数量受一致性协议Quorum节点数的限制的问题,达到整个系统的数据一致性、正确性与健壮性,另外,前端机(Frontend)和本地存储系统(OTS)可以实现按需要进行扩展。

[0224] 本发明一实施例的前端机中,还包括读取装置,用于接收用户的读取对应分区的日志请求;根据所述读取对应分区的日志请求,从所述前端机所在地域的本地存储系统获取对应分区的日志,

[0225] 若从所述本地存储系统获取到,则将获取到对应分区的日志回复至所述用户。从而可以避免所有的非事务性操作请求都发送到后端一致性总系统中的对应分区,以减轻后端一致性总系统的工作负担,另外,通过对应地域的前端机从前端机所在地域的本地存储系统获取对应的日志也可以提高用户的读取日志请求的响应速度,从而提高读取数据性能,例如在纽约的用户可以通过纽约的前端机(Frontend)从纽约的本地存储系统读取日志,在北京的用户可以通过纽约的前端机(Frontend)从北京的本地存储系统读取日志。

[0226] 本发明一实施例的前端机中,所述读取装置,还用于若未从所述本地存储系统获取到,通过向对应分区发送所述读取日志请求,从对应分区中获取日志后,将获取到对应分区的日志回复至所述用户。本实施例是对上一实施例所述前端机从其所在地域的本地存储系统获取对应的日志的方案的优化补充,避免当用户需要读取的日志尚未拉取到对应地域的本地存储系统中的情况发生时,也能及时响应用户的日志读取请求,直接从后端一致性总系统的对应分区中读取即可。

[0227] 本发明一实施例的前端机中,所述前端机采用主/备服务器架构。在此,采用master/slave架构,可避免Frontend单点故障,导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新。

[0228] 本发明一实施例的前端机中,还包括发起注册装置,用于供所述主服务器和备服务器向后端一致性总系统发送注册请求;所述主服务器和备服务器从后端一致性总系统接收注册成功信息后,所述主服务器从对应分区获取当前已经拉取的日志在所在的分区中的位置。在此,每一个地域的Frontend包括所述主服务器和备服务器都需要注册到后端一致性总系统上,这样后端一致性总系统中的对应分区才能够找到对应的Frontend发送请求结果,具体来说,如图6所示,每个Frontend启动的时候,可先调用register注册到后端一致性总系统,由于主服务器负责事务性操作,注册成功之后主服务器可调用GetTrimPoint来获取对应分区当前已经删除的日志下一个位置ID号,例如当前删除到ID号为3号的日志,则获取到当前已经拉取的日志的位置为4,即主服务器下一次从位置4开始拉取日志。

[0229] 本发明一实施例的前端机中,由一分布式锁服务系统对所述前端机中的主服务器和备服务器进行切换。在此,为了避免Frontend单点故障,导致本地域(region)的本地存储系统(Bulk storage system)中的日志数据得不到更新了,这里采取master/slave架构,保证每次只有主服务器(mas ter)进行事务性操作,当主服务器(master)故障之后能够通过Nuwa Lock Service重新进行选主,也就是选择master,Nuwa Lock Service就是可以提供分布式锁协议,能够适用于frontend进行master选主。

[0230] 本发明一实施例的前端机中,还包括发起注销装置,用于下线的主服务器和备服务器,向所述分布式锁服务系统发送注销请求,从后端一致性总系统接收注销结果信息;

[0231] 所述发起注册装置,还用于新上线的主服务器和备服务器向后端一致性总系统发

送注册请求，并从后端一致性总系统接收注册成功信息后，新上线的主服务器从对应分区获取当前已经拉取的日志在所在的分区中的位置。在此，每一个地域的Frontend都需要注册到后端一致性总系统上，包括新上线的主服务器和备服务器，这样后端一致性总系统中的对应分区才能够找到对应的Frontend发送请求结果，具体来说，如图6所示，每个新上线的主服务器和备服务器启动的时候，可先调用register注册到后端一致性总系统，由于只有主服务器(master)进行事务性操作，所以注册成功之后主服务器(master)可调用GetTrimPoint来获取后端NuwaLog当前已经删除的日志下一个位置ID号，例如当前删除到ID号为3号的日志，则获取到当前已经拉取的日志的位置为4，即主服务器(master)下一次从位置4开始pull日志。另外，当下线掉Frontend包括主服务器和备服务器的时候，需要调用unregister来从后端一致性总系统中下线掉Frontend。

[0232] 本发明一实施例的前端机中，所述发起更新装置，用于供所述主服务器或备服务器从一负载均衡服务器获取用户的更新对应分区的日志请求，所述主服务器或备服务器通过向所述后端机发送所述用户的更新对应分区的日志请求，将日志更新入对应分区；

[0233] 所述拉取装置，用于所述主服务器从所述对应分区拉取日志，并将拉取到的日志存储至前端机所在地域的本地存储系统，并向所述对应分区回复已经拉取的日志在所在的分区中的位置。从一负载均衡服务器(VIP Server)获取用户的更新日志请求，通过向所述后端机发送所述用户的更新日志请求，将对应的日志更新入后端机的工作，可由所述主服务器或备服务器完成；

[0234] 而不断从对应分区上拉取日志(Pull logs)，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(Apply logs)，并向所述对应分区回复已经拉取的日志在所在的分区中的位置(Send Ack)的事务性工作，只由所述主服务器来完成，从而保证整个系统的数据一致性。

[0235] 本发明一实施例的前端机中，所述读取装置用于供所述主服务器或备服务器从所述负载均衡服务器接收用户的读取对应分区的日志请求；及所述主服务器或备服务器根据所述读取对应分区的日志请求，从所述前端机所在地域的本地存储系统获取对应分区的日志，若从所述本地存储系统获取到，则将获取到的日志回复至所述用户。负载均衡服务器(VIP Server)起到负载均衡的作用，client/user只需要访问一个IP或者域名，也就是VIP server的IP地址或者域名地址，VIP Server可以设置挂载多个Frontend，配置上每个机器上的frontend的ip和port即可。用户client/user发来的请求会由VIP Server按照某种负载均衡的策略选择其中一台主服务器或备服务器并向其发送请求。

[0236] 本发明一实施例的前端机中，所述读取装置，还用于若未从所述本地存储系统获取到，所述主服务器或备服务器通过向对应分区发送所述读取日志请求，从对应分区中获取日志后，将获取到的日志回复至所述用户。在此，非事务性的从对应分区获取对应的日志，可所述负载均衡服务器(VIP Server)选择其中一台主服务器或备服务器完成，起到更好的负载均衡的效果。

[0237] 根据本发明的另一面，还提供一种基于计算的设备，包括：

[0238] 处理器；以及

[0239] 被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

[0240] 将多个后端机组成后端一致性总系统，将后端机分成多个分区，为不同的用户分配对应的分区；

[0241] 从前端机接收用户的更新对应分区的日志请求，根据所述请求将对应的日志更新入对应分区；

[0242] 向所述前端机发送日志，并从所述前端机接收当前已经拉取的日志在所在的分区中的位置，根据所述所在的分区中的位置删除对应分区中所述已经拉取的日志

[0243] 根据本发明的另一面，还提供一种基于计算的设备，包括：

[0244] 处理器；以及

[0245] 被安排成存储计算机可执行指令的存储器，所述可执行指令在被执行时使所述处理器：

[0246] 接收用户的更新对应分区的日志请求，通过向对应分区发送所述请求，将日志更新入对应分区中，其中，不同的用户分配有对应的分区，后端机分成多个分区，多个后端机组成后端一致性总系统；

[0247] 从对应分区拉取日志，并将拉取到的日志存储至前端机所在地域的本地存储系统，并向所述对应分区回复当前已经拉取的日志所在的分区中的位置，以供所述分区根据所述所在的分区中的位置删除其内部存储的所述已经拉取的日志。

[0248] 本发明可应用于现在电商类的很多业务系统，比如网络交易平台的用户信息等，此时的日志的内容可为用户信息，都需要满足用户随处可以快速访问到自己的淘宝信息，这种需求就需要一个分布式的跨地域的中心化的数据一致性，满足每个区域的数据访问；本发明还可应用于金融业务中，也同样有这样的需求，此时的日志的内容可为金融数据，用户的金融数据需要在多个region进行备份，满足金融数据的高可靠。

[0249] 综上所述，本发明通过将多个后端机组成一个后端一致性总系统，将后端机分成多个分区，各个分区分别响应对应的请求，分担请求压力，避免由同一个后端机响应所有请求，导致压力过大的问题，实现了后端机的水平扩展，另外，通过为不同的用户分配对应的分区，能够给每个用户所写的元数据资源空间做Quota配额，而且用户之间的元数据得到分区与隔离，安全性也得到保证，也便于区分不同的用户进行Quota配额的计费，此外，通过前端机(Frontend)不断从某个后端机(NuwaLog)的对应分区上拉取日志，并不断将拉取到的日志存储至前端机所在地域的本地存储系统(OTS)，向对应分区回复当前已经拉取的日志的位置，所述对应分区根据所述位置删除其内部存储的所述已经拉取的日志，这样对应分区不需要存储所有的日志，只需要存储还未拉取到本地存储系统(OTS)，可以避免后端机(NuwaLog)的单机存储性能的下降和存储容量的限制，数据存储站点数量受一致性协议Quorum节点数的限制的问题，达到整个系统的数据一致性、正确性与健壮性。

[0250] 显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。

[0251] 需要注意的是，本发明可在软件和/或软件与硬件的组合体中被实施，例如，可采用专用集成电路(ASIC)、通用目的计算机或任何其他类似硬件设备来实现。在一个实施例中，本发明的软件程序可以通过处理器执行以实现上文所述步骤或功能。同样地，本发明的软件程序(包括相关的数据结构)可以被存储到计算机可读记录介质中，例如，RAM存储器，

磁或光驱动器或软磁盘及类似设备。另外，本发明的一些步骤或功能可采用硬件来实现，例如，作为与处理器配合从而执行各个步骤或功能的电路。

[0252] 另外，本发明的一部分可被应用为计算机程序产品，例如计算机程序指令，当其被计算机执行时，通过该计算机的操作，可以调用或提供根据本发明的方法和/或技术方案。而调用本发明的方法的程序指令，可能被存储在固定的或可移动的记录介质中，和/或通过广播或其他信号承载媒体中的数据流而被传输，和/或被存储在根据所述程序指令运行的计算机设备的工作存储器中。在此，根据本发明的一个实施例包括一个装置，该装置包括用于存储计算机程序指令的存储器和用于执行程序指令的处理器，其中，当该计算机程序指令被该处理器执行时，触发该装置运行基于前述根据本发明的多个实施例的方法和/或技术方案。

[0253] 对于本领域技术人员而言，显然本发明不限于上述示范性实施例的细节，而且在不背离本发明的精神或基本特征的情况下，能够以其他的具体形式实现本发明。因此，无论从哪一点来看，均应将实施例看作是示范性的，而且是非限制性的，本发明的范围由所附权利要求而不是上述说明限定，因此旨在将落在权利要求的等同要件的含义和范围内的所有变化涵括在本发明内。不应将权利要求中的任何附图标记视为限制所涉及的权利要求。此外，显然“包括”一词不排除其他单元或步骤，单数不排除复数。装置权利要求中陈述的多个单元或装置也可以由一个单元或装置通过软件或者硬件来实现。第一，第二等词语用来表示名称，而并不表示任何特定的顺序。

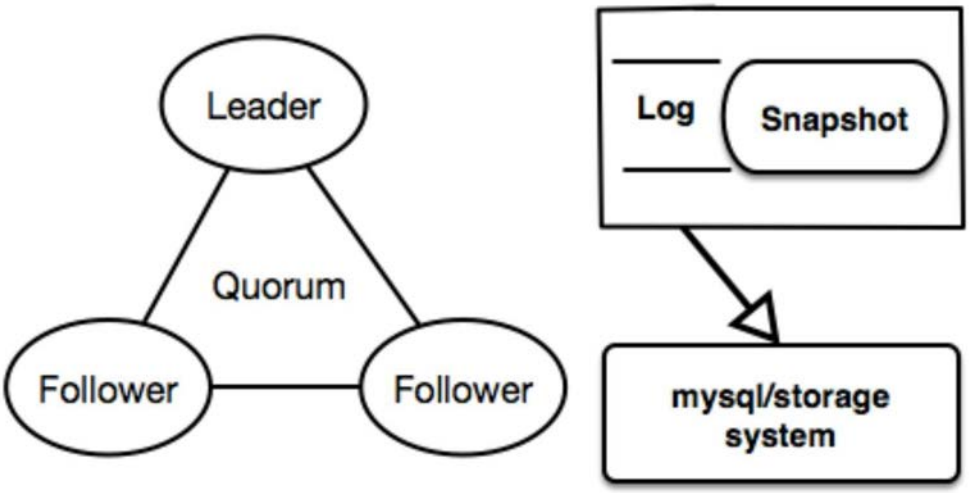

Leader
Quorum
Follower
Follower
Log
Snapshot
mysql/storage system

图1

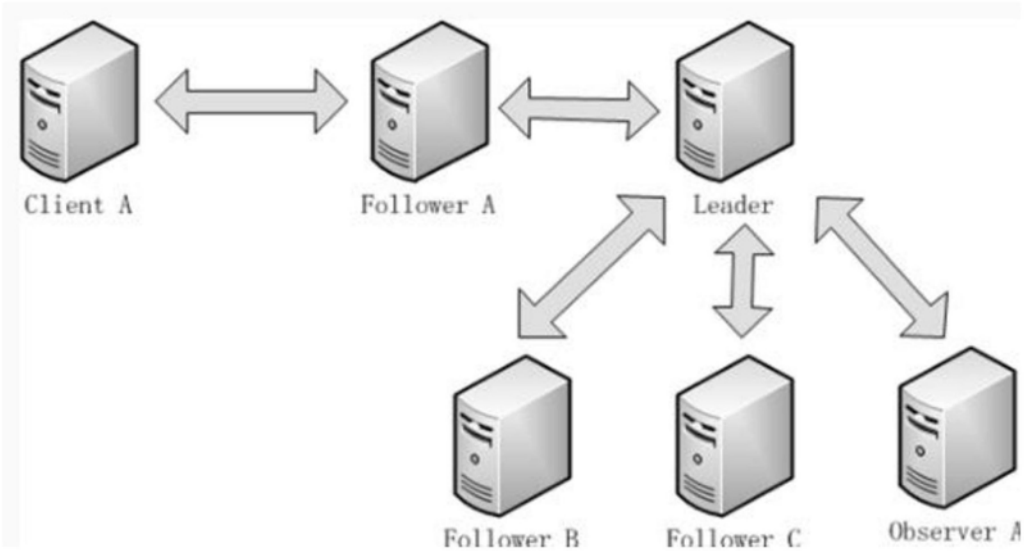

Client A
Follower A
Leader
Follower B
Follower C
Observer A

图2

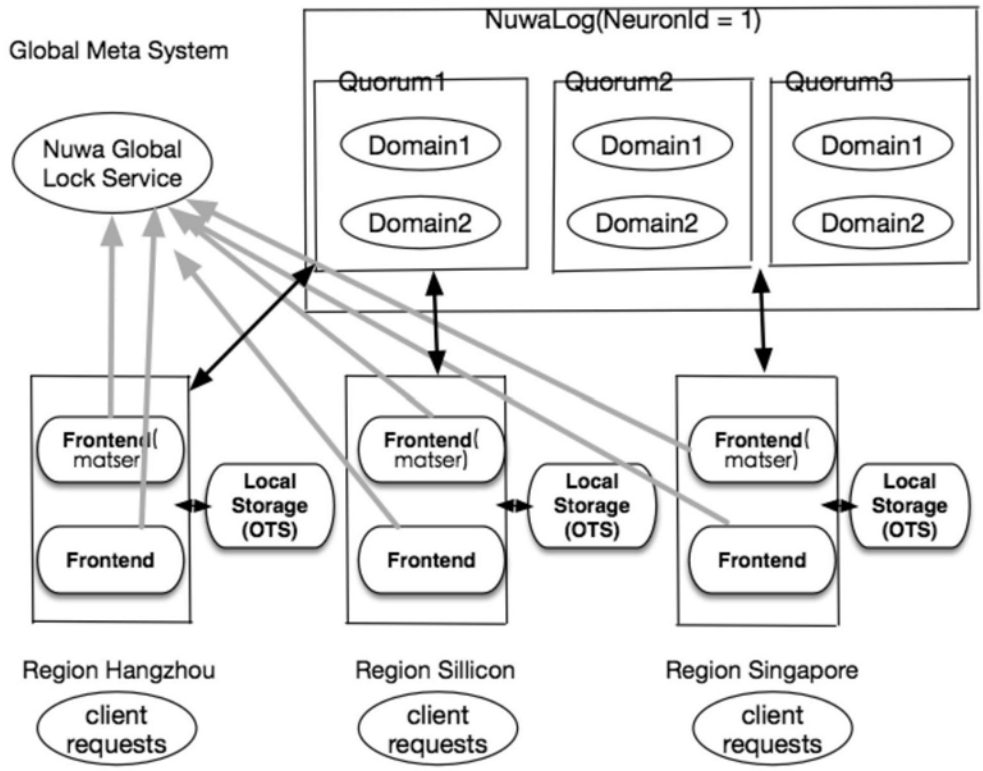

Global Meta System
NuwaLog(NeuronId = 1)
Quorum1
Quorum2
Quorum3
Domain1
Domain2
Nuwa Global Lock Service
Frontend(matser)
Frontend
Local Storage (OTS)
Region Hangzhou
Region Sillicon
Region Singapore
client requests

图3

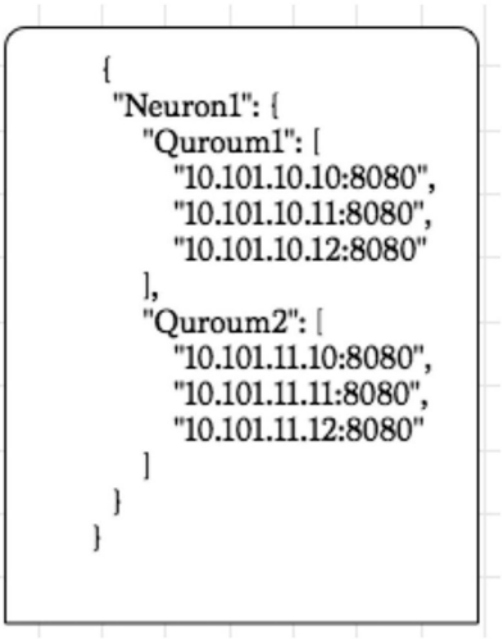

{
"Neuron1": {
"Quroum1": [
"10.101.10.10:8080",
"10.101.10.11:8080",
"10.101.10.12:8080"
],
"Quroum2": [
"10.101.11.10:8080",
"10.101.11.11:8080",
"10.101.11.12:8080"
]
}
}

图4

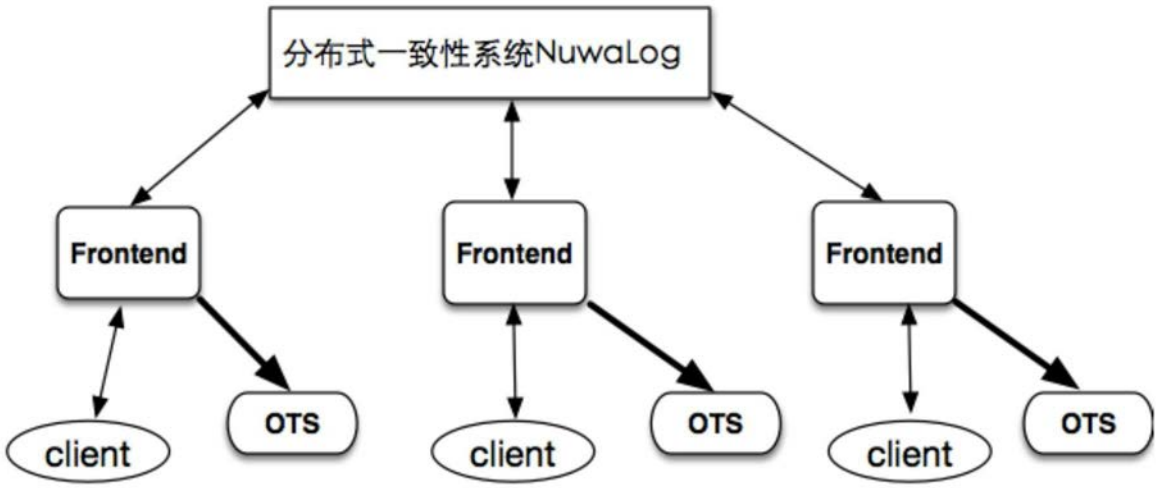

分布式一致性系统NuwaLog
Frontend
Frontend
Frontend
client
OTS
client
OTS
client
OTS

图5

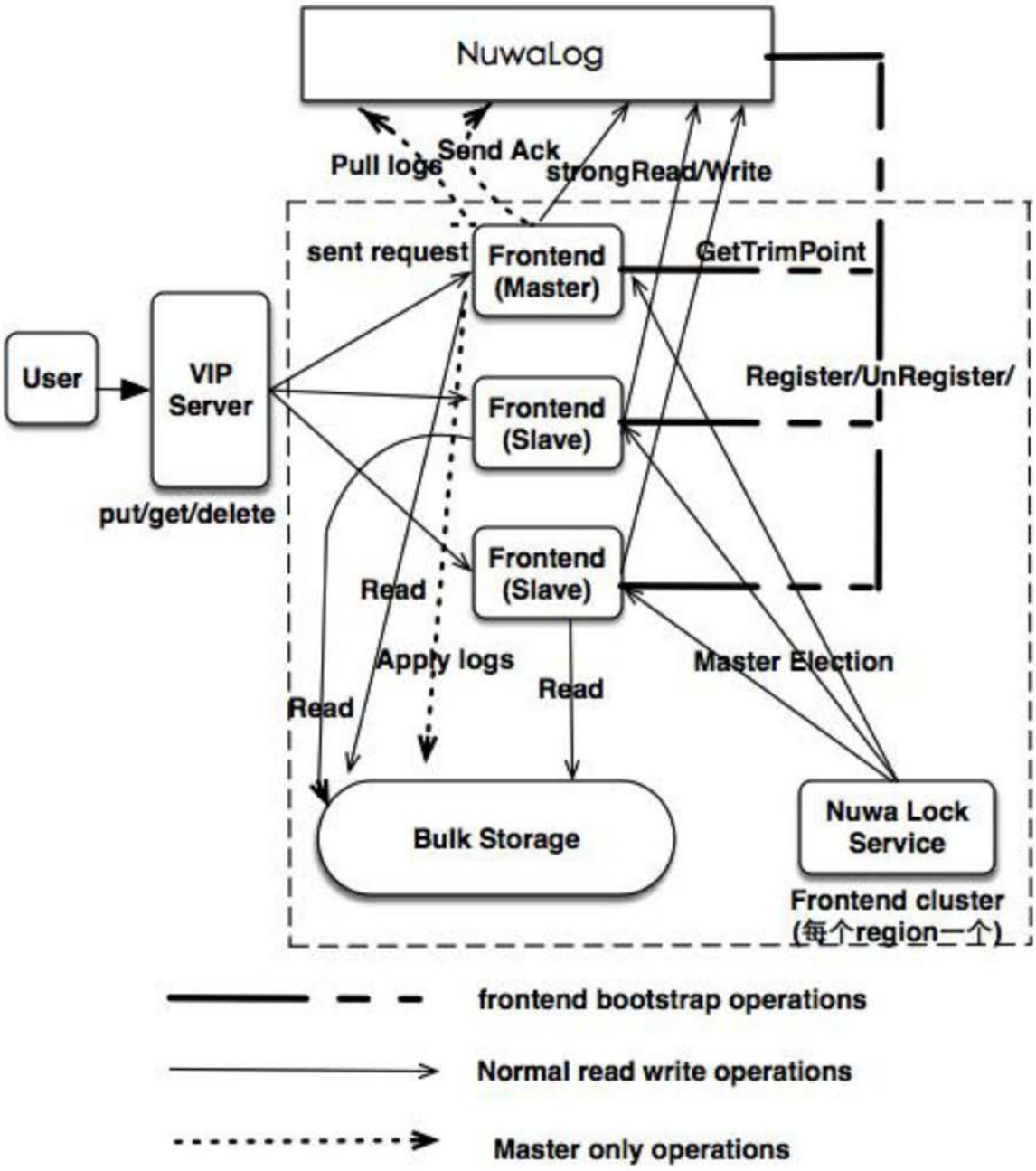

NuwaLog
Pull logs
Send Ack
strongRead/Write
sent request
Frontend (Master)
GetTrimPoint
User
VIP Server
Register/UnRegister/
Frontend (Slave)
put/get/delete
Frontend (Slave)
Read
Apply logs
Master Election
Read
Read
Bulk Storage
Nuwa Lock Service
Frontend cluster (每个region一个)
frontend bootstrap operations
Normal read write operations
Master only operations

图6