Manu：一个云原生向量数据库管理系统

郭、滦小凡、龙翔、小燕、小一、罗日高[†§]
程莎桂、徐伟志[†]罗家瑞[‡]，弗兰克刘[†]曹贞贞[†]乔彦良[†]王挺[†]
波唐沙桂，谢查尔斯[†]
南方科技大学慕尼黑技术大学
[†]{firstname.lastname}@zilliz.com
‡ {xiangl3@mail., yanx@, 11911419@mail., tangb3@}sustech.edu.cn [§] jigao.luo@tum.de

摘要

随着基于学习的嵌入模型的发展，嵌入向量被广泛应用于非结构化数据的分析和搜索。由于向量收集超过十亿规模，需要完全管理和水平可伸缩的向量数据库。在过去的三年里，通过与我们的1200名+行业用户的交互，我们为下一代向量数据库应该拥有的特性勾画了一个愿景，包括长期的可进化性、可调的一致性、良好的弹性和高性能。

我们提供了Manu，一个实现这些特性的云原生向量数据库。如果我们遵循传统的DBMS设计规则，那么很难集成所有这些特性。由于大多数向量数据应用程序不需要复杂的数据模型和强大的数据一致性，我们的设计理念是放松数据模型和一致性约束，以换取上述特性。具体来说，Manu首先公开了预写日志（WAL）和binlog作为骨干服务。其次，写组件被设计为日志发布器，而所有的只读分析和搜索组件都被设计为日志服务的独立订阅者。最后，我们利用多版本并发控制（MVCC）和增量一致性模型来简化系统组件之间的通信和协作。这些设计实现了系统组件之间的低耦合，这对弹性和演化至关重要。我们还通过具有硬件感知的实现和支持复杂的搜索语义来广泛优化Manu的性能和可用性。Manu已被用于许多应用程序，包括，但不限于，推荐、多媒体、语言、医学和安全。我们在三个典型的应用场景中评估了Manu，以演示其效率、弹性和可伸缩性。

PVLDB参考格式：

郭仁通、滦小凡、龙翔、小燕、易小孟、罗日高、程启雅、徐伟志、罗家瑞、刘伟郎、曹珍山、乔阎良、王婷、唐波、谢查理。Manu：一个云原生向量数据库管理系统。PVLDB, 15(12): 3548 - 3561, 2022.doi:10.14778/3554821.3554843

*合著者按字母顺序排列。

‡与齐利兹一起工作时完成的工作，通信给唐波。

《VLDB捐赠基金的诉讼程序》，第1卷。15, No.12 ISSN 2150-8097。
doi:10.14778/3554821.3554843

PVLDB工件可用性：

源代码、数据和/或其他工件已在https://github.com/milvus-io/milvus/tree/2.0.

1介绍

根据IDC的数据，在2020年生成的4万eb字节的新数据中，文本、图像和视频等非结构化数据约占80%，由于人工生成的富媒体[47]数量的增加，其比例持续上升。随着基于学习的嵌入模型的兴起，特别是深度神经网络，利用嵌入向量来管理非结构化数据已经在电子商务、社交媒体和药物发现[48,62,67]等许多应用中变得很普遍。这些应用程序的一个核心特性是，它们将非结构化数据的语义编码为一个高维向量空间。考虑到嵌入向量的表示能力，可以通过基于相似度的向量搜索来实现推荐、搜索和向量搜索等操作。为了支持这些应用程序，许多专门的向量数据库都是这样做的
用于管理向量数据的构建[10、12、17-19、80]。

在2019年，我们开放了Milvus [80]，我们之前的向量数据-基地，隶属于LF人工智能与数据基金会。从那时起，我们收集从1200多个行业用户的反馈中发现，Milvus所采用的一些设计原则并不合适。Milvus遵循了关系数据库的设计原则，这些数据库针对事务[51]或分析[80]工作负载进行了优化，并专注于功能支持(e。g., 属性过滤和多向量搜索)和执行效率（例如，SIMD和缓存优化）。然而，矢量数据库应用程序在以下三个方面有不同的需求，这促使我们从头开始重组Manu，重点关注云本地架构。

不需要支持复杂的事务。基于学习的模型不是将实体表示分解为单个向量，而不是将复杂的混合数据语义编码为不同的字段或表。因此，不需要多行或多表事务；行级的ACID对于大多数向量数据库应用程序来说就足够了。

可调的性能一致性权衡是很重要的。不同的用户有不同的一致性要求；一些用户更喜欢高吞吐量和最终的一致性，而另一些用户则需要一定程度的保证一致性，即，新插入的数据应该立即或在预先配置的时间内对查询可见。传统的关系数据库通常支持强一致性或最终一致性；这些数据库之间几乎没有定制的空间

两个极端。因此，可调谐的一致性是云原生向量数据库的一个关键属性。

较高的硬件成本需要具有细粒度的弹性。一些向量数据库操作（例如，向量搜索和索引构建）是计算密集型的，而硬件加速器(e。g.gpu或fpga)和/或一个大的工作内存需要良好的性能。但是，根据应用程序类型的不同，数据库功能之间的工作负载会有所不同。因此，如果向量数据库没有细粒度的弹性，那么资源可能会被浪费或分配不当。这就需要仔细地将功能层和硬件层解耦；系统级别的解耦，如读写逻辑的分离是不够的，弹性和资源隔离应该在功能级别而不是系统级别进行管理。

总之，现代向量数据库应该具有可调谐的一致性、功能级的解耦和每个组件的可伸缩性。遵循传统关系数据库的设计原则使得实现这些设计目标变得非常困难，如果不是不可能的话。实现这些设计目标的一个关键机会在于放松交易复杂性的可能性。

Manu遵循“以日志作为数据”的范式。具体来说，Manu将整个系统构建为一组日志发布/订阅微服务。预写日志（WAL）和组件间的消息以“日志”的形式发布，即可以订阅的持久数据流。读取端组件，如搜索和分析引擎，都是作为日志订阅者构建的。该体系结构提供了一种简单而有效的方法来解耦系统功能；它支持读与写、无状态与有状态、存储与计算的解耦。每个日志条目都被分配了一个全局唯一的时间戳，并且被称为时间标记的特殊日志条目（类似于ApacheFlicelink[25]中的水印）被定期插入到每个日志通道中，为日志订阅者发送事件时间的进展。时间戳和时间标记构成了可调谐的一致性机制和多版本一致性控制（MVCC）的基础。为了控制一致性级别，用户可以指定查询的时间戳和订阅者使用的最新时间标记之间的一个可容忍的时间延迟。

此外，我们还广泛地优化了Manu的性能和可用性。Manu支持向量搜索的各种索引，包括向量量化[21,33,36,82]、反向索引[23]和接近图[32]。特别是，我们定制了实现，以更好地利用现代cpu和gpu的并行化能力，以及ssd比hdd改进的读/写速度。Manu还集成了来自Milvus [80]的重构功能，如属性过滤和多向量搜索。此外，构建一个可视化工具，允许用户实时跟踪Manu的性能，并包括一个自动配置工具，推荐使用机器学习索引算法参数。

综上所述，本文的贡献如下：

我们总结了通过与他人交流而得到的经验教训

在三年内拥有1200名行业用户。我们阐明了向量数据库的典型应用需求，并展示了它们与传统关系数据库的不同。然后，我们概述了矢量数据库应该满足的关键设计目标。

我们将Manu的关键架构设计作为云原生向量数据库，围绕放松事务复杂性的核心设计理念，以换取可调的一致性和细粒度弹性。

我们提出了重要的可用性和与性能相关的增强，e。g.，高级API、GUI工具、自动参数配置和SSD支持。

本文的其余部分组织如下。第2节提供了关于载体数据库的需求和设计目标的背景知识。第三节深入探讨了马努的设计。第4节强调了可用性和性能的关键特性。第5节讨论了Manu的代表性用例。第六节审查相关工作。第7节总结了论文，并概述了未来的工作。

2背景和动机

将视频推荐视为向量数据库的一个典型用例。其目标是帮助用户根据个人偏好和以前的浏览历史发现新的视频。使用机器学习模型（特别是深度神经网络），用户和视频的特征，如搜索历史、观看历史、年龄、性别、视频语言和标签被转换为嵌入向量。这些模型经过精心设计和训练，以将用户和视频向量之间的相似性编码到一个共同的向量空间。推荐是通过对指定用户向量的相似度评分从视频向量集合中检索候选视频来进行的。当新视频更新、删除部分视频、改变嵌入模型时，系统还需要处理向量的更新。

载体数据库的视频推荐和其他应用可以涉及数千亿次的载体，每天以数亿次的规模增长，并为每秒提供数百万次规模的查询（QPS）。现有的dbms（例如，关系数据库[9,11]、NoSQL [75,85]、NewSQL [39,73]）没有构建来管理该规模的向量数据。此外，它们的应用程序的底层数据管理需求与矢量数据库应用程序有很大的不同。 首先，与关系数据库相比，向量数据库的体系结构和理论还远未成熟。这样做的一个关键原因是，AIand数据驱动的应用程序仍然处于不断发展的状态，因此也需要继续对向量数据库进行架构和功能更改。

其次，矢量数据库不需要进行复杂的事务。在上面的示例中，推荐系统将用户和视频的所有任务特征编码为独立的向量，而不是在关系数据库中的多行或多列实体字段。因此，行级ACID就足够了；多表选择（如连接）是不必要的。

第三，矢量数据库应用程序需要一个灵活的性能、经济一致性的权衡。虽然有些应用程序采用了强的或最终的一致性模型，但也有一些应用程序介于这两个极端之间。用户可能希望放松一致性约束，以换取更好的系统吞吐量。在视频推荐示例中，在几秒钟后观察新上传的视频是可以接受的，但让用户等待推荐会损害用户体验。因此，该应用程序可以配置

允许最大的视频更新延迟，以提高系统吞吐量。

第四，与传统数据库相比，载体数据库的硬件要求更为严格和多样化。这主要有三个原因。首先，向量数据库操作是计算密集型的，因此像gpu这样的硬件加速器对于搜索和索引等计算功能至关重要。其次，对向量数据的访问（如搜索或更新）的局部性通常较差，因此需要较大的RAM才能获得良好的性能。第三，不同的应用程序对系统功能的资源需求有很大差异。

向量数据库的核心功能包括数据插入，索引、过滤和向量搜索。诸如视频推荐等应用程序需要在线插入和高并发性向量搜索。相比之下，对于药物发现等交互式用例，离线数据摄入和索引通常是可以接受的。尽管交互式应用程序通常比推荐系统需要更低的吞吐量，但它们对实时过滤、基于相似度的向量搜索和混合查询有很高的要求。高的硬件成本以及多样化的工作负载特性需要细粒度的弹性。

Manu的主要设计目标总结如下；这些设计目标不仅完全包含了上述特征，而且与一些基于云的通用数据库共享一些共同的目标。

长期演化性：为了马努功能的不断演化，必须控制整个系统的整体复杂性。如果不需要支持复杂的事务，那么就有机会将所有事件序列（如WAL和组件间的消息）建模为消息队列，以干净地解耦整个系统。通过这种方式，单个组件可以演化、添加或轻松替换，而对其他组件的干扰最小。这种设计呼应了大规模的数据分析平台，这些平台通常依赖于像Kafka这样的数据流系统来连接系统组件。

可调一致性：为了实现灵活的一致性-性能权衡，Manu应该引入介于强一致性和最终一致性之间的增量一致性，其中读取操作返回在最多增量时间单位之前产生的最后一个值。值得注意的是，该模型的特殊情况可以实现强一致性和最终一致性，分别为零和无穷大。

良好的弹性：工作负载的波动会对单个系统组件造成不同的负载。为了动态地分配计算资源到高负载的任务，组件必须仔细地解耦，同时考虑到功能和硬件依赖性。系统弹性和资源隔离应该在组件级进行管理，而不是在系统级进行管理。将索引与查询解耦与读写解耦）。

高可用性：可用性是现代基于云的应用程序的必备功能；Manu必须在组件级别隔离系统故障，并使故障恢复透明。

高性能：查询处理性能是矢量数据库的关键。为了获得良好的性能，需要针对硬件进行广泛优化的实现。此外，框架应仔细设计，以尽量减少查询服务的系统开销。

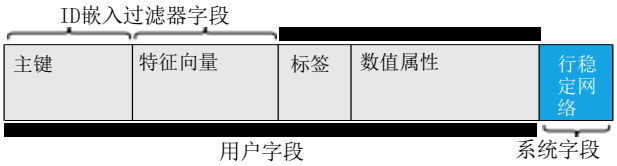

图1：Manu模式的一个例子。

强大的适应性：我们的客户在各种环境中使用矢量数据库，从笔记本电脑上的原型设计到云上的大规模部署。向量数据库应该提供一致的用户体验，并减少跨环境的代码/数据迁移开销。

3手动系统

在本节中，我们将首先介绍Manu的基本概念。接下来，我们介绍了系统设计，包括整个系统架构、日志主干，以及Manu如何进行向量搜索和构建向量搜索索引。

3.1方案、集合、碎片和分段

模式：Manu的基本数据类型是向量、字符串、布尔值、整数和浮点数。图1中给出了一个模式示例。假设每个实体由五个字段组成，并对应于一个电子商务平台上的一个产品。主键是实体的ID。它可以是一个整数或一个字符串。如果用户未指定此字段，系统将自动为每个实体添加一个整数主键。特征向量是乘积的嵌入。标签是产品的类别，如食品、书籍和布。“数值”属性是与产品关联的浮点数或整数，例如价格、重量或生产日期。Manu支持每个实体中的多个标签和数字属性。请注意，这些字段将用于过滤，而不是连接或聚合。逻辑序列号（LSN）是一个对用户隐藏的系统字段。

集合：集合是一组类似于关系数据库中的表的概念的实体。例如，一个集合可以包含一个电子商务平台的所有产品。关键的区别在于集合彼此之间没有关系；因此，不支持关系代数，比如连接操作。

Shard：与插入/删除通道的对应。在插入/删除过程中，实体会根据它们的主键被散列成多个碎片。Manu的数据放置单元是段状的，而不是碎片状的。[1]

分段：从每个碎片中的实体被组织成分段。一个线段可以处于增长状态，或处于密封状态。密封的段是只读的，而不断增长的段可以接受新的实体。当一个增长的段达到预定义大小（默认设置为512MB）或者一段时间没有插入（例如10秒）时，它将切换到密封状态。就像某些部分一样

[1]使用分段进行数据放置比碎片更灵活，因为碎片的数量是静态的，而分段的数量会随着收集的体积的增加而增加。

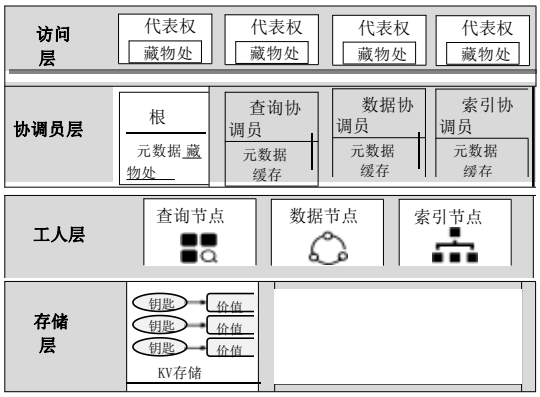

图2：Manu的体系结构。

小(e。g.，当插入到达率低时)，Manu将小段合并为较大段，以提高搜索效率。

3.2系统架构

Manu采用了一种面向服务的设计[64]来实现系统组件之间的细粒度解耦。如图2所示，从上到下，Manu有四个层，即访问层、协调器层、工作层和存储层。

访问层由作为用户端点的无状态代理组成。它们并行地工作，以接收来自客户端的请求，将请求分发到相应的处理组件，并在返回到客户端之前聚合部分搜索结果。此外，代理缓存元数据的副本，以验证搜索请求的合法性（例如，要搜索的集合是否存在）。搜索请求验证是轻量级的，将其移动到代理有两个关键的好处。首先，会提前拒绝验证失败的请求，从而降低了其他系统组件的负载。其次，它减少了请求的路由跳数，从而缩短了请求处理延迟。

协调器层管理系统状态，维护集合的元数据，并协调处理任务的系统组件。有四个协调员，每个人负责不同的任务。根协调器处理数据定义请求，例如创建/删除集合，并维护集合的元信息。数据协调器记录有关集合的详细信息(e。g.，存储上的路由)，并协调数据节点将数据更新请求转换为绑定日志[4]。查询协调器管理查询节点的状态，并调整段（以及相关索引）的分配，以查询节点进行负载平衡。索引协调器维护索引的元信息（例如，索引类型和存储路由），并协调索引构建任务中的索引节点。为了保证可靠性，一个协调器可以具有多个实例（例如，一个主备份和两个备份）。由于向量数据库通常没有关系数据库所具有的跨表操作，因此不同的协调器实例可以提供不同的集合以实现吞吐量。

工作层执行实际的计算任务。工作节点是无状态的——它们获取数据的只读副本来执行任务，并且不需要相互协调。这确保

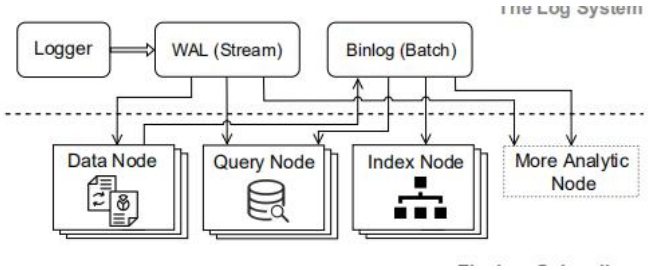

图3：Manu的日志系统的概述。

计算密集型（因此是昂贵的）工作节点可以很容易地按需缩放。我们使用不同的工作节点用于不同的任务，即查询节点用于查询处理，索引节点用于索引构建，数据节点用于日志归档。由于不同计算任务的工作负载随着时间和跨应用程序的不同而显著不同，因此每种工作类型都可以独立地进行扩展。由于不同的计算任务有不同的QoS需求，该设计也实现了资源隔离。

存储层持久化系统状态、元数据、集合和相关索引。Manu使用etcd[7]（一个键值存储）来托管协调器的系统状态和元数据，因为etcd提供了故障恢复的高可用性。更新元数据后，将更新后的数据首先写入etcd，然后同步到协调器。由于其他数据（例如，binlog、数据、索引）的量很大，Manu使用AWSS3[13]（一种对象存储）进行持久性。许多其他对象存储系统的API都与AWS S3兼容。这允许Manu在必要时轻松地交换存储引擎。目前，支持AWS S3、MinIO [8]和Linux文件系统等存储引擎。请注意，对象存储带来的高延迟并不是性能瓶颈，因为工作节点在内存中的只读数据副本上执行计算任务。

3.3日志主干

日志系统是Manu的主干，它连接着解耦的系统组件。如图3所示，Manu将预写日志（WAL）和binlog作为骨干服务公开。WAL是系统日志的增量部分，而双日志是基本部分；它们在延迟、容量和成本上相互补充。记录器是将数据发布到WAL上的入口点。数据节点订阅了WAL，并将基于行的WAL转换为基于列的绑定日志。所有只读组件，如索引节点和查询节点，都是日志服务的独立订户，以保持其最新状态。这种体系结构完全解耦了写和读组件，从而允许组件（例如，WAL、binlog、数据节点、索引节点和查询节点）独立地扩展。

Manu记录所有更改系统状态改为日志的请求，包括数据定义请求（例如，创建/删除收集）、数据操作请求(e。g.，插入/删除向量）和系统协调消息（例如，加载/转储集合到内存）。请注意，向量搜索请求不会被写入日志，因为它们是只读操作，并且不会更改系统

状态我们使用逻辑日志而不是物理日志，因为逻辑日志专注于事件记录，而不是描述对物理数据页面的修改。这允许订阅者根据他们的功能以不同的方式使用日志数据。

图4说明了日志系统的详细架构。为了清晰起见，我们只说明与插入请求相关的部分。日志记录器被组织在一个哈希环中，每个日志记录器基于一致的哈希值来处理哈希环中的一个或多个逻辑桶。每个碎片对应于哈希环中的一个逻辑桶和一个WAL通道。插入请求中的每个实体都根据它们的ID散列到一个碎片（因此是通道）。当一个记录器收到一个请求时，它将首先验证请求的可读性，通过咨询中央时间服务LSN（TSO）为被记录的实体分配一个LSN，确定该实体应该去的段，并将该实体写入WAL。记录器还写入新实体ID的映射，将ID分割成本地LSM树，并定期将LSM树的增量部分刷新到对象存储，从而使用RocksDB的SSTable格式保持实体到分段映射。每个记录器通过查阅对象存储中的SSTable来缓存段映射（例如，用于检查要删除的实体是否存在）。

WAL是基于行的，并以流媒体的方式读取，用于低延迟和细粒度的日志发布/子版本。它是通过基于云的消息队列实现的，如卡fka或脉冲星。我们为WAL使用多个逻辑通道，以防止不同类型的请求相互干扰，从而实现高吞吐量。数据定义请求和系统协调消息使用它们自己的通道，而数据操作请求跨多个通道进行散列，以增加吞吐量。

数据节点订阅WAL，并将基于行的WAL转换为基于列的绑定日志。具体来说，来自同一字段的值(e。g.，属性和向量)以列格式一起存储在binlog文件中。binlog基于列的特性使得它适合于批量读取每个字段的值，从而提高了存储和IO效率。索引节点就是这种效率的一个例子。索引节点只从构建索引的绑定日志中读取所需的字段（例如，属性或向量），因此没有读取放大。

系统协调：组件间的消息也通过日志传递，例如，数据节点宣布段何时被写入存储，索引节点宣布何时建立索引。这是因为日志系统为广播系统事件提供了一个简单而可靠的机制。此外，日志系统的时间语义为协调消息提供了一个确定性的顺序。例如，当一个集合应该从内存中释放出来时，查询协调器会发布该请求以进行记录，并且不需要确认查询节点是否接收到该消息或处理查询节点失败。查询节点独立地订阅集合的日志和异步发布段。

3.4可调谐一致性

我们采用了一个delta一致性模型来实现灵活的性能一致性权衡，这保证了搜索查询所看到的数据的有限不变性。具体来说，查询看到的数据对于最后一个时间单位的时间可能是过时的

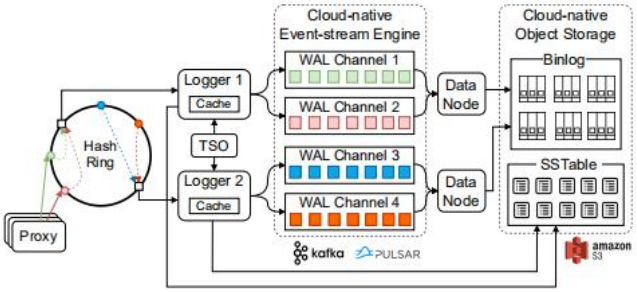

图4：Manu日志系统的详细结构。表1：Manu中的主要指标

矢量量化	PQ，OPQ，RQ，SQ
反向索引	IVF-平，IVF-PQ，IVF-SQ，IVF-HNSW，IMI
接近图	HNSW，NSG，NGT
数值属性	B树，排序列表

数据更新，其中delta是在虚拟时间中给出的一个由用户指定的“持久性容忍度”。

在实践中，用户更喜欢将时间容忍度定义为物理时间，例如，10秒。Manu通过使分配给每个请求的LSN非常接近物理时间来实现这一点。Manu在TSO中使用一个混合逻辑时钟来生成时间戳。每个时间戳都有两个组件：跟踪物理时间的物理组件和跟踪事件顺序的逻辑组件。需要该逻辑组件，因为多个事件可能发生在同一物理时间单元中。由于时间戳被用作请求的LSN，因此物理组件的值表示Manu接收到请求时的物理时间。

对于一个日志订阅者，e。g.，一个查询节点，要运行增量一致性模型，它需要知道三件事：(1)用户指定的持久性容忍度T，(2)最后一次数据更新的时间，以及(3)搜索请求的问题时间。为了让每个日志订阅者都知道(2)，我们引入了一个时间标记机制。被称为时间标记（类似于ApacheFlink[25]中的水印）定期插入到每个日志通道（例如，WAL通道）中，以表示数据同步的进展。表示用户使用的最新时间为L_S和查询的问题时间为利比里亚如果利比里亚- L_S<T不满足，查询节点将等待下一次计时后再执行查询。

注意，强一致性和最终一致性是增量一致性的两种特殊情况，其中增量分别等于0和无穷大。据我们所知，我们的工作是第一次支持向量数据库中的delta一致性。

.53索引楼

通过蛮力在大的集合中搜索类似的向量，即扫描整个数据集，通常会产生不可接受的长延迟。已经提出了许多索引来加速向量搜索，并且Manu自动构建用户指定的索引。表1总结了Manu目前支持的索引，我们正在根据最新的索引算法不断添加新的索引。这些索引的属性和用例都有所不同。向量量化（VQ）[33,44]方法压缩向量

以减少内存占用和向量距离/相似度计算的成本。例如，标量量化（SQ）[90]将向量的每个维度（数据类型通常为int32和浮动）映射到单个字节。反向索引将[68]向量分组到集群中，并且只扫描最有前途的集群以进行查询。接近图[32,41,60]将相似的向量连接成一个图，并以高内存消耗[53]为代价实现了高精度和低延迟。除了向量索引外，Manu还支持在实体的属性字段上的索引，以加速基于属性的过滤。

在Manu中有两种索引构建场景，即批处理索引和流索引。当用户为整个集合构建一个索引时（例如，当使用一个新的嵌入模型更新所有向量时），就会发生批处理索引。在这种情况下，索引协调器从数据协调器获取集合中所有段的路径，并指示索引节点为每个段构建索引。当用户不断地插入新的实体，并且动态地异步地构建索引而不停止搜索服务时，就会发生流索引。具体来说，在一个段累积了足够数量的向量后，其驻留数据节点密封该段并将其作为双日志写入对象存储。

数据协调器然后通知索引协调器，这是什么

指示一个索引节点为该段构建索引。index节点只加载所需的列(e。g., 向量或属性)的段从对象存储的索引构建，以避免读取放大。对于实体删除，Manu使用位图记录被删除的向量，并在删除了足够数量的实体时为一个段重建索引。在批处理和流索引场景中，在为段构建所需的索引后，索引节点将其保存在对象存储中，并将路径发送给索引协调器，该协调器通知查询协调器，以便查询节点可以加载处理查询的索引。

索引协调器还将监视索引节点的状态

并关闭空闲的索引节点，以节省成本。由于向量索引通常具有亚线性的搜索复杂度w.r.t.向量的数量，搜索一个大段比几个小段便宜，Manu在适当的时候在多个段上建立联合索引。

3.6向量搜索

Manu支持经典的向量搜索、属性过滤和多向量搜索。对于经典的向量搜索，距离/相似度函数可以是欧几里得距离、内积或角距离。当搜索类似于受某些属性约束影响的查询的向量时，属性过滤很有用。例如，一个电子商务平台可能希望找到能让客户感兴趣且成本低于100$的产品。Manu支持三种属性过滤策略，并使用基于成本的模型为每个分段选择最合适的策略。当一个实体由多个向量编码时，需要多向量搜索，例如，一个乘积可以通过其图像的嵌入和其文本描述的嵌入来描述。在这种情况下，实体之间的相似函数被定义为构成向量上的相似函数的组成。Manu支持两种多向量搜索策略，并根据该方法选择一种进行使用

实体相似性函数。有关Manu如何处理属性过滤和多向量搜索的更多细节，请访问感兴趣的读者

可以参考Milvus [80]。

对于向量搜索，Manu将一个集合划分为段，并将这些段分布在查询节点之间，以便并行执行。[2]代理通过查询查询协调器来缓存查询节点上段的分布副本，并向保存搜索集合的段的查询节点发送查询请求。查询节点在其局部段上执行向量搜索，而无需使用两阶段的减少过程进行协调。对于top-k向量搜索请求，查询节点搜索它们的局部段以获得分段方向的top-k结果。这些结果被每个查询节点合并，以形成节点级的top-k结果。然后，通过全局top-k结果的代理聚合节点级的top-k结果，并返回到应用程序。为了处理向量的删除，查询节点使用位图记录每个段中被删除的向量，并从分段的搜索结果中过滤被删除的向量。用户可以将Manu配置为批处理搜索请求，以提高效率。在这种情况下，如果尚未返回以前批的结果，代理会组织缓存搜索请求。在缓存中，相同类型的请求（即针对相同的集合并使用相同的相似性函数）被组织成一批，并由Manu一起处理。Manu还允许维护一个集合的多个热副本，以提供可用性和吞吐量的查询。

查询节点从三个来源获取数据，即WAL、索引文件和binlog。对于不断增长的数据段中的数据，查询节点订阅WAL，并使用暴力扫描进行搜索，以便可以在短时间的延迟内搜索更新。段大小的一个困境是，一旦建立了索引，更大的尺寸会产生更好的搜索效率，但对不断增长的段进行蛮力扫描也更昂贵。为了解决这个问题，我们将每个段划分为切片（每个切片默认包含10,000个向量）。新数据按顺序插入到切片中，在切片满后，为其构建一个轻量级的临时索引（例如，IVF-FLAT）。根据经验，我们观察到，临时指数使搜索不断增长的节段的速度提高了10倍。当段从增长状态变为密封状态时，其索引将由索引节点构建，然后存储在对象存储中。之后，将通知查询节点加载索引并替换临时索引。

当查询节点之间的段的分布发生变化时，查询节点访问数据的双日志，这可能发生在缩放、负载平衡、查询节点故障和恢复过程中。具体来说，查询协调器管理段分布，并监控查询节点的活性和工作负载，以协调故障恢复和缩放。在故障恢复时，故障查询节点处理的段及其相应索引（如果存在）被加载到正常的查询节点。[3]在缩小的情况下，一旦其他查询节点从对象存储中加载其处理的段的索引，就可以删除查询节点。当进行扩展时，查询协调器会分配一些

[2]Manu将所有数据加载到查询节点，因为不同的查询可能访问数据的不同部分，并且对于低延迟需要一个热计算侧缓存。这与解耦计算和存储的一般云dbms（例如，雪花[28]）不同，后者只根据请求获取所需的数据来计算端。

[3]由失败的查询节点订阅的WAL通道也被分配给健康的通道。

表2：基于python的PyManu API的主要命令

方法	描述
集合（名称、架构）	创建名称为str和模式模式的集合
收集插入（vec）	将矢量vec插入到集合中
收集删除（expr）	从集合中删除满足布尔表达式expr的向量
收集创建_ 索引（字段、参数）	使用参数参数在向量的字段上创建索引
收集搜索(vec、参数）	向量搜索具有参数参数的vec
收集查询（vec、参数、expr）	向量搜索vec与布尔表达式expr作为过滤器

连接到新添加的节点的数据段。新的查询节点可以在加载分配的段后加入，现有的查询节点可以释放不再由它们处理的段。查询协调器还通过迁移段来平衡查询节点的工作负载（和内存消耗）。注意，Manu不确保段重新分配是原子的，一个段可以驻留在多个查询节点上。这不会影响正确性，因为代理删除查询的重复结果向量。

4个特征突出显示

在这部分中，我们将介绍Manu在可用性和性能方面的几个关键特性。

4.1个云是本地的和自适应的

Manu的主要设计目标是成为一个云原生向量数据库，以便它很好地适合基于云的数据管道。为此，Manu将系统功能解耦到整体设计中的存储、协调器和工作人员中。对于存储，Manu使用事务KV用于元数据，使用消息队列用于日志，使用对象KV用于数据，这些都是主要云供应商提供的通用存储服务，因此易于部署。对于管理系统功能的协调器，Manu使用标准的一个主备份配置和两个热备份配置，以实现高可用性。对于工人来说，Manu可以解耦向量搜索、日志归档和索引构建任务，以实现组件级缩放，这是一个适用于基于云的按需资源供应的模型。日志主干允许系统组件通过自己的方式写入/读取日志进行交互。这使得系统组件能够独立地发展，并使新组件的添加更加容易。日志主干还在系统中提供了一致的时间语义，这对于确定性的执行和故障恢复至关重要。我们的客户在其应用程序的整个生命周期中都在使用向量数据库。例如，一个应用程序通常从数据科学家在他们的个人电脑上进行概念验证（PoC）开始。然后，将其迁移到专用的集群中进行测试，并最终部署到云上。因此，为了降低迁移成本，我们的客户希望向量数据库能够适应不同的部署场景，同时提供一组一致的api。为此，Manu为系统组件定义了统一的接口，但为不同的平台提供了不同的调用方法和实现。例如，在云、本地集群和个人计算机上，Manu分别使用云服务api、远程过程调用（RPC）和直接函数调用来调用系统功能。对象KV可以是本地文件系统(e。g.，MinIO [8])在个人电脑上，和S3在AWS上。因此，马努应用程序可以在不同的部署场景中很少或不更改迁移。

.24良好的可用性

数据管道以简单的方式与Manu交互：向量集合、向量数据的更新和搜索请求被提供给Manu，Manu返回每个搜索请求的搜索结果的标识符，这些标识符可用于检索对象(例如。图像，广告，电影)在其他系统中。由于不同的用户采用不同的编程语言和开发环境，Manu提供了流行语言的api，包括Python、Java、Go、C++，以及RESTfulapi。作为一个例子，我们在表2中显示了基于python的PyManu API的关键命令，该命令使用了对象-关系映射（ORM）模型，并且大多数命令都与集合类相关。如表2所示，PyManu允许用户管理集合和索引、更新集合，并进行向量搜索。搜索命令用于基于相似度的向量搜索，而查询命令主要用于属性过滤。我们展示了一个通过指定参数来进行top-k向量搜索的例子。

```
query_param = {
    "vec": [[0.6, 0.3, ..., 0.8]],
    "field": "vector",
    "param": {"metric_type": "Euclidean"},
    "limit": 2,
    "expr": "product_count > 0",
}
res = collection.search(**query_param)
```

在上述示例中，搜索请求提供了一个高维向量[0]。6, 0.3, ...,0.8]作为查询和搜索集合的特征向量字段。相似度函数为欧氏距离，目标为集合中的前2个相似向量(i。e., 与限制=为2)。

为了便于系统管理，Manu提供了一个名为Attu的GUI工具，其屏幕截图如图5所示。在系统视图中，用户可以观察整个系统状态，包括每秒处理的查询（QPS）、平均查询延迟和屏幕顶部的内存消耗。通过单击一个特定的服务（例如，数据服务），用户可以在侧面查看该服务的工作节点的详细信息。我们还允许用户通过鼠标点击来添加和删除工作节点。在集合视图中，用户可以检查系统中的集合、从内存中加载/转储集合、删除/导入集合、检查为集合构建的索引以及构建新的索引。在矢量搜索视图中，用户可以检查每个集合上的搜索流量和性能，并配置

图5：Aanu的GUI工具Attu的屏幕截图。

要用于每个集合的索引和搜索参数。矢量搜索视图还允许对功能测试发出查询。

对于向量搜索，使用不同的参数作为索引(e。g.，对于HNSW[60]，邻居大小M和队列大小L)会在成本、准确性和性能之间产生不同的权衡。然而，即使是专家也发现很难设置适当的索引参数，因为参数是相互依赖的，它们的影响不同的集合。Manu采用超带贝叶斯优化（BOHB）[31]方法，自动探索良好的索引参数配置。用户提供了一个实用程序函数来给配置进行评分(e。g.，根据搜索召回率、查询吞吐量)，并设置一个预算来限制参数搜索的成本。BOHB从一组初始配置开始，并评估它们的实用程序。然后，利用贝叶斯优化方法根据历史试验生成新的候选配置，利用超带方法将预算分配到配置空间的不同区域。这个想法是为了优先探索接近高公用事业配置的区域，以找到更好的配置。Manu还支持对路径集合的一个子集进行采样，以降低搜索成本。我们仍在改进自动参数搜索模块，并计划将其扩展到搜索系统配置(e。g.，查询节点的数量和类型)。

.34时间旅行

用户通常需要回滚数据库来修复损坏的数据或代码错误。Manu允许用户为数据库恢复指定一个目标物理时间T，并联合使用检查点和日志重放进行回滚。我们用每个段的进度L来标记它，并定期检查一个集合的段映射，该集合包含其所有段的信息（这样的路由，而不是数据）。为了在时间T恢复数据库，我们在T之前读取最近的检查点，在段映射中加载所有段，并从其本地进程L中重放每个段的WAL日志。这种设计减少了存储消耗，因为我们不会为每个检查点编写整个集合。相反，没有更改的段将在检查点之间共享。重播开销也减少了，因为每个段都有自己的进度。用户还可以指定一个过期时间来删除过时的日志和段，以减少存储消耗。

4.4硬件优化

Manu为CPU、GPU和SSD进行了广泛的效率优化实现。有关我们的CPU和GPU优化的更多细节，感兴趣的读者可以参考Milvus [80]。

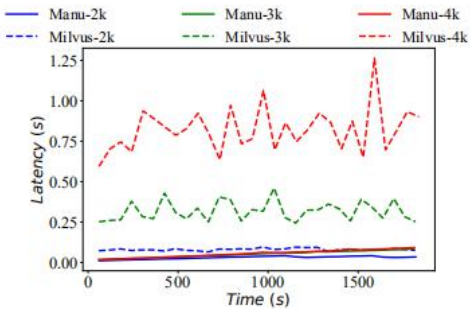

图6： Manu和Milvus的混合工作负载，图例背后的数字(e。g.，1k)表示插入速率。

SSD比dram便宜100倍，其带宽比硬盘大10倍。因此，Manu支持使用SSD在存储容量有限的廉价查询节点上存储大型向量集合。挑战在于SSD带宽仍然比dram小得多，这可能导致低查询处理吞吐量，因此需要仔细设计存储布局和索引结构。由于SSD读取是用4kb的块进行的（即，读取小于4kb与读取4kb的成本相同），Manu将向量组织成大小接近但小于4kb的桶。[4]这是通过对向量进行分层的k-均值和控制簇的大小来实现的。每个桶存储在SSD上的4kb对齐块上，以实现有效读取，并由dram中的kmeans中心表示。这些中心使用现有的向量搜索索引进行组织(e。g.，ivf平面，hnsw)。

使用SSD的矢量搜索分两个阶段进行。首先，我们在dram中搜索集群中心中与查询最相似的中心。然后，从SSD扫描。为了减少从SSD获取的数据量，我们使用标量量化压缩向量，根据我们的试验，这对搜索结果的质量的影响可以忽略不计。另一个问题是，k-means可以将类似于查询的向量放到几个桶中，但是一些桶的中心可能与查询不相似，这导致了低召回率。为了解决这个问题，Manu使用了一种类似于局部敏感哈希[40]中的多个哈希表的策略。分层的k-means被执行多次，每次将一个向量分配给一个桶。这意味着一个向量在SSD中被复制多次，我们在dram中索引所有集群中心进行桶搜索。Manu的SSD解决方案在神经元2021[3]上赢得了十亿规模近似最近邻搜索挑战的轨道2（使用SSD搜索）。测试结果表明，在相同的查询处理吞吐量下，Manu的解决方案将竞争基线的召回率提高了高达60%。[5]我们注意到另一项工作对基于ssd的向量搜索[26]采用了类似的设计。

5个用例和评估

在介绍Manu的用例之前，我们首先将Manu与我们之前的向量数据库Milvus进行了比较。Milvus采用了一个最终的一致性模型，因此不支持Manu的可调一致性。来展示马努的观点所带来的优势

[4]我们将桶的大小设置为几次(e。g.，4和8)，如果单个向量的大小较大，则为4kb。
[5]有关结果的更多细节，请参考[71]。

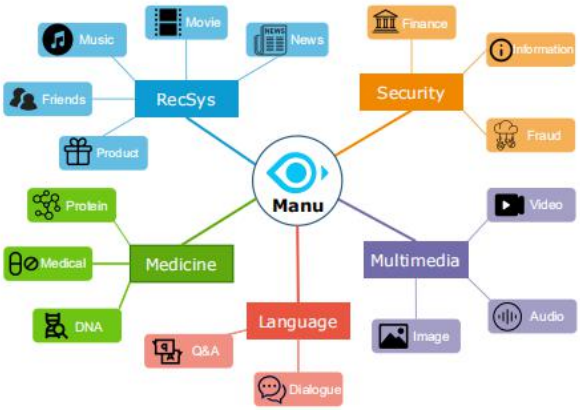

图7：Manu的用例。

通过细粒度的功能分解，我们创建了一个混合的工作负载。具体来说，我们从一个空的集合开始，以固定的速率插入向量（例如，每秒2k个向量），并度量搜索请求随时间变化的延迟。Manu和Milvus都使用6个节点，并正确配置了良好的性能。图6中的结果显示，Milvus的搜索延迟明显长于Manu，特别是当插入率很高时（例如，在3k和4k时）。Milvus有多个读节点，但只有一个写节点，以确保最终的一致性。写入节点负责数据插入和索引构建，从而写入任务和索引构建任务争夺资源。因此，索引构建延迟很长，并且对大量数据使用了强力搜索。相比之下，使用专用的索引节点，Manu快速完成索引构建，因此在整个期间搜索延迟保持很低。

.15用例概述

在图7中，我们将客户划分为5个应用程序域，并简要说明如下。**推荐：电子商务[78]、音乐[76]、news [54]、视频[27]和社交网络[43]平台记录用户内容交互，利用数据将用户和内容映射到ALS [74]和深度学习[49]等技术嵌入向量。**寻找用户感兴趣的内容是通过搜索与用户向量具有较大相似性分数（通常是内积）的内容向量来完成的。

多媒体：多媒体内容（如图像、视频和音频）越来越流行，从大型语料库中搜索多媒体内容在网络上很常见。一般的做法是使用CNN [48]和RNN [86]等工具将用户查询和语料库内容嵌入到向量中。多媒体内容的搜索是通过寻找与用户查询相似的向量来进行的。语言：自动任务应答和机器对话最近随着Siri [14]和小冰[20]等产品引起了广泛关注，对文本内容的搜索是一种普遍需求。使用诸如Word2Vec [62]和BERT [30]等模型，语言序列被嵌入到向量中，从而检索语言

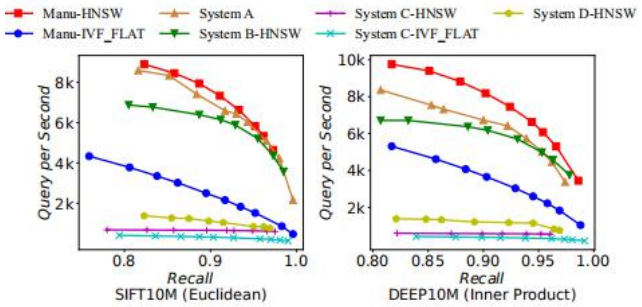

图8：召回vs。整个比较。

内容可以归结为查找类似的内容向量
用户查询。

安全性：阻止垃圾邮件和扫描病毒对安全很重要。常见的做法是使用散列[58]或定制算法[38]将垃圾邮件和病毒映射到向量中。之后，可以通过在语料库中找到最相似的候选邮件来检查可疑的垃圾邮件和病毒，以便进一步检查。

医学：许多医学应用程序寻找某些化学结构和基因序列，用于药物发现或健康风险识别。利用GNN [67]和LSTM [83]等工具，可以将化学结构和基因序列嵌入到向量中，并将其搜索任务转换到向量搜索中。

成熟的向量数据库是必要的，因为除了向量搜索之外，它们还需要更复杂的功能支持。具体来说，由于向量数据集很大，而且应用程序对吞吐量有很高的要求，因此它们需要具有多个节点的分布式计算来实现可伸缩性。当新的用户/内容出现、用户行为改变或嵌入模型被更新时，向量也会不断更新。由于这些应用程序大多数服务于最终用户，因此它们需要高可用性和持久性。我们的一些客户已经在他们的生产环境中部署了Manu，他们发现Manu在可用性、性能、弹性和适应性方面都令人满意。在下面的内容中，我们模拟了客户的一些典型应用场景，以演示Manu的优势。

5.2示例用例

由于业务安全问题，客户的姓名是匿名的。在实验中，我们使用了两个广泛用于向量搜索研究的数据集，i。e.，SIFT[5]（128个dim向量）和DEEP[2]（96个dim向量），并提取所需大小的子数据集。默认情况下，我们使用两个查询节点，一个数据节点和一个索引节点。每个工作节点都是一个运行在AmazonLinuxAMI5.4.129版上的EC2 m5.4xlarge实例。对于指标，我们采用IVF-FLAT [45]和HNSW [60]在实践中广泛应用。当比较Manu与其他系统时，我们总是确保系统使用相同的资源并正确配置。由于时间和费用的限制，我们只能与实验子集中的一些向量数据库进行比较。我们为每个查询搜索最相似的前50个向量，并确保如果没有明确报告召回率，平均搜索召回率高于0.8。

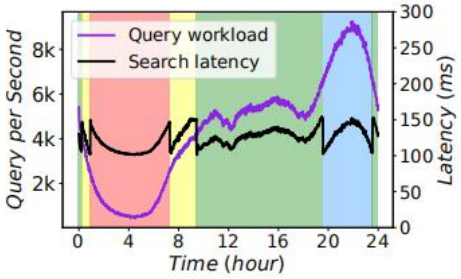

图9：搜索工作负载、查询延迟和Manu长期使用的查询节点数量。不同的颜色表示使用了不同的查询节点数量。

电子商务推荐。A公司是中国领先的在线购物平台，主要销售服装和化妆品。他们使用Manu进行推荐，并根据用户与用户嵌入向量的相似度得分推荐给用户。他们对向量数据库有三个主要要求：(1)高吞吐量，因为他们需要处理许多并发客户的请求；(2)高质量的搜索结果，以获得良好的推荐效果；(3)良好的低成本弹性，因为他们的搜索请求随着时间有很大的波动（高峰在晚上，但在午夜很低，在促销活动很高）。

在图8中，我们比较了Manu与四种流行的开源向量搜索系统（如系统a、系统B、系统C和系统D）在使用单个节点时的召回吞吐量性能。我们使用SIFT的欧氏距离和DEEP的内积来测试不同的相似性函数。由于系统A只支持它自己的基于图的索引，系统B和系统D只支持HNSW索引[60]，所以我们在每个图中只有一条曲线。结果表明，Manu在不同的数据集和相似性函数上始终优于基线。系统C和系统D在同一召回率下实现的查询处理吞吐量明显低于Manu。这是因为系统C用于搜索结果的复杂聚合过程引入了很高的开销，而系统D是一个基于磁盘的解决方案。系统A和系统B的性能明显优于系统C和系统D，但仍低于Manu。我们推测这是因为Manu有更好的优化CPU缓存和SIMD的实现。

为了测试Manu的弹性，我们使用了一个电子商务平台在一天时间内的搜索流量[16]，如图9所示。结果表明，搜索工作负载随时间的推移波动剧烈，峰值远远高于山谷。我们使用SIFT100M作为数据集，欧氏距离作为相似度函数。Manu被配置为将查询节点减少0。当搜索延迟小于100 ms时增加5倍，当搜索延迟超过150 ms时添加查询节点到2倍。图9中的颜色表示Manu使用的查询节点数，说明Manu具有良好的弹性来适应查询工作负载。黑线报告了搜索延迟，并显示Manu可以通过缩放将搜索延迟保持在目标范围内。

视频重复数据删除。B公司是欧洲的一个视频分享网站，用户可以在它上面上传视频并与他人分享。

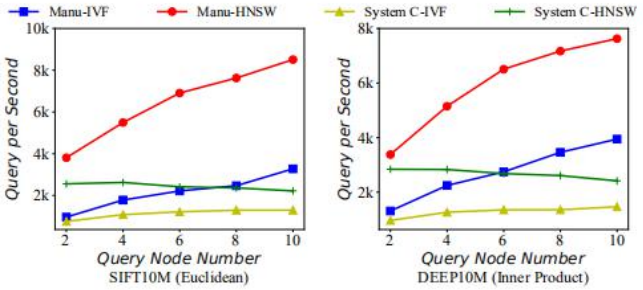

图10：Manuw的可扩展性。r.t.查询节点。

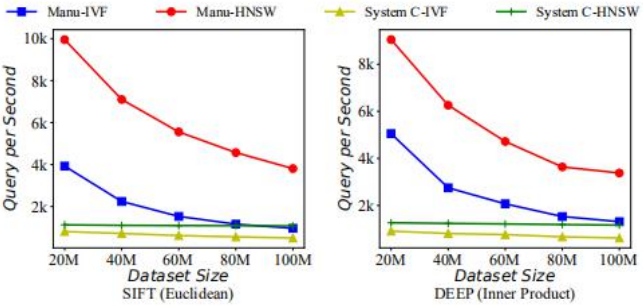

图11：Manu w.r.t.的可扩展性数据卷宗

他们发现有许多重复的视频导致较高的管理成本，因此在存档视频之前进行重复数据删除。他们将视频建模为关键帧，并将每一帧编码成一个向量。他们使用向量搜索来在语料库中找到与新视频最相似的视频，并对入围的视频进行进一步检查，以确定新视频是否为副本。他们还使用矢量搜索来寻找与用户观看的推荐视频相似的视频。它们需要向量DBMS在语料库快速增长时对数据量和计算资源具有良好的可伸缩性。　　在图10和图11中，我们分别测试了Manu在更改查询节点数量和数据集大小时的可伸缩性。结果表明，查询处理吞吐量几乎与查询节点数和数据集大小的倒数呈线性关系。对不同的数据集、索引和相似度函数的观测结果是一致的。这是因为Manu使用段在查询节点之间分配搜索任务。在段大小固定的情况下，当数据量增加时，每个查询节点处理更多的段，而当查询节点的数量增加时，则处理更少的段。请注意，更好的可伸缩性，w.r.t.数据量可以通过配置Manu来使用更大的段来实现数据集大小增加。这是因为相似度搜索索引通常具有亚线性复杂度w.r.t.数据集大小。

病毒扫描。C公司是世界领先的软件安全服务提供商，其主要服务之一是扫描智能手机的病毒。他们有一个病毒库，可以不断地收集新的病毒，并开发专门的算法，将病毒和用户APK映射到载体嵌入。为了进行病毒扫描，他们会在其碱基中发现嵌入了类似于查询APK的病毒，然后将搜索结果与查询APK进行更详细的比较。它们对矢量DBMS有两个要求：(1)流媒体的短延迟

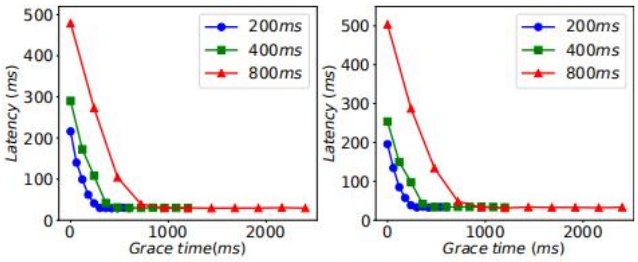

图12：搜索延迟和宽限期时间之间的关系，图例表示时间滴答间隔。

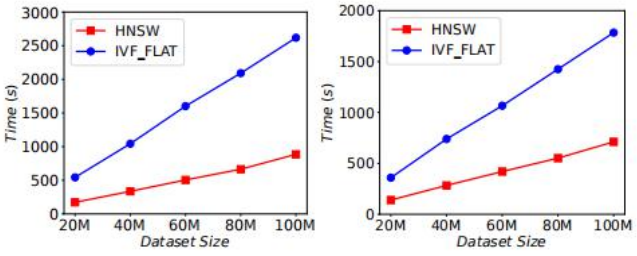

图13：Manuvs的指数建设时间。数据卷宗

随着新的病毒（载体）不断添加到其病毒基中更新，载体搜索需要观察最新的病毒；(2)快速建立索引，因为它们经常调整嵌入算法来解决问题，导致整个数据集的更新，需要重建索引。

在图12中，我们显示了对Manu的搜索请求的平均延迟。回想一下，宽限期时间（即T）意味着搜索请求必须观察在它之前的T时间发生的更新，并且用户可以配置。图例对应于不同的时间间隔，记录器将时间间隔写入WAL。结果表明，随着时间变宽的增加，搜索延迟迅速减小，时间间隔越短，搜索延迟越短。这是因为更长的宽时间，搜索请求可以容忍更长的更新延迟，不太可能等待更新。当时间间隔减少时，每个段可以确认所有更新，因此搜索请求等待的时间更短。在图13中，我们报告了改变数据量时Manu的索引构建时间。结果表明，索引构建的时间随数据量呈线性关系。这是因为Manu为每个段和更大的数据量引线构建索引更多的部分。

6相关工作

向量搜索算法。向量搜索算法有着悠久的研究历史，大多数工作都集中在大规模数据集上的有效近似搜索上。现有的算法大致可以分为树的空间划分（SPTS）、局部敏感散列（LSH）、向量量化（VQ）和接近图（PG）。SPT算法将空间划分为区域，以及使用树状结构快速将搜索结果缩小到某些区域[29,57,63,70,79]。LSH算法设计哈希函数，使相似的向量以高概率散列到相同的桶中，示例包括[34,35,40,42,52,55,56,59,69,89]。VQ算法通过使用一组矢量码本量化向量来压缩向量并加速相似度计算，著名的VQ算法包括[22,33,41,44,90]。PG算法通过将一个向量与数据集中最相似的向量连接起来，形成一个图，并通过图行走[32,60,72,88]进行向量搜索。不同的算法有不同的权衡，e。g.，LSH在索引构建方面很低，但结果质量很差，VQ减少了内存和计算，但也损害了结果质量，PG效率高，但需要大的内存。Manu支持一套全面的搜索算法，这样用户就可以在不同的因素之间进行权衡。

向量数据库。矢量数据管理解决方案已经经历了两个开发阶段。第一阶段的解决方案是库(e。g.，脸书Faiss [45]，微软SPTAG [15]，HNSWlib [60]和Annoy [1])和插件（如ES插件[6]，[11]）用于向量搜索。由于需要成熟的管理功能，因此它们对于当前的应用程序来说是不够的。g.，用于可伸缩性、在线数据更新和故障恢复的分布式执行。两个OLAP数据库系统，AnalyticDB-V [81]和PASE [84]通过添加一个表列来存储它们来支持向量数据，但缺乏针对向量数据的优化。

第二阶段的解决方案是成熟的矢量数据库，如Vearch [51]、Vespa [18]、编织[19]、Vald [17]、Qdrant [12]、松树[10]，以及我们之前的工作Milvus [80]。[6]Vearch使用Faiss作为底层搜索引擎，并采用三层聚合过程来进行分布式搜索。类似地，Vespa将数据分布在节点上，以实现可伸缩性。HNSW算法的一个改进版本用于支持向量数据的在线更新，Vespa还允许在搜索结果和基于学习的推理期间进行属性过滤（例如，用于重新排序）。Weaviate采用了GraphQL接口，并允许存储对象（如文本、图像）、属性和向量。用户可以直接导入向量或自定义嵌入模型来将对象映射到向量，Weaviate可以基于向量搜索结果检索对象。Vald通过将向量数据集划分为分段，并在不停止搜索服务的情况下构建索引，从而支持水平可伸缩性。Qdrant是一个广泛支持属性过滤的单机向量搜索引擎。它允许使用各种数据类型和查询条件（例如，字符串匹配、数值范围、地理位置）进行过滤，并使用定制的优化器来确定过滤策略。请注意，Vespa、Weaviate和Vald只支持接近图索引。

我们可以观察到，这些载体数据库关注的是不同的功能，例如，基于学习的推理、嵌入生成、对象检索和属性过滤。因此，当设计Manu时，我们很容易引入新的功能，将进化性作为一流的优先级。Manu在重要方面也不同于这些载体数据库。首先，Manu的日志主干提供了时间语义，并允许可调谐的一致性。

6Pinecone提供SaaS和闭源代码。因此，我们不知道它的设计细节。

其次，Manu以细粒度分解系统功能，并将其实例化为云服务，以实现性能和故障隔离，因此更适合于云部署。第三，Manu对可用性和性能进行了更全面的优化。g.，支持各种索引、硬件定制的实现和GUI工具。

云本地数据库。最近，许多OLAP数据库被设计成可以在云上运行，例子包括红移[37]、BigQuery [61]、雪花[28]和分析DB[87]。红移是一个数据仓库系统，作为亚马逊网络服务上提供的服务，并采用了一个无共享的架构。它通过添加或删除EC2实例来进行缩放，数据在列的粒度中重新分布。雪花使用共享数据架构，将Amazon数据存储委托给Amazon S3。计算节点是无状态的，可以获取任务数据的只读副本，因此可以很容易地缩放。为了提高效率，我们使用了高性能的本地磁盘来缓存热数据。

Aurora [77]和PolarDB无服务器[50]是两个云原生OLTP数据库。Aurora使用共享磁盘架构，并通过将事务处理向下推到存储引擎来提出“日志就是数据库”原则。它观察到，基于云的平台的瓶颈已经从计算和存储IO转向了网络IO。因此，它只为事务处理的重做日志，并根据LSN处理日志提交事务。PolarDB无服务器采用了一种解聚体系结构，它使用高速RDMA网络来解耦硬件资源(e。g.，计算、内存和存储)作为资源池。

我们的Manu遵循云本地数据库的一般设计原则，在细粒度上解耦系统功能，以实现高弹性、快速演化和故障隔离。然而，我们也考虑了向量数据库的独特设计机会来交换简单的数据模型，以及对性能、成本和灵活性的弱一致性要求。具体来说，不支持复杂的事务，并且利用日志主干来支持可调谐的一致性-性能权衡。此外，向量搜索、索引构建和日志归档任务被进一步解耦，因为它们的工作负载可能有显著的变化。

7个结论和未来的发展方向

本文介绍了Manu作为云原生向量数据库的设计。为了确保Manu适合向量数据应用，我们设置了雄心勃勃的设计目标，包括良好的可进化性、可调的一致性、高弹性、良好的效率等。为了满足这些设计目标，Manu用简单的向量数据模型和应用程序的弱一致性要求来换取性能、成本和灵活性。具体来说，Manu对系统功能进行细粒度的解耦，以进行组件级的扩展和演化，并使用日志主干来连接系统组件，同时提供时间语义和简化组件间的交互。我们还介绍了一些重要的特性，如高级API、GUI工具、硬件优化和复杂的搜索。我们认为Manu还远未达到完美，我们未来的一些发展方向包括：

多路搜索：许多应用程序联合搜索多种类型的内容，如向量和主键、向量和文本。Manu的日志系统允许添加其他搜索引擎内容(e。g.，主键和文本)作为协同处理器，通过订阅日志流。我们将探讨多个搜索引擎如何能够有效地交互，以及如何灵活地协调不同的搜索引擎以满足应用程序需求。

模块化算法：我们认为向量搜索算法可以被提炼为独立的组件，例如，压缩以减少内存和有效的计算，限制计算的一小部分向量的索引，以及对相似的向量进行分组。现有的向量搜索算法只探索了针对不同组件的一些技术组合。我们将提供一个统一的矢量搜索框架，使用户可以根据他们所期望的成本和性能之间的权衡，灵活地结合不同的技术。

分层存储感知索引：当前的向量搜索索引假设了一种单一类型的存储器，例如，GPU内存、主内存或磁盘。我们将探索可以联合利用存储层次结构中的所有设备的索引。例如，大多数应用程序都有一些热门载体（例如，电子商务中的流行产品），这些载体经常被搜索请求访问，可以放置在快速存储中。由于一个查询只访问一部分向量，而一个节点处理许多并发查询，因此存储交换延迟可能会被管道隐藏。

高级硬件： NVM [66]在单位容量上的成本约为DRAM的三分之一，但提供了可比较的读取带宽和可比较的延迟，这使得它成为在存储大型数据集时替换昂贵的DRAM的良好选择。RDMA [24,46]显著降低了节点间的通信延迟，而NVLink [65]直接以比PCIe更大的带宽连接gpu。通过利用这些快速连接，我们将探索联合使用多个设备的索引和搜索算法。我们还与硬件供应商合作，将FPGA和MLU应用于向量搜索和索引构建。

嵌入生成工具箱：为了更好地实现应用程序级别集成，我们计划合并一个面向应用程序的工具箱来生成嵌入向量。这个工具箱除了提供许多可以提供现成使用的预训练模型外，还将包含模型微调，从而允许快速原型化。

确认

Manu是由Zilliz开发的一个为期多年的开源项目。马努岛的开发涉及到其社区中的许多工程师。我们特别要感谢孙炳毅、朱伟达、蔡玉东、莫宜华、西葛、戴一好、龙、蔡张、下刚霞、杨宣、吕彬彬、刘晓云、朱文兴、宗玉芬、曾杰、少月、陈景佳、曾昭、陈嘉莲、闵敏、燕王等社区其他贡献者的贡献。我们也感谢菲利普·哈尔特梅耶对论文的校对，以及提高论文质量的宝贵建议。

参考文献

[1] 2021.大约是最近的邻居，哦，是的。https://github.com/spotify/annoy.
[2] 2021.十亿级相似性搜索的基准。https://research.氰化合物。com/datasets/biganns.

[3] 2021.10亿个规模的近似最近邻搜索挑战。https://big-ann-基准。com.
[4] 2021.binlog.https://hevodata.com/learn/using-mysql-binlog/.
[5] 2021.用于近似最近邻搜索的数据集。http://corpustexmex.irisa.fr/.
[6] 2021.弹性搜索：开源的，分布式的，实时的搜索引擎。https://github.com/elastic/elasticsearch.
[7] 2021.etcd.https://etcd.io/.
[8] 2021.米尼奥。https://min.io/.
[9] 2021.MySQL。https://www.mysql.com/.
[10] 2021.松果酮。https://www.pinecone.io/.
[11] 2021.世界上最先进的开源关系数据库。https://www.postgresql.org/.
[12] 2021.昆德兰特。https://qdrant.tech/.
[13] 2021.S3.https://aws.amazon.com/cn/s3/.
[14] 2021.西里。https://www.apple.com/siri/.
[15] 2021.SPTAG：一个用于快速近似最近邻搜索的库。https://github.com/microsoft/SPTAG.
[16] 2021.淘宝用户行为数据。https://tianchi.aliyun.com/dataset/dataDetail?dataId=649.
[17] 2021.阀门。https://github.com/vdaas/vald.
[18] 2021.黄蜂牌小型摩托车https://vespa.ai/.
[19] 2021.武器。https://github.com/semi-technologies/weaviate.
[20] 2021.小冰。https://en.维基百科。org/wiki/Xiaoice.
[21] Reza·阿克巴里尼亚，埃斯特·帕西蒂，和帕特里克·瓦尔杜里斯。2007.最佳位置Algo顶级查询的条件。在第33届大型数据库国际会议（奥地利维也纳）。VLDB捐赠，495-506。
[22]法比安·安德烈，安妮-玛丽·克马雷克和尼古拉斯·勒·斯库阿内克。2015.藏物处局部性是不够的：高性能的最近邻搜索与产品量化快速扫描。过程VLDB恩多。9，4，288－299.
[23] Artem Babenko和Victor S。兰皮茨基。2015.反向多索引。电器和电子工程师学会反式模式肛门。马赫数。知识37，6（2015），1247－ 1260.https://doi.org/10.1109/.2TPAMI014.2361319
[24]魏曹，张英强，杨新村，李飞飞，王胜，清大胡，荀-程涛、陈宗志、刘振军、景方等。2021.PolarDB无服务器：一个用于分解数据中心的云本机数据库。在2021年数据管理国际会议的论文集上。2477－2489.
[25]巴黎·卡本，阿斯特里奥斯·卡特西福迪莫斯，斯蒂芬·埃文，沃尔克·马克尔，赛义夫·哈里迪，和科斯塔斯Tzoumas。2015.Apache flink：单个引擎中的流和批处理。IEEE计算机学会数据工程技术委员会公告36,4（2015）。
陈齐[26]、赵兵、王海东、李明勤、刘传杰、郑勇、毛阳和王京东。2021.斯潘：高效的十亿个规模的近似最近的社区搜索。神经信息处理系统的研究进展34（2021年）。
[27]，保罗·卡温顿，杰伊·亚当斯和埃姆雷·萨尔金。2016.深度神经网络youtube建议。在第十届ACM关于推荐系统的会议的会议记录中。191－ 198.
[28]贝诺瓦·达格维尔，克鲁阿内斯，祖科夫斯基，瓦迪姆·安东诺夫，阿丁艾文斯，乔恩博克，乔纳森克莱博，丹尼尔恩戈瓦托夫，马丁亨切尔，黄建生，艾利森W。李，阿什什·莫蒂瓦拉，阿卜杜勒·Q。穆尼尔、史蒂文·佩利、彼得·波维内克、格雷格·拉恩、特里安塔菲利斯和菲利普·昂特布伦纳。2016.雪花弹性数据仓库。2016国际数据管理会议，2016，美国旧金山，2016年6月26日-7月1日。ACM，215-226。
[29] Sanjoy·达斯古普塔和Yoav·弗罗因德。2008.随机投影树和低维流形。在第四十届ACM计算理论研讨会论文集。537－546.
[30]雅各布·德夫林，张明伟，李和克里斯汀娜·图塔诺娃。2019.对语言理解的深度双向转换器的预培训。计算语言学协会北美分会2019年会议记录：人类语言技术，NAACL-HLT 2019，明尼阿波利斯，美国，2019年6月27日，第一卷（长和短论文）。计算语言学协会，4171-4186。https://doi.org/10.18653/v1/n19-1423
[31] Stefan·福克纳，亚伦·克莱因，和弗兰克·哈特。2017.结合超带和贝叶斯优化在NIPS 2017贝叶斯优化研讨会上（2017年12月）上。
[32]从福、赵翔、王长旭、邓蔡。2019.快速近似使用导航扩展图进行最近邻搜索。《VLDB捐赠程序程序》12,5（2019），461-474。
葛铁正[33]、何开明、齐法可、孙吉安。2013.优化产品近似最近邻搜索的量化。在IEEE关于计算机视觉和模式识别的会议的论文集上。2946－2953.
[34]阿里斯蒂迪斯，彼得因代克，拉杰夫莫特瓦尼等。1999.相似性搜索高尺寸通过哈希。在Vldb，Vol。99.518－529.
龚[35]、王仪、小原光典、徐军。2020.可索引的用于近似最近邻搜索的距离估计代码。VLDB捐赠基金的诉讼程序13,9（2020年）。
郭[36]瑞琪、库马尔、乔罗曼斯基、辛查。2016.全基于实体化的快速内部产品搜索。在人工智能和统计学方面。PMLR，482－490.
[37] Anurag古普塔，迪帕克·阿加瓦尔，德里克·谭，雅库布·库莱萨，拉胡尔·帕塔克，Ste-法诺·斯蒂凡妮和维迪亚·斯里尼瓦桑。2015.亚马逊的红移和更简单的数据仓库的情况。2015 ACM SIGMOD国际数据管理会议记录，澳大利亚维多利亚州，2015年5月31-6月4日。ACM，1917-1923年。
[38]，迈克尔·赫索维奇，米查尔·雅各维，约埃尔·马雷克，丹·佩勒格，梅纳赫姆·施塔尔-海姆和西格利特·乌尔。1998.鲨鱼搜索算法。一个应用程序：量身定制的网站映射。计算机网络和ISDN系统30、1-7（1998），317-326。
[39]黄东旭、刘齐、邱崔、朱和芳、马小雨、徐飞、李沈、刘唐、周宇兴、黄梦龙等。2020.TiDB：一个基于筏子的HTAP数据库。VLDB捐赠基金的诉讼程序13,12（2020），3072-3084。
[40]，Piotr，Indyk和Rajeev·莫特瓦尼。1998.近似的最近的邻居：朝向消除了对维度的诅咒。在第三十届ACM年度计算理论研讨会的论文集上。604－613.
[41]岩崎弘和宫崎骏大助。2018.索引的优化基于高维数据中接近搜索的k-最近邻图。arXiv预印本arXiv：1810（2018）。.07355
[42] Omid Jafari，帕斯·纳加卡尔和乔纳森·蒙塔诺。2020.mmLSH：Prac-多媒体数据上近最近邻查询的高效处理技术。在关于相似性搜索和应用的国际会议上。施普林格，47-61岁。
[43] Mohsen Jamali和马丁·埃斯特。2010.一种矩阵分解技术在社交网络中进行推荐的信任传播。在第四届ACM关于推荐系统的会议的会议记录中。135－ 142.
[44] Herve Jegou，马蒂伊斯·Douze，和科迪莉亚·施密德。2010.产品量化对于最近的邻居搜索。在模式分析和机器上的IEEE事务情报33、1（2010），117-128。
[45]杰夫·约翰逊，马蒂伊斯·杜兹和赫尔夫·杰古。2019.使用gpus进行数十亿个规模的相似性搜索。IEEE大数据交易（2019年）。
[46] Anuj Kalia，迈克尔·卡明斯基和大卫·安德森。2014.使用RDMA有效地为关键价值的服务。在2014年ACM会议会议记录中数据通信专业组295－306。
[47]蒂莫西国王。2019.你的80%的数据在5年内都是非结构化的年https://solutionsreview.com/data-management/80-percent-of-your-datawill-be-unstructured-in-five-years/.
[48] Yann LeCun，约舒亚·本吉奥，等人。1995.图像的卷积网络，语言和时间序列。脑理论和神经网络手册3361,10（1995），1995年。
[49] Yann LeCun，约舒亚·本吉奥和杰弗里·辛顿。2015.深度学习自然521，7553（2015），436－444.
[50]菲菲李。2019.阿里巴巴的云本地数据库系统：机遇和机遇挑战。过程VLDB恩多。12，12（2019），2263－2272.
李[51]杰、刘海峰、桂创华、陈建宇、倪镇远、王宁、和袁晨。2018.一个建立在jd电子商务平台上的实时可视化搜索系统的设计与实现。在第19届国际中间件会议行业的论文集中。9－ 16.
[52]李明杰、张英、孙亿芳、王伟、曾义华、学敏直线的2020.基于学习函数的I/O有效近似最近邻搜索。2020年IEEE第36届国际数据工程会议（ICDE）。IEEE，289-300。
李文、张英、孙亿芳、王伟、李明杰、张文杰、林学敏。2019.在高维数据上的近似最近邻搜索-实验、分析和改进。IEEE《知识与数据工程学报》32,8（2019），1475-1488。
[54]刘家慧，彼得·多兰和伊林Rønby佩德森。2010.基于点击行为的个性化新闻推荐。在第15届智能用户界面国际会议的论文集上。31－40.
刘万奇[55]、王汉臣、张英、王魏、陆秦。2019.I-LSH：在高维空间中的I/O有效的c-近似最近邻搜索。2019年IEEE第35届国际数据工程会议（ICDE）。IEEE，1670-1673。
陆科景[56]和工藤敏一。2020.一个最近邻搜索方案基于二维投影空间。2020年IEEE第36届国际数据工程会议（ICDE）。IEEE，1045-1056。
[57]陆开景、王洪雅、王伟、工藤一。2020.VHP：approxi-通过虚拟超球面划分进行配对最近邻搜索。《VLDB捐赠程序程序》13,9（2020），1443-1455。
[58]罗莱龙，郭德克，马，罗登奇，罗雪山。2018.优化bloom过滤器：挑战、解决方案和比较。IEEE通信调查和教程21,2年（2018年），1912-1949年。
[59]秦吕，威廉约瑟夫森，王哲，摩西查里卡，李凯。2017.在-局部敏感哈希的远程探测：多探针LSH及以上。(2017).
[60] Yu·马尔科夫和德米特里·亚什宁。2018.高效鲁棒近似最近邻搜索使用层次的可导航的小世界图。电器和电子工程师学会

在模式分析和机器智能方面的交易42,4（2018），824-836。
[61]谢尔盖梅尔尼克，安德烈圭巴列夫，静静龙，杰弗里罗默，湿婆湿婆-
阿库玛尔，马特·托尔顿，和西奥·瓦西拉基斯。2010.网络规模数据集的交互式分析。过程VLDB恩多。3，1（2010），330－339。
[62]托马斯米科洛夫，伊利亚苏特斯科弗，陈凯，格雷格S科拉多，和杰夫迪恩。2013.
单词和短语的分布表示及其组合性。在神经信息处理系统的研究进展中。3111－3119.
[63] Marius Muja和大卫·G·Lowe。2014.可扩展的最近邻算法
用于高维数据。在模式分析和机器上的IEEE事务
情报36,11（2014），2227-2240。
[64]迈克尔P帕帕佐格卢和威廉-简范登赫维尔。2006.面向服务
设计和开发方法。国际Web工程与技术杂志2,4（2006），412-442。
[65]、钟信、苏拉、吴文梅、熊晋军。2018.
功率/NVLink多gpu系统的numa感知数据传输测量。在关于高性能计算的国际会议上。蹦跳的人
448－454.
[66]杰，张敏佳、董丽。2020.在异构内存上进行高效的十亿点最近邻搜索。神经信息处理系统的研究进展33（2020年）。
[67]弗朗哥斯卡塞利，马可戈里，阿钟宗，马库斯哈根布奇纳，和
加布里埃尔蒙法迪尼。2008.图的神经网络模型。神经网络上的IEEE交易20、1（2008），61-80。
[68]福尔克·肖勒，休·威廉姆斯，约翰·扬尼斯和贾斯汀·佐贝尔。2002.压缩
用于快速查询评估的反向索引。在第25届ACM SIGIR国际信息检索研究与发展会议论文集上。222－229.
[69]安舒马利斯里瓦斯塔瓦和平。2014.次线性的不对称LSH（ALSH）
时间最大内部产品搜索（MIPS）。神经信息的进步
处理系统27（2014）。
[70] Chanop Silpa-Anan和理查德·哈特利。2008.优化的kd树为快速图像
描述符匹配。2008年IEEE计算机视觉和模式识别会议。IEEE，1-8。
[71]哈沙瓦德汉西哈德里，乔治·威廉姆斯，马丁阿穆勒，马蒂斯杜兹，
巴年科、巴兰丘克、陈琦琦、卢卡斯霍塞尼、拉维斯汉卡尔克里什纳斯瓦米、戈帕尔斯里尼瓦萨等。2022.NeurIPS的21挑战在十亿规模的近似近邻搜索的结果。arXiv预印本arXiv：2205.03763（2022年）。
[72] Suhas，[72]，[72]，拉维斯-
汉卡尔·克里什纳斯瓦米和罗汉·卡德科迪。2019.在单个节点上快速精确的十亿点最近邻搜索。在神经信息处理系统的进展32：2019年神经信息处理系统年会，NeurIPS2019,2019年12月814日，温哥华，BC，加拿大。13748－ 13758.
[73]，丽贝卡·塔夫脱，伊尔凡·谢里夫，安德烈·马泰，内森·范本肖顿，乔丹·刘易斯，托拜厄斯格里格，凯尼米，安迪伍兹，安妮比尔津，拉斐尔波斯，等人。2020.弹性地理分布式sql数据库。在2020年ACM SIGMOD数据管理国际会议的论文集上。1493－1509.
[74] Gabor Takacs和[74]Tikk。2012.交替最小二乘的个人-
了排名。在第六次ACM关于推荐系统的会议的会议记录上。
83－90.
[75] Ashish图修，快乐的森沙尔玛，耆那教，郑绍，查卡，
苏雷什安东尼，刘浩，皮特威科夫，和拉戈瑟姆默蒂。2009.蜂巢：一个建立在地图简化框架之上的仓库解决方案。《VLDB捐赠程序程序》2,2（2009），1626-1629。
[76]亚伦范登奥尔德，桑德迪勒曼，和本杰明施劳文。2013.深
基于内容的音乐推荐。在神经信息处理系统会议（NIPS 2013）上，卷。26.神经信息处理系统基础（NIPS）。
[77]亚历山大维比特斯基，阿努拉格古普塔，德班扬萨哈，穆拉利布拉玛德森，
卡迈勒·古普塔、拉曼·米塔尔、克里希那穆蒂、桑多尔·莫里斯、哈拉蒂什维利和包晓峰。2017.亚马逊Aurora：高吞吐量云本地关系数据库的设计考虑事项。2017 ACM国际数据管理会议，SIGMOD会议2017，芝加哥，IL，美国，2017年5月19日。ACM，1041-1052。
王吉哲、黄培、赵桓、张志博、赵宾强、李迪伦。2018.为阿里巴巴的电子商务推荐提供的数十亿美元规模的商品嵌入。在第24届ACM SIGKDD知识发现和数据挖掘国际会议的论文集上。839－848.
王京东，王奈彦，贵佳，吉安李，曾刚，查宏斌，
华贤盛。2014.用于近似最近邻搜索的三元投影树。IEEE跨。模式肛门。马赫数。知识36，2（2014），388－403.
王建国、易小孟、郭仁东、金、徐鹏、李胜军、习
王小姐玉、郭香洲、李成明、许晓海等。2021.Milvus：一个专门构建的向量数据管理系统。在2021年数据管理国际会议的论文集上。2614－2627.
[81]魏创贤、吴斌、王胜、楼仁杰、詹超群、李菲飞、
和蔡元哲。2020.一个用于结构化和非结构化数据查询融合的混合分析引擎。过程VLDB恩多。13，12（2020），3152－3165.
吴，郭瑞琪，苏雷什，库马尔，丹尼尔
霍尔特曼-赖斯，大卫西姆查，和费利克斯余。2017.用于快速相似度搜索的多尺度量化。神经信息处理系统的研究进展30（2017），5745-5755。
[83]新行健、陈中龙、王浩、杨地燕、王伟健、
王春宇。2015.卷积LSTM网络：一种用于降水临近预报的机器学习方法。在神经信息处理系统的研究进展中。802－810.
[84]，李涛，盖方，洪伟。2020.PASE：PostgreSQL超高
维度近似最近邻搜索扩展。2020年国际数据管理会议论文集，SIGMOD会议2020，在线会议[波特兰，或，美国]，2020年6月19日。ACM，2241-2253。
[85] Matei·扎哈里亚，莫沙拉夫·乔杜里，迈克尔·富兰克林，斯科特·申克，离子
斯托卡等人。2010.Spark：带有工作集的集群计算。HotCloud 10，10-10（2010），95.
[86]沃伊切赫·扎雷姆巴，伊利亚·苏茨克弗和奥里奥尔葡萄酒。2014.复发性神经
网络正则化。arXiv预印本arXiv：1409.2329（2014）。
[87]詹超群，苏茂孟、魏创贤、彭晓强、林梁、盛
王、陈哲、李飞飞、岳潘、方等。2019.分析方法：阿里巴巴云上的实时olap数据库系统。《VLDB捐赠基金论文集》第12期、第12期（2019年），2059-2070年。
赵伟杰，谭树龙，李平。2020.SONG：接近最近的Neigh-
在GPU上搜索。在第36届IEEE数据工程国际会议上，ICDE 2020，达拉斯，德克萨斯州，美国，2020年4月20-24日。IEEE，1033-1044。
郑[89]博龙，赵喜，翁梁良贵，阮元，鸿越，刘杭，
和基督教Jensen。2020.PM-LSH：一个快速和准确的LSH框架
高维近似神经网络搜索。《VLDB捐赠程序程序》13,5（2020），643-655。
[90]周文刚、陆一娟、李后强、齐田。2012.标量量化
大规模图像搜索。第20届ACM国际会议会议记录
在多媒体上。169－ 178.