(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 106716412 B
(45)授权公告日 2020.08.14

(21)申请号 201580051004.5

(22)申请日 2015.09.25

(65)同一申请的已公布的文献号
申请公布号 CN 106716412 A

(43)申请公布日 2017.05.24

(30)优先权数据
62/055,483 2014.09.25 US
62/055,494 2014.09.25 US

(85)PCT国际申请进入国家阶段日
2017.03.22

(86)PCT国际申请的申请数据
PCT/US2015/052458 2015.09.25

(87)PCT国际申请的公布数据
WO2016/049575 EN 2016.03.31

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 H·拉加 C·普尔蒂 G·格莱泽

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 李晓芳

(51)Int.Cl.
G06F 16/22(2019.01)

(56)对比文件
US 6694323 B2,2004.02.17,
US 2010011283 A1,2010.01.14,
US 5857196 A,1999.01.05,
CN 1316696 A,2001.10.10,

审查员 黄端

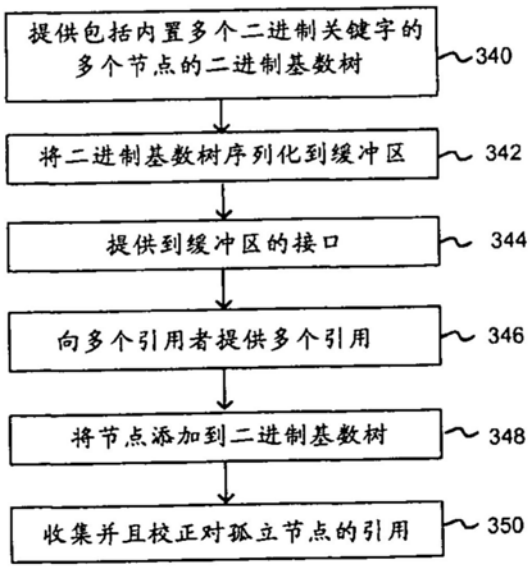
权利要求书3页 说明书19页 附图7页

(54)发明名称

用于支持分布式计算环境中的零拷贝二进制基数树的系统和方法

(57)摘要

一种系统和方法支持诸如分布式数据网格之类的分布式计算环境中的关键字管理。二进制基数树被用于内置多个二进制关键字。二进制基数树被序列化到字节缓冲区，并且二进制数据的视图被创建。到序列化的二进制基数树的节点的字节序列接口允许使用引用序列化的二进制基数树中的位置的引用，而不需要内置关键字的字节数组拷贝。使用对字节数组的引用来代替内置关键字的字节数组拷贝减少了与引用者(诸如引用与多个二进制关键字相关联的值的反向索引)相关联的存储器开销。存储器开销的减少增强了诸如分布式数据网格之类的分布式计算环境的性能和能力。



CN 106716412 B

1.一种用于支持分布式计算环境中的关键字管理的方法,所述方法包括:

提供包括内置多个二进制关键字的多个节点的二进制基数树,其中每个二进制关键字用于检索对应的对象、值或数据;

将所述二进制基数树序列化到缓冲区;

提供到所述缓冲区的接口,其中所述接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述二进制基数树的节点以及所述多个二进制关键字中的二进制关键字相关联;以及

向多个引用者提供所述多个引用,所述多个引用者通过维护对所述二进制基数树中的节点的引用来使用所述二进制基数树检索与二进制关键字对应的值,由此使得所述多个引用者能够从所述多个引用实现所述多个二进制关键字并且定位与所述多个二进制关键字对应的值,而无需维护所述多个二进制关键字的拷贝。

2.如权利要求1所述的方法,还包括:

在所述二进制基数树中提供多个卫星节点,其中所述多个卫星节点利用指示它们在所述二进制基数树中的位置的信息被增强。

3.如权利要求1所述的方法,还包括:

在所述二进制基数树中提供多个卫星节点,其中所述多个卫星节点利用指示它们在所述二进制基数树中的位置的位置信息被增强;以及

通过在朝正确的叶子节点遍历之前向上遍历到卫星节点,使用卫星节点中编码的所述位置信息来提高所述二进制基数树的遍历的效率。

4.如权利要求1至3中任何一项所述的方法,其中提供到所述缓冲区的接口包括:

提供到所述缓冲区的字节序列接口,其中所述字节序列接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联。

5.如权利要求1至3中任何一项所述的方法,其中提供到所述缓冲区的接口包括:

提供到所述缓冲区的字节序列接口,其中所述字节序列接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联;以及

利用所述字节序列接口支持byteAt(n)方法,其中所述byteAt(n)方法返回所述字节序列中的第n个位置处的字节。

6.如权利要求1至3中任何一项所述的方法,其中提供到所述缓冲区的接口包括:

提供到所述缓冲区的字节序列接口,其中所述字节序列接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联;以及

利用所述字节序列接口支持长度方法,其中所述长度方法能够确定所述字节序列的节点中表示的数据的字节数。

7.如权利要求1至3中任何一项所述的方法,其中:

所述多个引用者包括多个反向索引,并且其中每个反向索引包括引用集合。

8.如权利要求1至3中任何一项所述的方法,其中:

所述多个引用者包括多个反向索引,并且由此所述多个引用者实现二进制关键字并且

根据所述多个反向索引来定位与二进制关键字相关联的值，而无需维护二进制关键字的拷贝。

9.如权利要求1至3中任何一项所述的方法，还包括：

将所述二进制基数树中的每个节点与和所述节点相关联的关键字所关联的值的地址相关联。

10.如权利要求1至3中任何一项所述的方法，还包括：

提供长整型存储数组，所述长整型存储数组保持用于与所述多个二进制关键字相关联的值的多个地址；以及

将所述二进制基数树中的每个节点与所述长整型存储数组中的槽相关联，从而将每个节点与和所述节点相关联的关键字所关联的值的地址相关联。

11.一种用于支持分布式计算环境中的关键字管理的系统，所述系统包括：

计算机系统，所述计算机系统包括微处理器和存储器；

在所述计算机系统上操作的所述分布式计算环境的服务器节点；以及

其中，所述服务器节点被配置为：

提供包括内置多个二进制关键字的多个节点的二进制基数树，其中每个二进制关键字用于检索对应的对象、值或数据；

将所述二进制基数树序列化到所述计算机系统的存储器中的被分配给所述节点的缓冲区；

提供到所述缓冲区的接口，其中所述接口被配置为提供多个引用，其中所述多个引用中的每个引用与所述二进制基数树的节点以及所述多个二进制关键字中的二进制关键字相关联，以及

向多个引用者提供所述多个引用，所述多个引用者通过维护对所述二进制基数树中的节点的引用来使用所述二进制基数树检索与二进制关键字对应的值，由此使得所述多个引用者能够从所述多个引用实现所述多个二进制关键字并且定位与所述多个二进制关键字对应的值，而无需维护所述多个二进制关键字的拷贝。

12.如权利要求11所述的系统，其中所述服务器节点还被配置为在所述二进制基数树中提供多个卫星节点，其中所述多个卫星节点利用指示它们在所述二进制基数树中的位置的信息被增强。

13.如权利要求11所述的系统，其中所述服务器节点还被配置为：

在所述二进制基数树中提供多个卫星节点，其中所述多个卫星节点利用指示它们在所述二进制基数树中的位置的位置信息被增强；以及

通过在朝正确的叶子节点遍历之前向上遍历到卫星节点，使用卫星节点中编码的所述位置信息来提高所述二进制基数树的遍历的效率。

14.如权利要求11至13中任何一项所述的系统，其中到所述缓冲区的所述接口包括：

到所述缓冲区的字节序列接口，其中所述字节序列接口被配置为提供多个引用，其中所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联。

15.如权利要求11至13中任何一项所述的系统，其中到所述缓冲区的所述接口包括：

到所述缓冲区的字节序列接口，其中所述字节序列接口被配置为提供多个引用，其中

所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联;以及

其中所述字节序列接口被配置为利用所述字节序列接口支持byteAt(n)方法,其中所述byteAt(n)方法返回所述字节序列中的第n个位置处的字节。

16.如权利要求11至13中任何一项所述的系统,其中到所述缓冲区的所述接口包括:

到所述缓冲区的字节序列接口,其中所述字节序列接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述字节序列接口中的所述二进制基数树的节点的位置以及所述多个二进制关键字中的二进制关键字相关联;以及

其中所述字节序列接口被配置为利用所述字节序列接口支持长度方法,其中所述长度方法可以确定所述字节序列的节点中表示的数据的字节数。

17.如权利要求11至13中任何一项所述的系统,其中:

所述多个引用者包括多个反向索引,并且由此所述多个引用者实现二进制关键字并且根据所述多个反向索引来定位与二进制关键字相关联的值,而无需维护二进制关键字的拷贝。

18.如权利要求11至13中任何一项所述的系统,其中所述二进制基数树中的每个节点与和所述节点相关联的关键字所关联的值的地址相关联。

19.如权利要求11至13中任何一项所述的系统,还包括:

长整型存储数组,所述长整型存储数组保持用于与所述多个二进制关键字相关联的值的多个地址;以及

其中所述二进制基数树中的每个节点与所述长整型存储数组中的槽相关联,从而与和所述节点相关联的关键字所关联的值的地址相关联。

20.一种非暂态计算机可读介质,所述非暂态计算机可读介质包括存储在其上的用于支持分布式计算环境中的关键字管理的指令,所述指令当被执行时,使得所述分布式计算环境中的节点执行步骤,所述步骤包括:

提供包括内置多个二进制关键字的多个节点的二进制基数树,其中每个二进制关键字用于检索对应的对象、值或数据;

将所述二进制基数树序列化到缓冲区;

提供到所述缓冲区的接口,其中所述接口被配置为提供多个引用,其中所述多个引用中的每个引用与所述二进制基数树的节点以及所述多个二进制关键字中的二进制关键字相关联;以及

向多个引用者提供所述多个引用,所述多个引用者通过维护对所述二进制基数树中的节点的引用来使用所述二进制基数树检索与二进制关键字对应的值,由此使得所述多个引用者能够从所述多个引用实现所述多个二进制关键字并且定位与所述多个二进制关键字对应的值,而无需维护所述多个二进制关键字的拷贝。

21.一种非暂态计算机可读介质,所述非暂态计算机可读介质包括存储在其上的用于支持分布式计算环境中的关键字管理的指令,所述指令当被执行时,使得所述分布式计算环境中的节点执行如权利要求2至10中任何一项所述的方法。

22.一种包括用于执行如权利要求1至10中任何一项所述的方法的部件的装置。

用于支持分布式计算环境中的零拷贝二进制基数树的系统和方法

[0001] 版权声明：

[0002] 本专利文档的公开内容的一部分包含受版权保护的素材。版权拥有者不反对任何人对专利文档或专利公开内容按照它在专利商标局的专利文件或记录中出现的那样进行传真复制，但是除此之外在任何情况下都保留所有版权。

技术领域

[0003] 本发明一般涉及计算机系统，并且具体涉及分布式计算环境。

背景技术

[0004] 分布式数据网格是其中计算机服务器的集合在一个或多个集群中一起工作以管理分布式环境或集群环境内的信息和相关操作(诸如计算)的系统。分布式数据网格可以被用于管理跨服务器共享的应用对象和数据。分布式数据网格提供低响应时间、高吞吐量、可预测的可伸缩性、连续可用性和信息可靠性。作为这些能力的结果，分布式数据网格非常适合于在计算密集型的、有状态的中间层应用中使用。在特定示例中，分布式数据网格(诸如，Oracle®Coherence数据网格)将信息存储在存储器中以实现较高的性能，并且在保持该信息的拷贝跨多个服务器同步时采用冗余，从而确保在服务器故障的情况下的系统的弹性以及数据的持续可用性。

发明内容

[0005] 在实施例中，本公开描述了可以支持分布式计算环境中的关键字管理的系统和方法。系统可以使用二进制基数树来内置(intern)二进制关键字，其中该二进制关键字由分布式计算环境中的多个引用者引用。此外，系统可以提供对二进制基数树中的与分布式计算环境中的多个引用者所引用的二进制关键字相关联的节点的引用。然后，每个所述引用者可以使用对二进制基数树中的节点的引用来实现二进制关键字，而无需在本地维护二进制关键字的拷贝。

[0006] 本文还描述了可以支持分布式计算环境中的引用存储库的系统和方法。系统可以将票据(ticket)与引用存储库相关联，其中该引用存储库包含多个引用。此外，系统可以使用票据将引用存储库暴露给分布式计算环境中的一个或多个消费者。引用存储库可以响应于需要存储的引用的数量而扩充或缩小。附加地，系统可以向所述一个或多个消费者发信号通知(signal)关于引用存储库发生的一个或多个变化。

[0007] 当参照附图阅读时，从各种实施例的以下描述中，本发明的这些目的和优点以及其它目的和优点对于本领域技术人员来说将变得显而易见。

附图说明

[0008] 图1示出了根据本发明的实施例的分布式数据网格。

[0009] 图2示出了根据本发明的实施例的分布式计算环境中的零拷贝二进制基数树的使用。

[0010] 图3A示出了根据本发明的实施例的分布式计算环境中的零拷贝二进制基数树。

[0011] 图3B示出了根据本发明的实施例的分布式计算环境中的零拷贝二进制基数树方法。

[0012] 图4示出了根据本发明的实施例的分布式计算环境中的引用存储库。

[0013] 图5示出了根据本发明的实施例的分布式计算环境中的引用存储库。

[0014] 图6示出了根据本发明的实施例的分布式计算环境中的引用存储库方法。

具体实施方式

[0015] 本文描述的是可以支持分布式计算环境中的关键字管理的系统和方法。系统可以使用二进制基数树来内置二进制关键字,其中该二进制关键字由分布式计算环境中的多个引用者引用。此外,系统可以提供对二进制基数树中的与分布式计算环境中的多个引用者所引用的二进制关键字相关联的节点的引用。然后,每个所述引用者可以使用对二进制基数树中的节点的引用来实现二进制关键字,而无需在本地维持二进制关键字的拷贝。本文还描述了用于不同大小的引用集合的存储器高效的存储的引用存储库。如本文所描述的用于支持二进制基数树以内置二进制关键字以及支持引用存储库的系统和方法在下文参考图1描述的分布式数据网格中具有特定的效用。如本文描述的用于支持二进制基数树以内置二进制关键字以及支持引用存储库的系统和方法还可以被应用于各种各样的替代分布式计算环境和应用中。

[0016] 在以下描述中,将在附图的图中通过示例的方式而非限制的方式示出本发明。对本公开中的各种实施例的引用不一定是对同一实施例的引用,并且这样的引用意味着至少一个。虽然讨论了具体实现,但是要理解的是,这仅是为了说明性的目的而提供的。相关领域的技术人员将认识到,在不脱离本发明的范围和精神的情况下,可以使用其它组件和配置。

[0017] 此外,在某些实例中,将阐述许多具体细节以提供对本发明的详尽描述。然而,对于本领域技术人员将显而易见,可以在没有这些具体细节的情况下实践本发明。在其它实例中,公知的特征没有被详细地描述,以免使本发明模糊。

[0018] 借助于示出特定功能的性能及其关系的功能构建块来描述本发明。为了方便描述,本文通常任意地定义这些功能构建块的界限。因此,被示为由相同元件执行的功能在替代实施例中可以由不同元件执行。并且被示为在分开的元件中执行的功能可以替代地被组合到一个元件中。只要特定功能及其关系被适当地执行,就可以定义替代界限。因此,任何这样的替代界限在本发明的范围和精神之内。

[0019] 贯穿附图和详细描述,公共的附图标记被用来指示相似的元素;因此,如果元素在其它地方被描述,则附图中使用的附图标记可以或可以不在特定于该图的详细描述中被引用。三位数字附图标记中的第一个数字指示该元素首次出现的图的系列。

[0020] 分布式数据网格

[0021] 在以下描述中,描述了具有分区的高速缓存的**Oracle®**Coherence数据网格。然而,本领域的普通技术人员将理解的是,在不脱离本发明的范围的情况下,例如在以上发明

内容中描述的本发明可以被应用于本领域已知的任何分布式数据网格。此外，虽然描述了Oracle®Coherence分布式数据网格的许多具体细节以提供对本发明的详尽描述，但是对于本领域技术人员来说将显而易见，可以在没有这些具体细节的分布式数据网格中实践本发明。因此，在一些实施例中，实施本发明的分布式数据网格的特定实施方式可以不包括某些特征和/或包括与下文描述的分布式数据网格不同的或经修改的特征，而不脱离本发明的范围。

[0022] 图1示出了存储数据并且向客户端150提供数据访问的分布式数据网格100的示例。“数据网格集群”或“分布式数据网格”是包括多个计算机服务器(例如，120a、120b、120c和120d)的系统，这些计算机服务器在一个或多个集群(例如，100a、100b、100c)中一起工作以存储和管理分布式环境或集群环境内的信息以及诸如计算之类的相关操作。虽然分布式数据网格100被示为包括四个服务器120a、120b、120c、120d，其中在集群100a中有五个数据节点130a、130b、130c、130d和130e，但是分布式数据网格100可以包括任何数量的集群以及每个集群中的任何数量的服务器和/或节点。分布式数据网格可以将信息存储在存储器中以实现较高的性能，并且在保持该信息的拷贝跨多个服务器同步时采用冗余，从而确保在服务器故障的情况下的系统的弹性以及数据的持续可用性。在实施例中，分布式数据网格100实现例如在上面的发明内容和下面的详细描述中所描述的本发明。

[0023] 如图1所示，分布式数据网格通过在一起工作的若干服务器(例如，120a、120b、120c和120d)上分布数据来提供数据存储和管理能力。数据网格集群的每个服务器可以是常规的计算机系统，诸如，例如具有一个到两个处理器插座以及每处理器插座两个到四个CPU核的“商用x86”服务器硬件平台。每个服务器(例如，120a、120b、120c和120d)被配置为具有一个或多个CPU、网络接口卡(NIC)和存储器，该存储器包括例如最小4GB的RAM至多达64GB或更大的RAM。服务器120a被示为具有CPU 122a、存储器124a和NIC 126a(这些元件在其它服务器120b、120c、120d中也存在但没有示出)。可选地，每个服务器还可以被提供有闪存存储器(例如SSD 128a)以提供溢出(spillover)存储容量。当被提供时，SSD容量优选地是RAM的大小的十倍。使用高带宽NIC(例如，PCI-X或PCIe)将数据网格集群100a中的服务器(例如，120a、120b、120c、120d)连接到高性能网络交换机120(例如，千兆以太网交换机或更好的交换机)。

[0024] 集群100a优选地包含最少四个物理服务器以避免在故障期间数据丢失的可能性，但是典型的安装具有更多的服务器。每个集群中存在的服务器越多，故障转移和故障恢复越高效，并且服务器故障对集群的影响被减少。为了最小化服务器之间的通信时间，每个数据网格集群理想地被限制于在服务器之间提供单跳通信的单个交换机102。因此，集群可能受交换机102上的端口数量的限制。因此，典型的集群将包括4个到96个之间的物理服务器。

[0025] 在分布式数据网格100的大多数广域网(WAN)配置中，WAN中的每个数据中心具有独立但互连的数据网格集群(例如，100a、100b和100c)。WAN可以例如包括比图1中示出的更多的集群。此外，通过使用互连但独立的集群(例如，100a、100b、100c)和/或使互连但独立的集群位于彼此远离的数据中心中，分布式数据网格可以保护对于客户端150的数据和服务，以防止由自然灾害、火灾、洪水、长时间电力瘫痪等引起的一个集群中的所有服务器的同时丢失。贯穿企业并且跨地域维护的集群构成了企业数据的自动“备份存储”和高可用性服务。

[0026] 一个或多个节点(例如,130a、130b、130c、130d和130e)在集群100a的每个服务器(例如,120a、120b、120c、120d)上操作。在分布式数据网格中,节点可以是例如软件应用、虚拟机等,并且服务器可以包括节点在其上操作的操作系统、管理程序等(未示出)。在Oracle®Coherence数据网格中,每个节点是Java虚拟机(JVM)。取决于CPU处理能力和服务器上可用的存储器,可以在每个服务器上提供若干JVM/节点。JVM/节点可以按照分布式数据网格的需要被添加、启动、停止和删除。运行Oracle®Coherence的JVM在启动时自动加入集群。加入集群的JVM/节点被称为集群成员或集群节点。

[0027] 在Oracle®Coherence数据网格中,集群成员使用Tangosol集群管理协议(TCMP)进行通信。TCMP是被用于发现集群成员、管理集群、供应服务以及在集群成员之间传送数据的基于IP的协议。TCMP协议提供所有消息的完全可靠的、有序的递送。由于底层UDP/IP协议不提供可靠的或有序的递送,因此TCMP使用排队的、完全异步的、基于ACK和NACK的机制以用于可靠的消息递送,其中唯一整体标识(unique integral identity)用于保证与在服务器上操作的JVM相关联的队列中的消息的排序。不管集群大小如何,TCMP协议仅需要每个JVM/节点三个UDP/IP套接字(一个多播,两个单播)和六个线程。

[0028] 数据网格集群的功能基于由集群节点提供的服务。集群节点提供的每个服务具有特定的功能。每个集群节点就提供和消费集群服务而言可以参与若干集群服务(是若干集群服务的成员)。一些集群服务由集群中的所有节点提供,而其它服务由集群中的仅一个或仅一些节点提供。每个服务具有唯一地识别数据网格集群内的服务的服务名称以及定义服务可以做什么的服务类型。可以存在由数据网格集群中的节点提供的每种服务类型(除了根集群服务之外)的多个命名实例。所有服务优选地提供没有任何数据丢失的故障转移和故障恢复。

[0029] 由集群节点提供的每个服务实例通常使用一个服务线程来提供服务的具体功能。例如,由节点提供的分布式高速缓存服务由节点的单个服务线程提供。当在JVM/节点中解析分布式高速缓存的模式定义时,利用模式中指定的名称来实例化服务线程。该服务线程管理使用模式定义创建的高速缓存中的数据。一些服务可选地支持可以被配置为向服务线程提供附加处理资源的工作者(worker)线程的线程池。服务线程与线程池中的工作者线程合作以提供服务的具体功能。

[0030] 在Oracle®Coherence数据网格中,集群服务(例如,136a、136b、136c、136d、136e)跟踪集群中的成员资格和服务。每个集群节点始终具有恰好一个该类型的服务在运行。集群服务被自动启动以使得集群节点能够加入集群。集群服务负责检测其它集群节点、检测集群节点的故障(死亡)、以及注册集群中的其它服务的可用性。代理服务(例如,138c)允许来自在集群外部运行的客户端的连接(例如,使用TCP的连接)。调用服务(例如,134d)允许应用代码调用代理以对集群中的任何节点、或节点的任何组、或跨整个集群执行操作。虽然被示为各自仅在一个节点上,但是调用服务和代理服务可以被配置在分布式数据网格的节点中的任何数量的节点(多达所有节点)上。

[0031] 在Oracle®Coherence数据网格中,分布式高速缓存服务(例如,132a、132b、132c、132d、132e)是提供分布式数据网格中的数据存储并且在读/写/存储高速缓存数据的集群的所有节点上可操作的服务,即使该节点被禁用存储。分布式高速缓存服务允许集群节点跨集群100a分布数据(将数据分区),以使得高速缓存中的每条数据仅由一个集群节点

主要管理(保持)。分布式高速缓存服务处理诸如put(放置)、get(获取)等之类的存储操作请求。分布式高速缓存服务管理在分布式模式定义中定义的并且在集群的节点之间分区的分布式高速缓存(例如,140a、140b、140c、140d、140e)。

[0032] 分区是在分布式数据网格中并且被存储在分布式高速缓存(例如,140a、140b、140c、140d和140e)中的受管理数据的基本单元。数据在逻辑上被划分成跨多个集群节点分布的主分区(例如,142a、142b、142c、142d和142e),以使得集群中恰好一个节点负责高速缓存中的每条数据。每个高速缓存(例如,140a、140b、140c、140d和140e)可以保持若干分区。每个分区(例如,142a、142b、142c、142d、142e)可以保持一个数据(datum),或者它可以保持许多数据。当需要或期望时,可以将分区从一个节点的高速缓存迁移到另一节点的高速缓存。例如,当节点被添加到集群时,分区被迁移,以使得它们被分布在包括新添加的节点的可用节点中。在非复制分布式数据网格中,仅存在每个分区(主分区)的一个活动拷贝。然而,通常还存在用于故障转移的(被存储在不同的服务器上的)每个分区的一个或多个副本/备份拷贝。因为数据分散在被分布在集群的服务器中的分区中,因此管理数据和提供对数据的访问的责任被跨集群自动负载平衡。

[0033] 分布式高速缓存服务可以被配置为使得每条数据由一个或多个其它集群节点备份以支持没有任何数据丢失的故障转移。例如,如图1所示,每个分区被存储在主分区(例如,深色阴影正方形142a、142b、142c、142d和142e)和分区的一个或多个同步的备份拷贝(例如,浅色阴影正方形144a、144b、144c、144d和144e)中。每个分区的备份拷贝被存储在与和它同步的主分区分开的服务器/节点上。节点上的分布式高速缓存服务的故障转移涉及将分区的备份拷贝提升为主分区。当服务器/节点发生故障时,所有剩余的集群节点确定它们为故障节点上的主分区保持了哪些备份分区。然后,集群节点将备份分区提升为它们被保持的任何集群节点上的主分区(然后新备份分区被创建)。

[0034] 分布式高速缓存是数据对象的集合。每个数据对象/数据可以是例如数据库表的行的等价物。每个数据与识别该数据的唯一关键字相关联。每个分区(例如,142a、142b、142c、142d、142e)可以保持一个数据,或者它可以保持许多数据,并且分区被分布在集群的所有节点中。在Oracle®Coherence数据网格中,每个关键字和每个数据被存储为以被称为可移植对象格式(Portable Object Format,POF)的高效未压缩二进制编码被序列化的数据对象。

[0035] 为了找到特定的数据,每个节点具有将关键字映射到分区的映射,例如散列映射。该映射对集群中的所有节点是已知的,并且跨集群的所有节点被同步和更新。每个分区具有将与该分区相关联的每个关键字映射到存储在该分区中的对应数据的后备映射。与特定关键字/数据相关联的操作可以在分布式数据网格中的任何节点处从客户端接收。当节点接收到操作时,如果关键字与接收节点上的主分区相关联,则该节点可以提供对与该关键字相关联的值/对象的直接访问。如果关键字不与接收节点上的主分区相关联,则该节点可以(在一跳中)将操作直接引导到保持与该关键字相关联的主分区的节点。因此,通过使用散列映射和分区映射,每个节点可以提供对与分布式高速缓存中的每个关键字对应的每个数据的直接访问或一跳访问。

[0036] 在一些应用中,分布式高速缓存中的数据初始地从包括数据112的数据库110填充。数据库110中的数据112是序列化的、分区的并且被分布在分布式数据网格的节点中。分

布式数据网格100将从来自数据库110的数据112创建的数据对象存储在服务器120a、120b、120c、120d的存储器中的分区中，以使得客户端150和/或数据网格100中的应用可以直接从存储器访问这些数据对象。从分布式数据网格100中的数据对象读取和对分布式数据网格100中的数据对象写入比直接使用数据库110快得多，并且允许可以实现的更多的同时连接。数据的存储器中复制以及保证的数据一致性使得分布式数据网格适合于管理存储器中的事务，直到这些事务被持久存储到诸如数据库110之类的外部数据源以用于存档和报告。如果对存储器中的数据对象做出改变，则这些改变在主分区和备份分区之间进行同步，并且可以通过使用异步写入(后写入)随后被写回数据库110以避免瓶颈。

[0037] 虽然数据跨集群节点分散，但是客户端150可以连接到任何集群节点并且检索任何数据。这被称为位置透明性，它意味着开发人员不必基于高速缓存的拓扑来编码。在一些实施例中，客户端可以连接到特定服务，例如，特定节点上的代理服务。在其它实施例中，连接池或负载平衡器可以被用于将客户端引导到特定节点以及确保客户端连接被分布在一些数据节点或所有数据节点上。无论怎样被连接，分布式数据网格中的接收节点接收来自客户端150的任务，并且每个任务与特定数据相关联，因此每个任务必须由特定节点处理。接收到对于特定数据的任务(例如，针对高速缓存服务的调用)的任何节点识别其中存储该数据的分区和负责该分区的节点，然后接收节点例如通过进行远程高速缓存调用来将任务引导到保持所请求的分区的节点。由于每条数据仅由一个集群节点管理，因此通过网络的访问仅是“单跳”操作。这种类型的访问是极其可伸缩的，因为它可以使用点对点通信并且因此最佳地利用了诸如InfiniBand(无限带宽)之类的交换结构网络。

[0038] 类似地，高速缓存更新操作可以使用相同的单跳点对点方法，其中数据被发送到具有主分区的节点和具有该分区的备份拷贝的节点二者。直到所有备份都已确认接收时，对高速缓存的修改才被认为是完整的，这保证数据的一致性被维护，并且保证如果集群节点在写入操作期间意外发生故障，则不会丢失数据。分布式高速缓存服务还允许将某些集群节点配置为存储数据，而其它集群节点被配置为不存储数据。

[0039] 在一些实施例中，分布式数据网格可选地被配置为具有弹性数据特征，该弹性数据特征利用固态设备(例如SSD 128a)(最典型的是闪存驱动器)为高速缓存提供溢出能力。通过使用弹性数据特征，高速缓存被指定使用基于RAM或DISK日志的后备映射。日志提供用于存储对象状态变化的机制。关于具体关键字记录每个数据/值，并且存储器中的树被用于存储指向该数据的指针(可以直接存储在树中的微小数据/值)。这允许一些值(数据)被存储在固态设备(例如SSD 128a)中，同时将索引/存储器树存储在存储器(例如RAM 124a)中。弹性数据特征允许分布式数据网格支持每节点更大量的数据并且与完全基于RAM的解决方案相比几乎没有性能损失。

[0040] 诸如上文描述的Oracle®Coherence数据网格之类的分布式数据网格可以通过解决数据操作延迟问题以及通过实时高速缓存和处理数据来提高系统性能。应用在数据网格中高速缓存数据，从而避免对后端数据源的昂贵请求。共享数据高速缓存提供高速缓存的数据的单一的、一致的视图。从高速缓存读取比查询后端数据源更快并且随着应用层自然地伸缩。存储器中性能缓解了瓶颈并且减少了数据竞争，从而提高应用的响应性。并行查询和计算被支持，以提高基于数据的计算的性能。分布式数据网格是容错的，从而提供数据可靠性、准确性、一致性、高可用性和灾难恢复。分布式数据网格使得应用能够线性和动态

伸缩，以实现可预测的成本和改进的资源利用率。对于许多应用，分布式数据网格提供有价值的共享数据源解决方案。

[0041] 在实施例中，分布式数据网格100实现例如如下文描述和图2-6所示的用于关键字管理的一个或多个零拷贝二进制基数树和/或引用存储库。在特定实施例中，可以相对于基于RAM或DISK日志的后备映射来实现用于关键字管理的零拷贝二进制基数树。在实施例中，基于RAM或DISK日志的后备映射与利用固态设备(例如SSD 128a)(最典型的是闪存驱动器)的分布式数据网格的弹性数据特征结合利用，以便为高速缓存提供溢出能力。通过利用零拷贝二进制基数树，分布式数据网格减少或消除了与在服务器的RAM中维护关键字的多个拷贝相关联的存储器浪费，从而提高了分布式数据网格的性能。此外，分布式数据网格100可以实现引用存储库，以用于以存储器高效的方式存储对零拷贝二进制基数树中内置的关键字的引用。

[0042] 分布式数据网格中的关键字管理

[0043] 本文描述的是可以使用包括例如图1的分布式数据网格的分布式计算环境中的零拷贝二进制基数树来支持关键字管理的系统和方法。本文还描述了用于以存储器高效的方式存储对零拷贝二进制基数树中内置的关键字的引用的引用存储库。

[0044] 二进制基数树是某一类型的字典树(trie)。字典树是被用于存储其中关键字通常是串的动态集合或关联数组的有序的数据结构。与二进制搜索树不同，字典树中没有节点存储与该节点相关联的整个关键字；作为替代，节点在树中的位置定义与该节点相关联的关键字。节点的所有后代具有与该节点相关联的串的共同前缀，并且字典树的根节点与空串相关联。字典树可以被用于检索与关键字相关联的值。值不一定与每个节点相关联。字典树内部的一些节点仅仅引用附加节点。相反，值往往仅与叶子节点以及对应于感兴趣的关键字的一些内部节点相关联。

[0045] 基数树是表示空间优化的字典树的数据结构，其中作为父节点的唯一子节点的每个节点与它的父节点合并。有利的是，与常规的树不同，二进制基数树不需要从全部关键字的开始直到不相等的点来比较全部关键字整体。在二进制基数字典树中，当遍历字典树时，比较关键字的节点的部分被用于选择子节点中的一个子节点(例如，下一位＝1时选择左子节点，下一位＝0时选择右子节点)。特定内部节点的所有子节点共享相同的前缀直到不相等的点，即，它们在该特定内部节点处分开的位置。二进制基数树提供了空间高效的方式来存储二进制关键字集合，并且还实现用于遍历树以检索与给定二进制关键字相关联的值的高效机制。

[0046] 图2示出了使用零拷贝二进制基数树支持分布式数据网格中的弹性数据结构的图示。如图1所示，节点130a可以包括随机存取存储器(RAM 124a)和诸如固态盘(SSD 128a)之类的各种基于盘的设备。数据网格集群100的节点130a可以利用弹性数据结构，该弹性数据结构使得数据网格集群能够以接近存储器的速度在SSD 128a中存储数据和/或从SSD 128a读取数据(参见图1)。如图2所示，节点130a使用RAM日志210以用于将数据存储在存储器中，并且使用闪存日志220以用于将数据存储到基于闪存的设备。日志210、220记录与数据存储装置中的值/对象的一系列修改相关联的状态变化。此外，RAM日志210与闪存日志220一起工作，以实现从RAM存储装置到闪存盘存储装置的无缝数据溢出。可以以不同的方式使用RAM日志210和闪存日志220。例如，RAM日志210和闪存日志220可以被用于支持后备映射、支

持备份存储和/或支持节点130a中的复合高速缓存(例如,近高速缓存)。

[0047] 节点130a使用的日志210-220中的每一个可以包含多个条目。例如,RAM日志210可以包括RAM日志条目212-214,而闪存日志220可以包括闪存日志条目222-224。系统将被存储在每个不同的日志条目212、214、222、224中的每个值/对象与唯一关键字相关联。当对弹性数据存储装置应用改变时,每个关键字保持与包含值/对象的当前版本的日志条目相关联。因此,用户可以使用被存储在堆202上的相关联的关键字来找到被存储在弹性数据结构200中的日志条目中的值。

[0048] 为了找到这些值,每个日志条目与分开地存储在堆202上的日志票据相关联。日志票据识别条目在RAM日志210或闪存日志220中的位置。日志票据可以表示用于在堆外(诸如在对应于关键字的日志的条目中)存储的值的虚拟地址。附加地,日志票据可以包括与在堆外存储的值相关联的附加属性(诸如驱逐属性和到期属性)。日志票据使用长整型(long)原始数据类型。在Java中,长整型数据类型是64位有符号二进制补码整数。长整型存储数组230可以被提供以用于存储与条目中的每个条目相关联的日志票据。如图2所示,长整型存储数组230包括多个编号的槽/行232,每个编号的槽/行包含多个日志票据234中的一个,并且可选地还包括与日志条目234相关联的多个属性236。长整型存储数组230允许使用槽编号来检索槽/行中的日志票据。日志票据允许对应的条目从它位于的任何日志(RAM日志或闪存日志)中被检索。

[0049] 然而,请求者需要找到与特定关键字相关联的条目。实现这一点的简单方式将是向长整型存储数组230添加包括与每个条目相关联的关键字的列。然而,这种解决方案不是期望的,因为它需要在长整型存储数组中包括每个关键字的拷贝,并且因为搜索匹配的关键字将是昂贵的操作。作为替代,如图2所示,零拷贝二进制基数树240被提供,以用于以压缩的方式存储与日志条目相关联的关键字,并且允许对与对应于关键字的日志条目对应的日志票据的高效检索。零拷贝二进制基数树240将二进制关键字与长整型存储数组230的槽232相关联,以使得二进制关键字可以被用于找到日志210、220中的相关联的值。

[0050] 如图2所示,服务器节点130a使用零拷贝二进制基数树240以用于在节点130a的堆202上存储不同的关键字241-245。例如,存储在零拷贝二进制基数树240中的关键字241-245包括“ALBERT”241、“ALFRED”242、“ANDREW”243、“ANTONIO”244和“ANTONY”245。如图2所示,由于存储在零拷贝二进制基数树240中的所有关键字241-245以字符“A”开始,因此零拷贝二进制基数树240的根节点246存储字符“A”。此外,以字符“AL”开始的两个关键字“ALBERT”331和“ALFRED”332二者共享同一内部节点247。此外,关键字“ALBERT”331使用包含字符“BERT”的节点241,而关键字“ALFRED”使用包含字符“FRED”的节点242。类似地,关键字“ANDREW”333使用节点246、248和243,关键字“ANTONIO”使用节点246、248、249和244,关键字“ANTONY”使用节点246、248、249和245。注意到虽然为了便于解释示出了字母关键字,但是基数树将通常是二进制基数树,关键字将通常是二进制的,并且节点将表示二进制的特定位或位块。

[0051] 表示存储在零拷贝二进制基数树240中的压缩关键字的每个节点241-245还与长整型存储数组230中的编号的槽232中的一个编号的槽相关联,从而将每个特定压缩关键字与数组中的如下特定槽232相关联:该特定槽232包含特定日志票据234并且与闪存日志220或RAM日志210中的特定值/对象相关联。因此,零拷贝二进制基数树240充当将二进制关键

字与(以长整型格式的)日志票据相关联的二进制长整型映射,其中该日志票据识别包含与该关键字相关联的值/对象的日志条目。附加地,为了提高效率,可以将微小值(例如,具有多达七个字节的大小的值)直接存储在零拷贝二进制基数树240中,而不是堆外存储在日志210、220中。

[0052] 如图2所示,用于存储关键字“ALBERT”的节点241包括对长整型存储数组230的存储日志票据“0x112ca530”的槽1的引用。用于存储关键字“ALFRED”的节点242包括对长整型存储数组230的存储日志票据“0x38fce788”的槽2的引用。用于存储关键字“ANDREW”的节点243包括对长整型存储数组230的存储日志票据“0xfd738d80”的槽3的引用。用于存储关键字“ANTONIO”的节点244包括对长整型存储数组230的存储日志票据“0x3819e6c0”的槽4的引用。最后,用于存储关键字“ANTONY”的节点245包括对长整型存储数组230的存储日志票据“0xb381efd0”的槽5的引用。

[0053] 因此,如图2所示,当获得对与关键字“ALBERT”相关联的值的请求时,关键字“ALBERT”被应用于基数树240,基数树240将节点241识别为与关键字“ALBERT”对应。节点241包含对长整型存储数组230的槽1的引用,因此从长整型存储数组230的槽1使用日志票据“0x112ca530”。日志票据“0x112ca530”票据表示用于堆外存储在日志的条目224中、对应于关键字“ALBERT”的值225的虚拟地址。因此,系统使得能够响应于包含与该值相关联的关键字“ALBERT”的请求而从日志中检索值225。此外,通过利用这样的紧凑数据结构,系统可以存储用于非常大的数据集的元数据信息。

[0054] 例如,在于2014年7月2日提交的标题为“SYSTEM AND METHOD FOR SUPPORTING ELASTIC DATA METADATA COMPRESSION IN A DISTRIBUTED DATA GRID”的美国专利申请序列号No.14/322,576、于2014年7月2日提交的标题为“SYSTEM AND METHOD FOR SUPPORTING MEMORY ALLOCATION CONTROL WITH PUSH-BACK IN A DISTRIBUTED DATA GRID”的美国专利申请序列号No.14/298,458、以及于2013年12月31日公布的标题为“ELASTIC DATA TECHNIQUES FOR MANAGING CACHE STORAGE USING RAM AND FLASH-BASED MEMORY”的美国专利No.8,621,143中进一步描述了分布式数据网格中的二进制基数树的使用,这些申请和专利通过引用被并入本文。如本文所描述的,数据的关键字202、203、204被压缩到高效的二进制基数树数据结构中,该二进制基数树数据结构充当用于对存储在RAM和闪存盘中的存储器块中的值/对象的索引。

[0055] 在图1的分布式数据网格中,许多数据结构视关键字而定。因此,图2中所示的零拷贝二进制基数树240提供了用于(直接从树或者使用日志票据)对与关键字相关联的值/对象执行操作的手段。二进制基数树是空间高效的,尤其是因为树中的节点不存储整个关键字,而是仅存储选择子节点所需的关键字的块。然而,分布式数据网格还可以被配置为提供实现和/或便于基于值/对象的参数找到对象的若干索引和反向索引。例如,Oracle® Coherence数据网格为了高效的查询性能而对数据加索引。为了允许方便地执行查询,创建允许查询识别与查询值匹配的关键字的反向索引(IndexedValue->Set<Key>)。在存在大量索引值的情况下,这导致需要维护大量的关键字集合Set<Key>。

[0056] 在现有系统中,对值/对象的引用需要每个引用者维护用于该集合的每个成员的整个二进制关键字(例如“ANTONY”)的拷贝。因此,每个引用者必须维护与值对应的关键字的字节拷贝,以便使用二进制基数树来检索该值。例如,索引/引用者将必须包括用于集合

中的每个值/对象的关键字的字节拷贝。即，反向索引确实是用于满足索引值的对象的关键字的字节拷贝集合。此外，在多个引用者引用同一关键字的情况下，这些引用者中的每个引用者必须维护完整的二进制关键字的拷贝，从而导致每个完整的二进制关键字的多个拷贝。因为这些引用者/索引被存储在堆上(on-heap)，因此维护许多二进制关键字的多个拷贝导致与分布式数据网格中的关键字管理相关联的显著的存储器成本。

[0057] 零拷贝特征

[0058] 因此，期望减少由分布式数据网格中的引用者(例如索引)维护许多完整的二进制关键字的多个拷贝带来的存储器成本，以便提高分布式数据网格或其它分布式计算环境的性能。相应地，本公开描述了用于支持零拷贝二进制基数树的系统和方法，该零拷贝二进制基数树减少或消除了由在分布式数据网格中维护许多二进制关键字的多个拷贝带来的存储器成本。根据本发明的实施例，引用者依零拷贝二进制基数树240中的节点的位置来维护对零拷贝二进制基数树中的节点的引用，而不是维护完整的二进制关键字的字节拷贝。对零拷贝二进制基数树中的节点的位置的引用需要比完整的二进制关键字的拷贝少得多的存储器，从而减少或消除了由维护许多二进制关键字的多个拷贝带来的存储器成本，并且提高了分布式数据网格或其它分布式计算环境的性能。

[0059] 图2示出了根据本发明的实施例的在分布式计算环境中支持二进制基数树中的零拷贝特征。如图2所示，分布式计算环境中的多个引用者271-273可以引用存储在零拷贝二进制基数树240中的同一关键字，而无需在本地维护关键字的拷贝(即，零拷贝类型的特征)。例如，引用者271-273可以经由叶子节点241引用同一关键字“ALBERT”。此外，零拷贝二进制基数树240容许并发修改，并且可以是不可变的。因此，零拷贝二进制基数树240提供了非常高效的机制以访问适当二进制关键字中的字节，并且允许对零拷贝二进制基数树240的节点的安全外部引用。

[0060] 没有零拷贝特征的话，每个引用者将需要维护引用者所引用的关键字的拷贝。例如，不同引用者271-273中的每个引用者将需要在堆202上存储关键字“ALBERT”的分开的拷贝281-283(即，每引用者一个拷贝)，以便维护对存储在零拷贝二进制基数树240中的二进制关键字“ALBERT”的外部引用。通过使用零拷贝特征，不同引用者271-273中的每个引用者不必存储关键字“ALBERT”的分开的拷贝281-283。因此，零拷贝特征可以显著地减少与分布式计算环境200中的关键字管理相关联的堆上存储器成本。

[0061] 如图2所示，提供了将二进制关键字内置到二进制基数树210中的设施。二进制基数树210提供了允许共享字节被去重复(de-duplicate)的高度紧凑的数据结构。当高速缓存被启动以存储与高速缓存相关联的关键字时，二进制基数树被启动。对字节数组的引用通常需要这些字节的拷贝。然而，通过将关键字直接序列化到字节缓冲区(目的地:网络，盘)，零拷贝二进制基数树210中的二进制数据(binary)成为可用的，从而创建二进制数据的“视图”。二进制数据的“视图”将可导航字节数组暴露给消费者，并且导航存储在零拷贝二进制基数树中的二进制数据。二进制数据的“视图”中的字节可以由引用者访问，而无需拷贝底层字节。因此，零拷贝二进制基数树系统允许零拷贝引用，从而实现密度和引用这些字节的能力二者。

[0062] 然后，所有引用者211-213可以引用作为零拷贝二进制基数树240中的节点(例如，图2的节点241)的ReadBuffer中的位置。引用本质上是暴露可导航字节数组的接口。被称为

ByteSequence的接口提供诸如byteAt(n)和长度(length())之类的方法,因此该接口可以在for循环结构体中被使用,以有效地引用字节数组中的关键字。ByteSequence接口表示可导航字节数组中的字节序列。方法byteAt(n)确定字节序列的第n个字节。方法长度可以确定字节序列的节点中表示的数据的字节的数量。因此,诸如反向索引之类的引用者可以被实现为Set<ByteSequence>,Set<ByteSequence>提供允许对字节访问的到二进制基数树中的'view'/ByteSequence。与将反向索引引用者实现为要求集合中的每个二进制数据被利用二进制基数树中所有字节的拷贝来完全实现的Set<Binary>相比,这是有利的。因此,零拷贝二进制基数树系统允许零拷贝引用(即,没有二进制基数树的拷贝),从而实现密度和引用二进制基数树的字节的能力二者。零拷贝二进制基数树240中的位置表示被引用的关键字。这导致关键字的零拷贝,并且因此导致更少的存储器浪费。即,不需要引用者211-213来维护关键字"ALBERT"的拷贝281、282和283。

[0063] 零拷贝二进制基数树是流动的数据结构,因为关键字可以与关键字检索并发地被添加或删除。当关键字被添加时,新的叶子节点被添加并且旧的叶子节点被分割。零拷贝二进制基数树考虑了导致通过例如引用收集进程将树和孤立(orphaned)引用分割的树修改,在该引用收集进程中引用收集器线程收集并且校正对零拷贝二进制基数树240的孤立引用。引用者被发信号通知修改过时的引用。引用(ByteSequence)本身可以是自更新的。这意味着到BRT中的'view'/ByteSequence被发信号通知,以理解它不再有效并且有效的引用应当被导出。有效引用将替换无效引用。孤立节点的解析是提交到服务的守护程序(daemon)池的任务,因此不需要专用线程。收集器线程知道/被通知了已经被孤立的所有节点,并且反向索引集合可以基于身份被检查以导出孤立节点。

[0064] 此外,在节点被孤立的情况下,它仍然维持对它的父节点的单向引用,以使得它可以充当到树的入口点。孤立节点在功能上是正确的。可以通过检查它的父节点的引用来将孤立节点确定为孤立节点。如果所指示的父节点不再引用节点,则该节点是孤立节点。节点由于树中的分割而变为孤立的,因此父节点将仍然引用孤立节点中的字节中的一些字节。相应地,可以从父节点开始并且沿着树向下行走到新节点/有效节点来确定新节点/有效节点。

[0065] 虽然孤立节点在功能上是正确的并且可以被用于找到新节点/有效节点,但是对孤立节点解引用仍然是重要的。这是因为孤立节点是在例如分割之后替代它的节点的副本。孤立节点直到它被解引用才能被垃圾收集,因此孤立节点的存在降低了二进制基数树的有效存储器密度。相应地,发信号通知、收集和解引用被执行,以允许孤立节点被垃圾收集,这增加了二进制基数树的有效存储器密度并且降低了总体存储器使用。

[0066] 图3A示出了包括根节点302和多个叶子节点(306a-306l)的零拷贝二进制基数树300的实施例(为了说明的目的,示出了有限数量的节点,但是将存在大得多的节点集合)。虽然对二进制基数树的引用通过实现零拷贝特征的接口进行,但是二进制基数树本身以基本上常规的方式实现。如图3A所示,每个节点可以具有多个子节点,其中每个子节点在可以被二进制搜索(binary search)以找到给定第一字节的正确节点的数组中(多达255个子节点,因为每个子节点表示唯一的字节)。二进制基数树包括紧凑节点和字节数组节点。紧凑节点是可以在八个字节(长整型)中存储它的信息中的所有信息的节点。在信息比在8字节长整型中可以保持的更大的情况下,使用字节数组节点。叶子节点表示存储在二进制基数

树中的字节数组的尾部。例如，在二进制基数树中存储0xFFF1、0xFFF2和0xFFF1FF将导致加粗部分是叶子节点。此外，F1将既是叶子节点又是第三个关键字0xFFF1FF的父节点。二进制基数树中的节点位置基于存储的字节数组中的共享字节以及存储的关键字的数量。

[0067] 然而，随着二进制基数树的深度增加，必须被遍历以实现单个关键字的链接节点的数量增加，并且因此行走(walk)二进制基数树以实现关键字的成本也增加。提供了高效的行走机制来回答诸如返回树中的第n个位置处的字节的byteAt(n)之类的问题。高效的行走机制减少了必须被遍历以确定树中第n个位置处的字节的节点的数量。与常规的二进制基数树不同，本文所公开的二进制基数树被利用“卫星”节点来增强，这缩短了二进制基数树必须被行走以回答诸如byteAt(n)之类的问题的最大距离。“卫星”节点304将识别它们在树中的位置的信息编码，并且因此它们便于回答对于完整的二进制关键字的诸如byteAt(n)之类的问题，而无需行走整个树。因此，使用零拷贝二进制基数树240来检索值对于内置关键字来说是非常高效的。

[0068] 卫星节点304是“特殊的”节点，因为它们知道它们相对于根节点的位置。卫星节点位置信息被存储在长整型(紧凑节点)或字节数组中，以使得它可以在二进制基数树的遍历期间被使用。可以通过基于使用模型的自适应算法来确定卫星节点在二进制基数中的位置。自适应算法在存储位置信息所需的附加存储器和树遍历开销的减少之间进行折衷。例如，卫星节点可以被自适应地放置得更靠近被频繁行走的节点，以便减少对于被频繁访问的关键字必须遍历的节点的数量。可替代地，卫星节点被提供在从二进制基数树的叶子节点到根节点的距离的1/8处。卫星节点“知道”它们离根节点302有多远，因此，当行走树以回答byteAt(n)时，在朝正确的叶子节点遍历之前，遍历仅需要向上走到远至卫星节点。例如，考虑存储在BRT中的100个字节，并且函数byteAt(91)被调用;为了确定在位置91处的字节，必须行走到树的顶部以找到0位置并且向下行走到位置91。然而，在字节50处的卫星节点304将允许行走将导航限制为向上50字节和向下41字节，而不是向上100字节和向下91字节。

[0069] 除了卫星节点的存在之外，零拷贝二进制基数树和常规二进制基数树之间的主要差别是如何从二进制基数树返回关键字。在现有实现中，当引用者请求来自二进制基数树的关键字时，关键字的字节数组拷贝被返回。相比之下，在本文公开的零拷贝二进制基数树中，零拷贝二进制基数树替代地返回对字节数组中的节点的位置的引用。因此，例如反向索引330包含对字节数组310中的位置的一组引用(例如，336a、336d、336i、336j)，而不是二进制关键字的拷贝。这减少了用于索引的存储器开销。

[0070] 如上文所描述的，通过将关键字直接序列化为字节数组310，零拷贝二进制基数树210中的二进制数据成为可用的，从而创建二进制数据的“视图”。二进制数据的“视图”将叶子节点306a-3061的可导航字节数组310暴露给消费者。叶子节点306a-3061本身被暴露为名为ByteSequence的接口320的实现者。该ByteSequence接口320允许引用被存储在二进制基数树300中的字节数组310，而不拷贝实现字节数组的二进制基数树中的字节。二进制基数树中的节点从不直接暴露于二进制基数树之外，相反，ByteSequence接口320允许使用对字节数组中的特定位置处的节点的引用和字节数组的遍历来导航字节数组。

[0071] 因此，二进制数据的“视图”中的字节可以由引用者(例如反向索引330)访问，而无需拷贝底层字节。因此，对叶子节点的引用允许消费者导航二进制数据而无需创建二进制

关键字的拷贝。此外，引用的使用使得能够提供诸如优化的hashCode(散列码)之类的功能和相等的实现，并且因此允许在诸如反向关键字索引之类的基于散列的容器中的高效使用。参见例如下文描述的引用存储库实现。每个索引需要至少引用所有关键字至少两次(正向索引和反向索引)。因此，使用引用而不是创建关键字的字节拷贝显著地减少了用于索引的存储器开销，除此之外还便于将可导航二进制数据直接流传输到通过套接字发送字节的诸如DataOutput之类的流。

[0072] 图3B示出了根据本发明的实施例的分布式计算环境中的零拷贝二进制基数树方法。如图3B所示，该方法包括在步骤340处提供二进制基数树。二进制基数树包括内置多个二进制关键字的多个节点。在步骤342处，二进制基数树被序列化到缓冲区以提供二进制数据的视图。在步骤344处，提供到缓冲区的接口。例如，接口是支持诸如byteAt(n)和长度之类的方法的字节序列缓冲区。在步骤346处，向多个引用者提供对缓冲区中的节点的多个引用。引用者可以包括多个反向索引。引用允许引用者实现二进制关键字并且定位与关键字相关联的值，而无需维护关键字的拷贝。使用引用来代替二进制关键字的字节拷贝减少了与反向索引相关联的存储器开销。在步骤348处，可以将节点添加到二进制基数树。在步骤350处，收集器线程收集并且校正对孤立节点的引用。

[0073] 在一些实施例中，诸如反向索引之类的引用者可以利用如下文描述的引用存储库来存储对缓冲区中的节点的引用。

[0074] 引用存储库

[0075] 在诸如图1的分布式数据网格100之类的分布式计算环境中使用索引可能需要保持大量的引用(例如，以用于将索引值映射到关键字引用的存储库，以便允许高效地执行查询)。例如，为了高效的查询性能，Oracle®Coherence数据网格通过创建反向索引来索引数据，该反向索引识别指向与索引值对应的对象的关键字集合(IndexedValue->Set<Key>)。可能存在大量的索引值，并且这导致需要维护大量的Set<Keys>。此外，这些集合中的一些集合将是大的(例如，对于常见值)，而这些集合中的一些集合将是小的(例如，对于稀有值)。鉴于可能需要维护的大量Set<Keys>，以存储器高效的方式来存储它们是所期望的。然而，现有系统利用Set<Key>的常规JDK实现。这种常规Set实现没有针对存储少量元素或大量元素进行优化，并且因此不是存储器高效的。因此，期望提供以针对存储少量元素和大量元素二者进行优化的存储器高效的方式将索引值映射到关键字引用的存储库的系统和方法。

[0076] 根据本发明的实施例，可以在分布式计算环境中使用引用存储库，以用于以存储器高效的方式在关键字值存储库中存储引用。引用存储库的使用允许高效地存储引用集合。此外，用于特定引用集合的引用存储库的大小可以随着数据集改变而扩充或缩小以适应存储器需求的变化。引用存储库的扩充或缩小由从存储库中移除引用或向存储库添加引用来触发。作为存储库的任何插入或移除的结果，API返回指向存储库的句柄/票据。这个返回的句柄可以与原始存储库相同或者在扩充或缩小的情况下不同。

[0077] 图4示出了根据本发明的实施例的支持分布式计算环境中的引用存储库402的图示。如图4所示，分布式计算环境400(例如，分布式数据网格)中的引用存储库实例401可以包含多个引用410(例如，与反向索引相关联的各种关键字或对由上文描述的零拷贝二进制基数树实现提供的可导航字节数组的引用)。

[0078] 如图4所示，引用存储库402包括多个引用存储库实例401。每个引用存储库实例401可以针对存储少量引用被优化，并且还可以针对存储大量引用而被自适应地优化。此外，每个引用存储库实例401与票据420(例如，长整型变量)相关联。系统使用票据420以将引用存储库实例401暴露给分布式计算环境400中的一个或多个消费者412。消费者是需要知道引用存储库实例的改变的任何行动者(actor)。例如，如果反向索引引用引用存储库实例(关键字的集合)；则索引本身可以是一旦引用存储库实例改变就向其发信号通知的消费者，并且该索引可以查找新的引用存储库实例。此外，引用存储库402针对引用存储库实例401已发生的任何改变(例如，引用存储库实例401的扩充或缩小)发信号通知所述一个或多个消费者412。

[0079] 每个引用存储库实例可以从针对存储少量(大量)引用被优化的存储库扩充(或缩小)为针对存储大量(少量)引用被优化的存储库。此外，引用存储库实例被管理，以使得它们可以在扩充/缩小事件之后被重新使用，从而避免重新分配引用存储库所需的存储器。这为引用提供了高效的存储器使用和访问时间。消费者(引用引用存储库的消费者)可以注册以在存储库发生改变时被发信号通知。然后，注册的消费者可以消费来自引用存储库的信号，即，关于扩充/缩小事件的信号。当反向索引的数量增加并且每索引值的关键字的数量增加时，引用存储库变得更加有益。

[0080] 因此，创建了引用存储库并且利用票据来引用该引用存储库。票据利用长整型(long)原始数据类型来表示。在Java中，长整型数据类型是64位有符号二进制补码整数。将引用存储库实例暴露为长整型允许引用者使用紧凑的高效数据结构(即LongLongMap)来引用存储库。在Java中，LongLongMap提供具有从关键字到单个值的唯一绑定的关联容器。在这种应用中，LongLongMap将表示引用存储库的关键字与引用存储库的内容相关联。当用于特定索引的引用存储库实例被扩充或缩小时，与特定索引相关联的票据可以改变。因此，当用于特定索引的引用存储库实例被扩充或缩小时，新票据被指派给引用存储库中的索引。

[0081] 图5示出了引用存储库500，该引用存储库500结合多个引用存储库实例520-528来使用LongLongMap 510。引用存储库实例520将用于与对应于满足索引值“RED”的对象的关键字对应的节点的引用集合530存储到零拷贝二进制基数树中。引用存储库实例522将用于与对应于满足索引值“BLUE”的对象的关键字对应的节点的引用集合532存储到零拷贝二进制基数树中。引用存储库实例524将用于与对应于满足索引值“WHITE”的对象的关键字对应的节点的引用集合534存储到零拷贝二进制基数树中。引用存储库实例526将用于与对应于满足索引值“BLACK”的对象的关键字对应的节点的引用集合536存储到零拷贝二进制基数树中。引用存储库实例528将用于与对应于满足索引值“PURPLE”的对象的关键字对应的节点的引用集合538存储到零拷贝二进制基数树中。

[0082] LongLongMap 510包括将多个索引值514与多个票据512相关联的多个行。票据表示用于与索引值相关联的引用存储库的虚拟地址。因此LongLongMap 510将索引值“RED”与票据“0x112ca533”相关联、将“BLUE”与票据“0x38fce786”相关联、将“WHITE”与票据“0xfd738d84”相关联、将“BLACK”与票据“0x3819e6c3”相关联、以及将“PURPLE”与票据“0xb381efd2”相关联。因此，响应于对所有“RED”对象的查询，LongLongMap可以使用票据“0x112ca533”来识别引用存储库520，并且因此返回引用集合530。引用集合530识别零拷贝二进制基数树240中与用于满足索引值“RED”的值/对象的关键字对应的节点。因此，系统可

以响应于查询返回满足索引值“RED”的所有值/对象。当用于特定索引的引用存储库实例被扩充或缩小时，与特定索引相关联的票据可以改变。因此，当用于特定索引的引用存储库实例被扩充或缩小时，新票据被指派给LongLongMap 510中的索引。

[0083] 此外，取决于满足特定索引值的值/对象的普遍性(prevalence)，引用存储库实例中的每个引用存储库实例中的引用的数量可以明显不同。例如，“PURPLE”对象可能比“BLACK”对象明显地不那么普遍。为了优化引用存储库实例的存储器使用，引用存储库实例可以取决于存储的引用的数量而利用不同的数据结构。这与set<key>的现有实现不同，例如，该现有实现使用散列集合将索引值与关键字相关联，而无论集合中的关键字的数量有多少。这种实现在存储器使用和访问成本方面是昂贵的。尤其在引用的数量少的情况下，散列集合的使用具有比所需的更高的存储器和访问开销。

[0084] 因此，引用存储库500提供了针对存储不同大小的引用组在存储器使用和访问效率方面更高效的机制。引用存储库500基于存储库中的引用的数量来实现与需要存储与索引值对应的引用的索引的合约(contract)。例如，在存在64个或更少的引用的情况下，引用存储库500利用基于位掩码的引用存储库实例来存储引用。每当提供要存储的引用时，该引用被放置在基于位掩码的引用存储库实例中的64个槽中的不同的槽中。当所有槽都已满时，如果接收到另一个要存储的引用，则引用存储库实例被扩充。

[0085] 为了扩充引用存储库实例，新票据与索引相关联，其中该新票据引用新引用存储库，该新引用存储库取决于引用的数量可以是基于列表或基于散列的。新引用存储库实例可以在此时被创建或者可以已经被分配并且当前未被使用。在先前基于位掩码的引用存储库中的引用被迁移到新引用存储库。引用存储库保留用于现在未使用的基于位掩码的引用存储库的存储器分配，以使得如果需要的话该存储器分配可以被指派给另一个索引。

[0086] 基于列表的引用存储库实例是存储器高效的，并且可以保持无限数量的引用。然而，简单的基于列表的引用存储库缺乏散列语义，并且因此访问存储库中的引用的转换(cast)随着引用数量的增加而增加。因此，引用存储库利用基于列表的引用存储库实例来存储用于特定索引的引用直至达到阈值数量的引用，在达到阈值数量的引用的情况下访问成本太高而不能继续使用基于列表的引用存储库实例。在该点处，如果还有引用被添加到引用存储库实例，则引用存储库实例再次被扩充。

[0087] 为了扩充基于列表的引用存储库实例，新票据与索引相关联，其中该新票据引用基于散列的引用存储库实例。基于散列的引用存储库实例可以在此时被创建或者可以已经被分配并且当前未被使用。在先前基于列表的引用存储库中的引用被迁移到新的基于散列的引用存储库。引用存储库保留用于现在未使用的基于列表的引用存储库的存储器分配，以使得如果需要的话该存储器分配可以被指派给另一个索引。基于散列的引用存储库实例需要比基于列表的引用存储库实例更多的存储器，但是当存在大量引用时，散列语义降低了访问存储库中的引用的成本。基于散列的引用存储库被实现为开放散列集合，并且是最大的引用存储库实现。

[0088] 引用存储库500还可以缩小引用存储库实例，其中引用存储库中的引用的数量被减少，以使得引用可以被存储在较小的引用存储库实例中的一个引用存储库实例中。因此，可以通过提供用于基于列表的引用存储库的票据并且迁移引用来将基于散列的引用存储库实例缩小为基于列表的引用存储库实例。此外，可以通过提供用于基于位掩码的引用存

储库的票据并且迁移引用来将基于列表的引用存储库实例缩小为基于位掩码的引用存储库实例。在所有情况下，在迁移之后，可以维持未使用的数据结构，直至它被不同的索引需要，从而减少创建新的引用存储库以及为节点/JVM请求新的存储器分配的需要。在实施例中，引用存储库可以基于对每种类型的引用存储库实例的典型需求的理解，为每种类型的引用存储库实例预分配存储器。

[0089] 图6示出了根据本发明的实施例的使用引用存储库来存储用于索引的引用的方法。引用存储库为多个索引服务中的每个索引服务执行相同的方法。在步骤600处，该方法开始，并且实现与需要存储引用的索引的合约的引用存储库被创建。在步骤602处，索引尝试存储第一引用。当第一引用被添加时，基于位掩码的引用存储库被创建和使用。在步骤604处，引用存储库将索引与指向基于位掩码的引用存储库实例的票据相关联，并且将该票据返回到引用者。在步骤606处，第一引用被存储在基于位掩码的引用存储库实例中。在步骤608处，索引将引用添加到基于位掩码的引用存储库实例或从基于位掩码的引用存储库实例移除引用。在步骤610处，如果在添加或移除引用之后引用计数将＜＝64，则该方法返回到步骤608-即，基于位掩码的引用存储库实例继续被用于引用。在步骤610处，如果尝试添加第六十五个引用，则该方法移动到步骤612，以扩充引用存储库实例。

[0090] 在步骤612处，与索引相关联的引用存储库实例被扩充。引用存储库将新票据与索引相关联，并且该新票据引用基于列表的引用存储库实例。可替代地，列表可以被直接扩充为基于散列的引用存储库(步骤624)，并且基于列表的引用存储库仅在从基于散列的引用存储库缩小时才被使用。在步骤614处，引用65被添加到基于列表的引用存储库实例，并且其它64个引用被迁移到基于列表的引用存储库实例。在步骤616处，索引将引用添加到基于列表的引用存储库实例或从基于列表的引用存储库实例移除引用。在步骤618处，如果引用计数大于缩小阈值并且小于扩充阈值，则该方法返回到步骤616，即，基于列表的引用存储库实例继续被用于存储用于索引的引用。

[0091] 在步骤618处，如果引用计数小于或等于缩小阈值，则该方法移动到步骤620，以缩小引用存储库实例。缩小要求引用存储库中产生至少75％的浪费，或者与引用存储库中的总共的槽相比存在25％或更少的引用。还存在对初始使用的补偿，因此在使用稳定之前不发生缩小。在步骤618处，如果引用计数大于或等于扩充阈值，则该方法移动到步骤624以扩充引用存储库实例。

[0092] 在步骤620处，缩小基于列表的引用存储库包括将索引与指向基于位掩码的引用存储库实例的票据相关联。在步骤622处，引用从基于列表的引用存储库实例迁移到基于位掩码的引用存储库实例。然后，该方法返回到步骤608，在步骤608中引用可以被添加到基于位掩码的引用存储库实例或从基于位掩码的引用存储库实例移除。用于从基于列表的引用存储库缩小到基于位掩码的引用存储库的阈值可以是例如用于从基于位掩码的引用存储库扩充到基于列表的引用存储库的阈值的80％，以防止过度的迁移循环。

[0093] 在步骤624处，扩充基于列表的引用存储库实例包括将索引与指向基于散列的引用存储库实例的票据相关联。在步骤626处，(超过扩充阈值的)引用被添加到基于散列的引用存储库实例，并且已经存储的引用被迁移到基于散列的引用存储库实例。在步骤628处，引用可以被添加到与票据相关联的基于散列的引用存储库实例或从与票据相关联的基于散列的引用存储库实例移除。在步骤630处，如果引用计数大于缩小阈值，则该方法移动到

步骤628，即，索引继续使用与票据相关联的基于散列的引用存储库实例来存储引用。注意，由于基于散列的引用存储库是所使用的最大存储库，因此扩充是不可能的。在步骤630处，如果引用计数小于或等于缩小阈值，则该方法移动到步骤632，以缩小引用存储库实例。缩小要求在引用存储库中产生至少75％的浪费，或者与引用存储库中的总共的槽相比存在25％或更少的引用。还存在对初始使用的补偿，因此在使用稳定之前不会发生缩小，以防止过度的迁移循环。

[0094] 在步骤632处，缩小基于散列的引用存储库实例包括将索引与指向基于列表的引用存储库实例的票据相关联。在步骤634处，(低于阈值的)引用被移除，并且剩余的引用被迁移到基于列表的引用存储库实例。然后，该方法返回到步骤616，在步骤616中索引可以将引用添加到与票据相关联的基于列表的引用存储库实例或从与票据相关联的基于列表的引用存储库实例移除引用。

[0095] 一种实施例包括用于支持分布式计算环境中的引用存储库的方法，该方法包括：创建包括基于位掩码的存储库的引用存储库；在引用存储库中存储引用；将票据与引用存储库相关联；向一个或多个消费者提供票据；向基于位掩码的引用存储库添加多个引用；以及如果引用存储库中的引用的数量超过阈值，则扩充引用存储库。

[0096] 一种实施例包括用于如果引用存储库中的引用的数量超过阈值则扩充引用存储库的方法，该方法包括将引用存储库扩充为包括基于列表的引用存储库和基于散列的引用存储库之一的新引用存储库；将引用从引用存储库迁移到新引用存储库；将新票据与新引用存储库相关联；以及将新票据提供给所述一个或多个消费者。

[0097] 一种实施例包括用于如果引用存储库中的引用的数量超过阈值则扩充引用存储库的方法，该方法包括：将引用存储库扩充为包括基于列表的引用存储库和基于散列的引用存储库之一的新引用存储库；将引用从引用存储库迁移到新引用存储库；以及向所述一个或多个消费者发信号通知关于引用存储库的扩充。

[0098] 一种实施例包括用于将票据与索引值相关联的方法。

[0099] 一种实施例包括用于创建与多个票据相关联的多个引用存储库的方法，其中该多个引用存储库中的每个引用存储库与不同的索引值或多个索引值相关联，并且提供将该多个票据与该多个索引值相关联的映射。

[0100] 一种实施例包括用于响应于引用被添加到引用存储库或从引用存储库移除而扩充或缩小引用存储库的方法。

[0101] 一种实施例包括用于提供零拷贝二进制基数树的方法，并且其中引用存储库中的多个引用中的每个引用引用零拷贝二进制基数树中的节点。

[0102] 一种实施例包括用于在扩充或缩小之后回收引用存储库存储器结构的方法。

[0103] 一种实施例包括用于响应于被添加的引用超过扩充阈值而扩充引用存储库，以及响应于引用被移除以致于剩余引用的数量小于缩小阈值而缩小引用存储库的方法。

[0104] 一种实施例包括其中分布式计算环境是分布式数据网格的方法，并且其中该方法还包括使用引用存储库来保持与在分布式数据网格中高速缓存的一组对象相关联的一组引用。

[0105] 一种实施例包括用于支持分布式计算环境中的关键字管理的系统，该系统包括：包括微处理器和存储器的计算机系统；在所述计算机系统上操作的所述分布式计算环境的

服务器节点;并且其中服务器节点被配置为:在分配给所述节点的所述计算机系统的存储器中创建包括基于位掩码的存储库的引用存储库,在引用存储库中存储引用,将票据与引用存储库相关联,将票据提供给一个或多个消费者,将多个引用添加到基于位掩码的引用存储库,以及如果引用存储库中的引用的数量超过阈值,则扩充引用存储库。

[0106] 一种实施例包括系统,其中服务器节点被配置为通过以下操作来扩充引用存储库:创建包括基于列表的引用存储库和基于散列的引用存储库之一的新引用存储库;将引用从引用存储库迁移到新引用存储库;将新票据与新引用存储库相关联;以及将新票据提供给所述一个或多个消费者。

[0107] 一种实施例包括系统,其中服务器节点被配置为通过以下操作来扩充引用存储库:将引用存储库扩充为包括基于列表的引用存储库和基于散列的引用存储库之一的新引用存储库;将引用从引用存储库迁移到新引用存储库;以及向所述一个或多个消费者发信号通知关于引用存储库的扩充。

[0108] 一种实施例包括系统,其中票据与索引值相关联。

[0109] 一种实施例包括系统,其中所述服务器节点被配置为:创建与多个票据相关联的多个引用存储库,其中该多个引用存储库中的每个引用存储库与不同的索引值或多个索引值相关联;以及提供将该多个票据与该多个索引值相关联的映射。

[0110] 一种实施例包括系统,其中所述服务器节点被配置为响应于引用被添加到引用存储库或从引用存储库移除而扩充或缩小引用存储库。

[0111] 一种实施例包括系统,其中所述服务器节点还被配置为:提供零拷贝二进制基数树;以及其中引用存储库中的多个引用中的每个引用是对零拷贝二进制基数树中的节点的引用。

[0112] 一种实施例包括系统,其中所述服务器节点还被配置为在扩充或缩小之后回收引用存储库存储器结构。

[0113] 一种实施例包括系统,其中分布式计算环境是分布式数据网格,并且其中服务器节点被配置为使用引用存储库来保持与在分布式数据网格中高速缓存的一组对象相关联的一组引用。

[0114] 一种实施例包括非暂态计算机可读介质,该非暂态计算机可读介质包括存储在其上的用于支持分布式计算环境中的引用存储库的指令,这些指令当被执行时,使得分布式计算环境中的服务器节点执行步骤,这些步骤包括:创建包括基于位掩码的存储库的引用存储库;在引用存储库中存储引用;将票据与引用存储库相关联;将票据提供给一个或多个消费者;将多个引用添加到基于位掩码的引用存储库;以及如果引用存储库中的引用的数量超过阈值,则扩充引用存储库。

[0115] 虽然上文已经描述了本发明的各种实施例,但是应当理解的是,它们是作为示例而不是限制给出的。对于相关领域的技术人员将显而易见,在不脱离本发明的精神和范围的情况下,可以在其中进行形式和细节的各种改变。

[0116] 本发明的许多特征可以在硬件、软件、固件或其组合中、使用硬件、软件、固件或其组合或者在硬件、软件、固件或其组合的辅助下执行。可以使用包括根据本公开的教导编程的一个或多个处理器、存储器和/或计算机可读存储介质的一个或多个常规通用或专用数字计算机、计算设备、机器或微处理器方便地实现本发明。还可以使用例如诸如专用集成电

路(ASIC)和可编程逻辑器件之类的硬件组件在硬件中实现本发明的特征。为了执行本文所描述的功能的硬件状态机的实现对于相关领域的技术人员将显而易见。

[0117] 本发明的特征可以被结合在软件和/或固件中,以用于控制处理系统的硬件以及用于使得处理系统能够利用本发明的结果与其它机制交互。这样的软件或固件可以包括但不限于应用代码、设备驱动程序、操作系统和执行环境/容器。如对软件领域的技术人员将显而易见的,基于本公开的教导,熟练的程序员可以容易地准备适当的软件编码。

[0118] 在一些实施例中,本发明包括计算机程序产品,该计算机程序产品是具有存储在其上/其中的指令的存储介质或计算机可读介质(媒介),其中该指令可以被用来对计算机编程以执行本发明的过程中的任何过程。存储介质或计算机可读介质可以包括但不限于任何类型的盘(包括软盘、光盘、DVD、CD-ROM、微驱动器和磁光盘)、ROM、RAM、EPROM、EEPROM、DRAM、VRAM、闪存存储器设备、磁卡或光卡、纳米系统(包括分子存储器IC)、或者适于存储指令和/或数据的任何类型的介质或设备。在实施例中,存储介质或计算机可读介质可以是非暂态的。

[0119] 为了说明和描述的目的,提供了本发明的前述描述。它不旨在是详尽的或将本发明限制为所公开的精确形式。许多修改和变化对于本领域技术人员将是显而易见的。修改和变化包括所公开的特征的任何相关组合。选择和描述了实施例以便于最佳地解释本发明的原理及其实践应用,从而使得本领域其他技术人员能够针对各种实施例以及利用具有适于预期的特定使用的各种修改来理解本发明。旨在由以下权利要求及其等价物来限定本发明的范围。

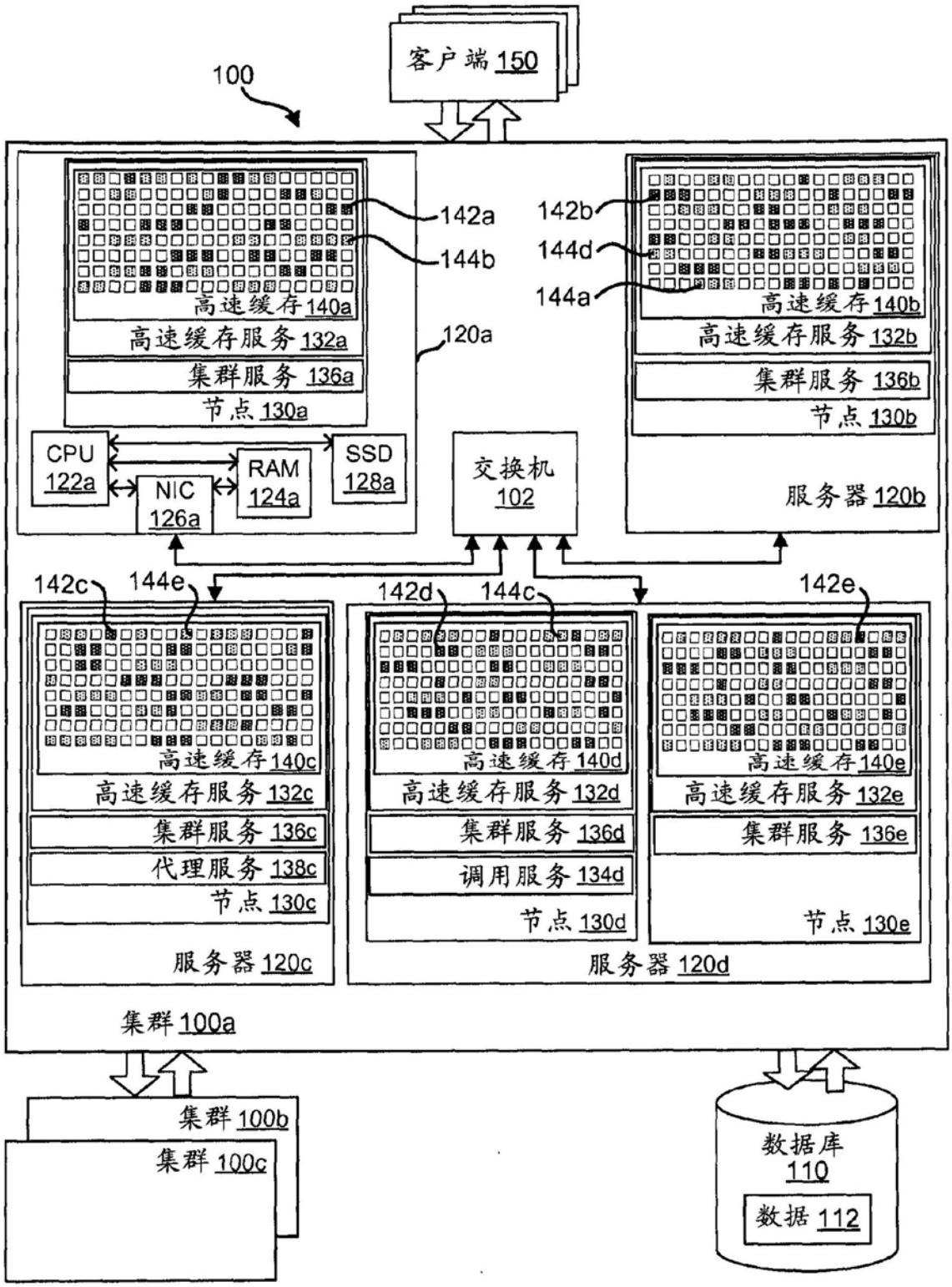

100
客户端 150
142a
144b
高速缓存 140a
高速缓存服务 132a
集群服务 136a
节点 130a
120a
CPU 122a
NIC 126a
RAM 124a
SSD 128a
142b
144d
144a
高速缓存 140b
高速缓存服务 132b
集群服务 136b
节点 130b
服务器 120b
交换机 102
142c
144e
高速缓存 140c
高速缓存服务 132c
集群服务 136c
代理服务 138c
节点 130c
服务器 120c
142d
144c
高速缓存 140d
高速缓存服务 132d
集群服务 136d
调用服务 134d
节点 130d
142e
高速缓存 140e
高速缓存服务 132e
集群服务 136e
节点 130e
服务器 120d
集群 100a
集群 100b
集群 100c
数据库 110
数据 112

图1

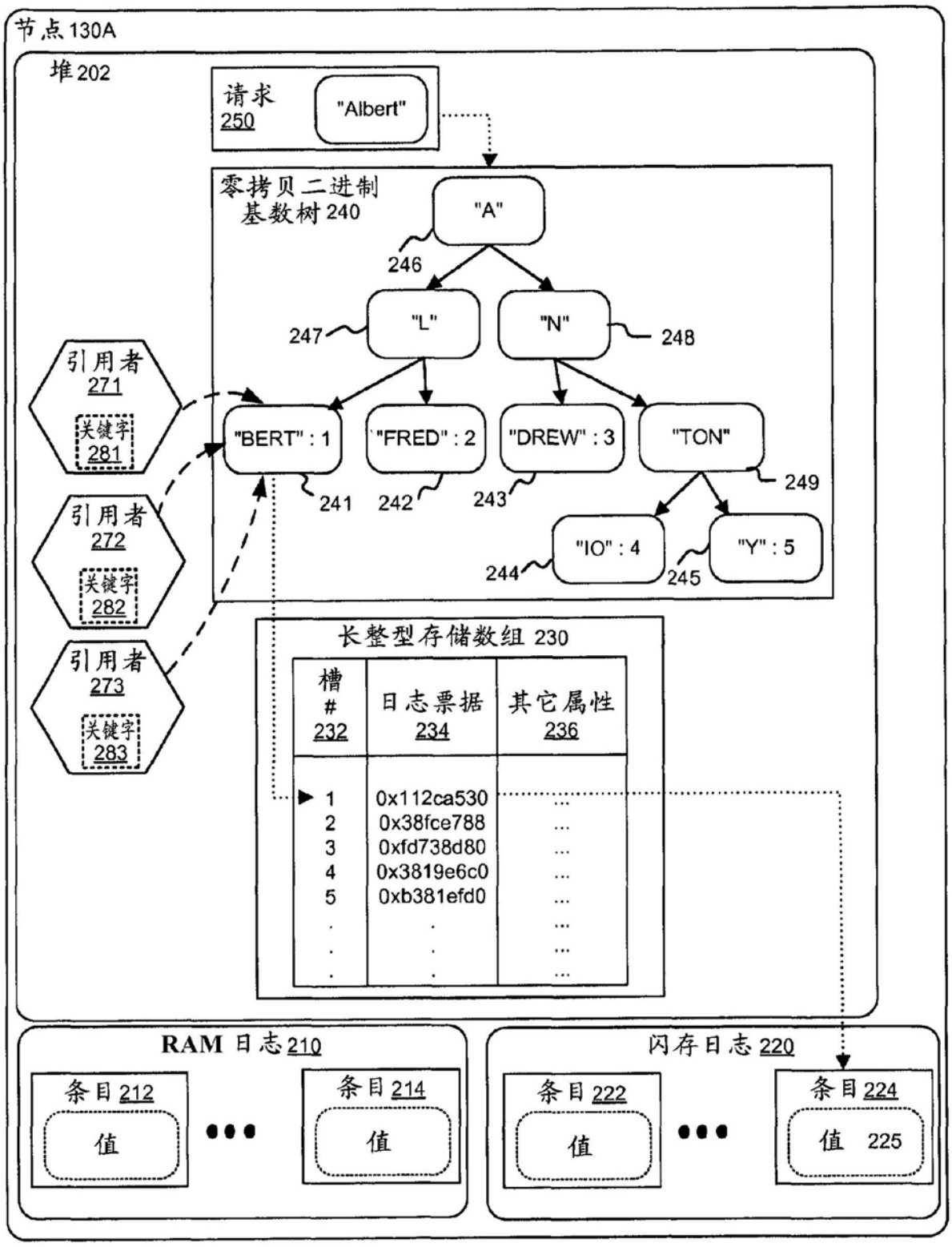

节点130A
堆202
请求 250
"Albert"
零拷贝二进制基数树240
"A"
246
"L"
"N"
247
248
"BERT" : 1
"FRED" : 2
"DREW" : 3
"TON"
241
242
243
249
"IO" : 4
"Y" : 5
244
245
引用者 271
关键字 281
引用者 272
关键字 282
引用者 273
关键字 283
长整型存储数组 230
槽 # 232
日志票据 234
其它属性 236
1 0x112ca530 ...
2 0x38fce788 ...
3 0xfd738d80 ...
4 0x3819e6c0 ...
5 0xb381efd0 ...
RAM 日志 210
条目 212
值
条目 214
值
闪存日志 220
条目 222
值
条目 224
值 225

图2

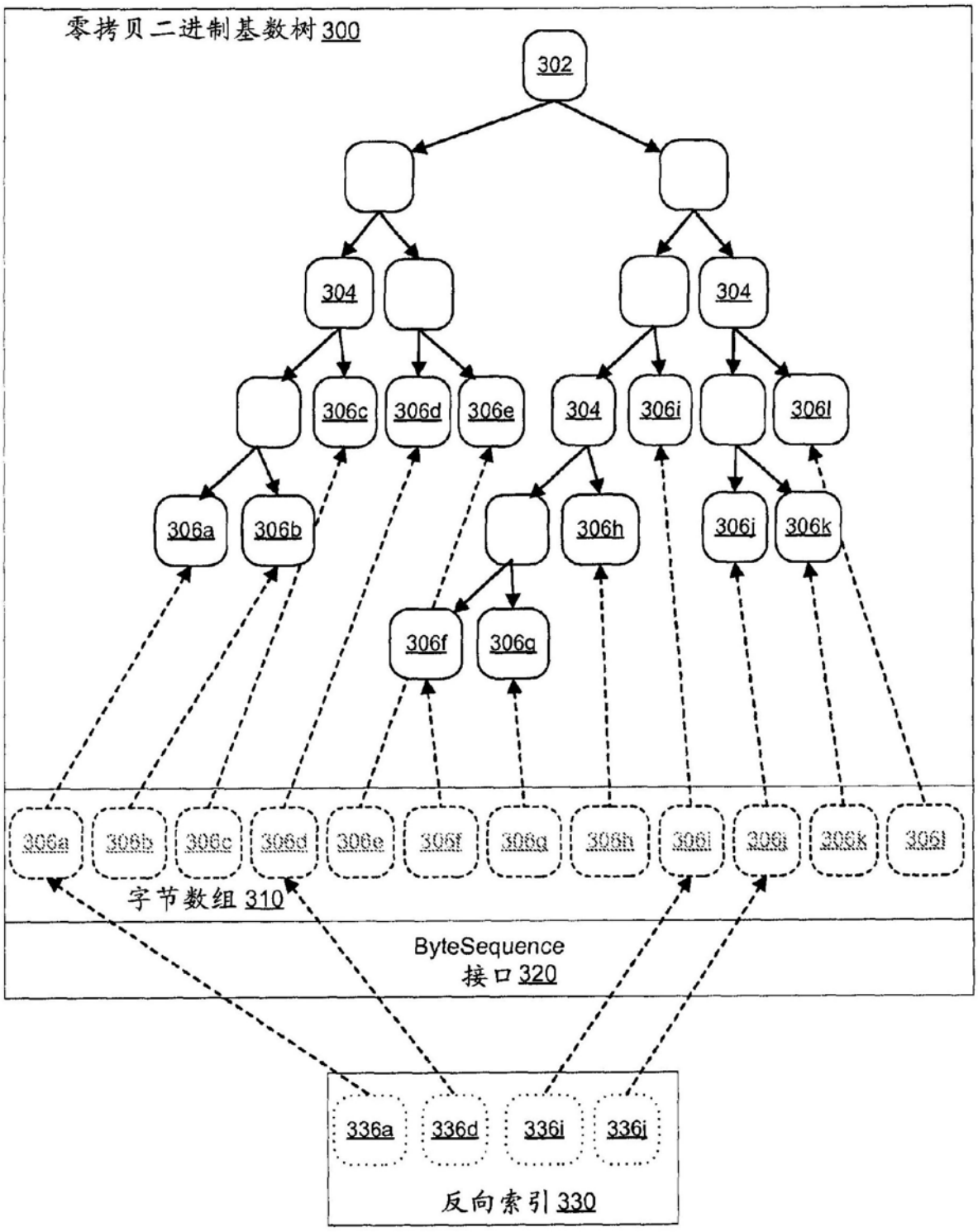

零拷贝二进制基数树 300
302
304
304
306c
306d
306e
304
306i
306l
306a
306b
306h
306j
306k
306f
306g
306a
306b
306c
306d
306e
306f
306g
306h
306i
306j
306k
306l
字节数组 310
ByteSequence
接口 320
336a
336d
336i
336j
反向索引 330

图3A

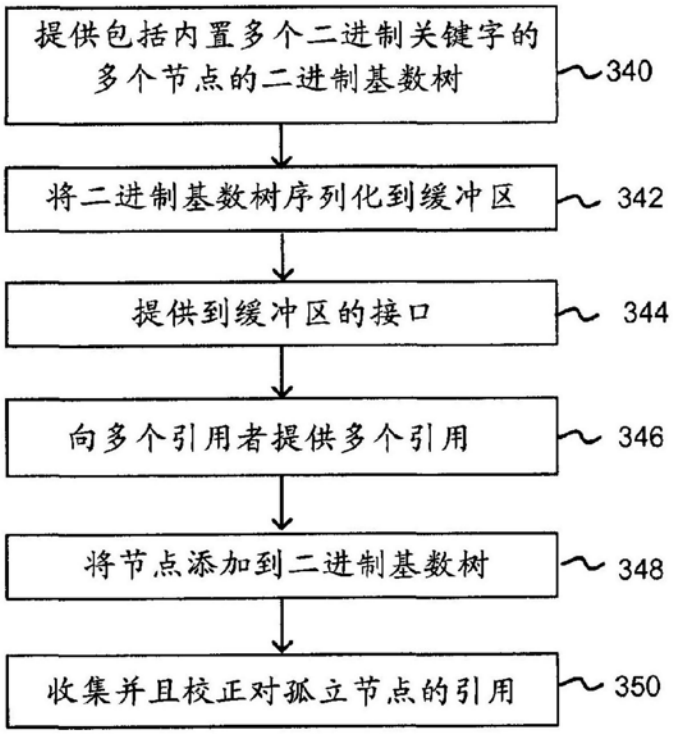

提供包括内置多个二进制关键字的多个节点的二进制基数树
340
将二进制基数树序列化到缓冲区
342
提供到缓冲区的接口
344
向多个引用者提供多个引用
346
将节点添加到二进制基数树
348
收集并且校正对孤立节点的引用
350

图3B

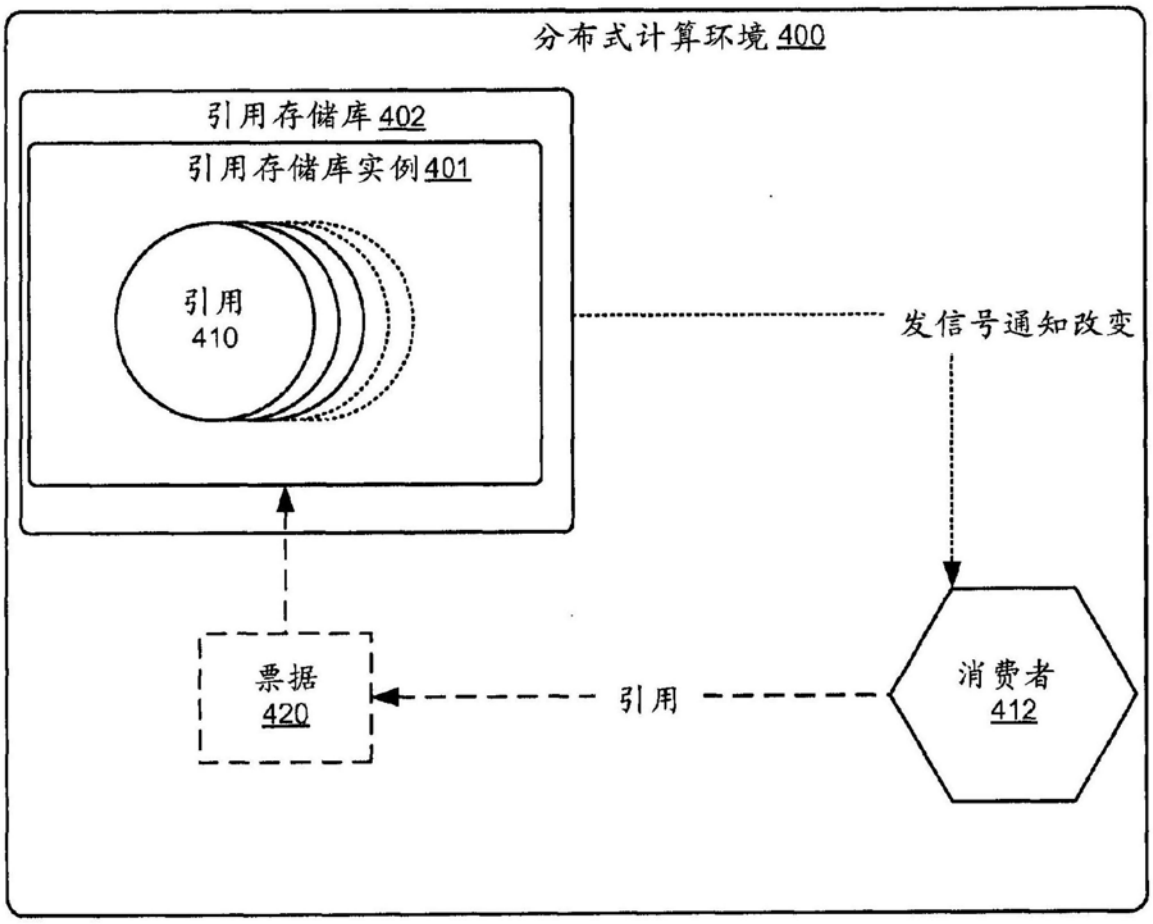

分布式计算环境 400
引用存储库 402
引用存储库实例 401
引用
410
发信号通知改变
票据
420
引用
消费者
412

图4

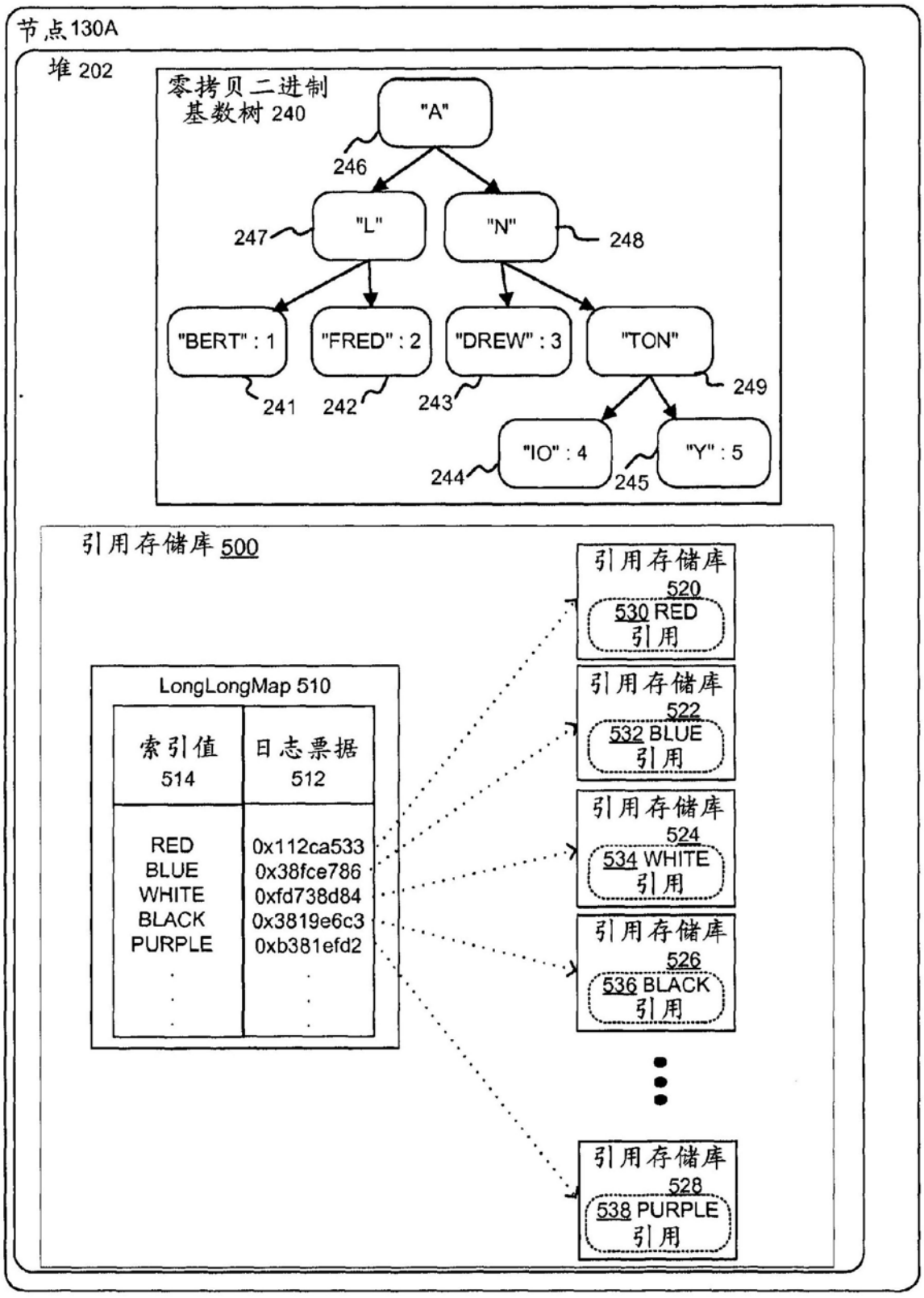

节点130A
堆 202
零拷贝二进制
基数树 240
"A"
246
"L"
247
"N"
248
"BERT" : 1
241
"FRED" : 2
242
"DREW" : 3
243
"TON"
249
"IO" : 4
244
"Y" : 5
245
引用存储库 500
LongLongMap 510
索引值
514
日志票据
512
RED
BLUE
WHITE
BLACK
PURPLE
0x112ca533
0x38fce786
0xfd738d84
0x3819e6c3
0xb381efd2
引用存储库
520
530 RED
引用
引用存储库
522
532 BLUE
引用
引用存储库
524
534 WHITE
引用
引用存储库
526
536 BLACK
引用
引用存储库
528
538 PURPLE
引用

图5

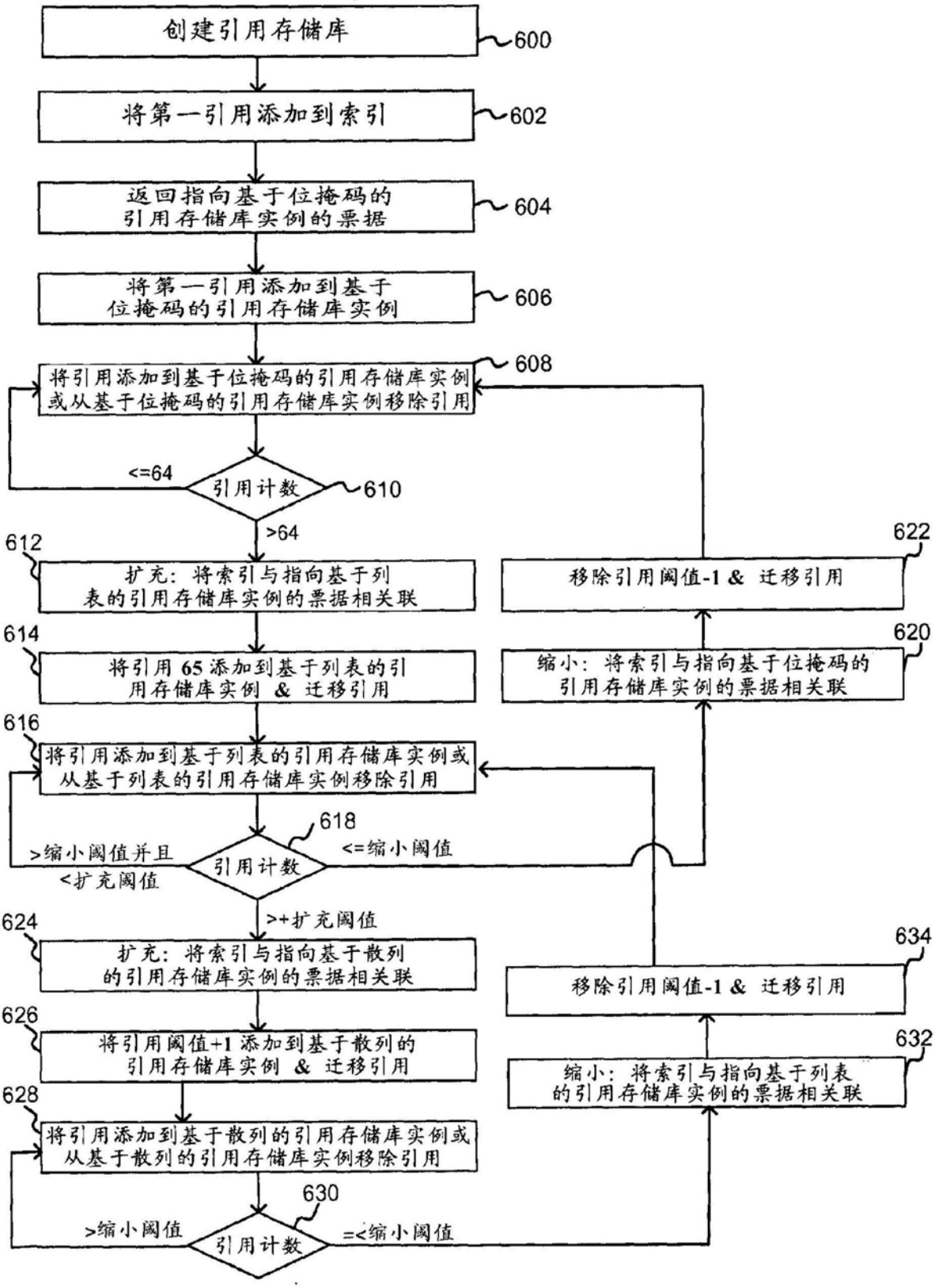

创建引用存储库
600
将第一引用添加到索引
602
返回指向基于位掩码的引用存储库实例的票据
604
将第一引用添加到基于位掩码的引用存储库实例
606
将引用添加到基于位掩码的引用存储库实例或从基于位掩码的引用存储库实例移除引用
608
<=64
引用计数
610
>64
612
扩充：将索引与指向基于列表的引用存储库实例的票据相关联
614
将引用65添加到基于列表的引用存储库实例 & 迁移引用
616
将引用添加到基于列表的引用存储库实例或从基于列表的引用存储库实例移除引用
618
>缩小阈值并且<扩充阈值
引用计数
<=缩小阈值
>+扩充阈值
624
扩充：将索引与指向基于散列的引用存储库实例的票据相关联
626
将引用阈值+1添加到基于散列的引用存储库实例 & 迁移引用
628
将引用添加到基于散列的引用存储库实例或从基于散列的引用存储库实例移除引用
630
>缩小阈值
引用计数
=<缩小阈值
632
缩小：将索引与指向基于列表的引用存储库实例的票据相关联
634
移除引用阈值-1 & 迁移引用
620
缩小：将索引与指向基于位掩码的引用存储库实例的票据相关联
622
移除引用阈值-1 & 迁移引用

图6