项目申请书

项目 ID:24f5f0165

支持IVFPQ向量索引算法

申请人:蒋嵩林

最后修改时间: 2024.5.30

目录

1. 申请人介绍

我叫蒋嵩林，目前是芬兰阿尔托大学安全与云计算专业的硕士研究生，即将在今年夏天毕业，秋季计划在本校继续攻读博士学位，研究机器学习系统和数据库有关课题，目前首先准备从优化RAG工作流以及向量召回加速入手。最近正在研究ANN算法以及它们的GPU加速（raft库），因为十分契合我的研究方向，所以非常感兴趣，希望能在暑期做这个项目。

我本科就读于兰州大学，刚进入大学时，在兰州大学开源社区学长们的指引下，我慢慢产生了对开源的热爱和激情。我建立了自己的GitHub账户（@HollowMan6），每当我编写了一些有趣的程序时，我都会为它建立一个存储库，然后发布到GitHub上。我十分欣赏开源精神，并乐于为此奉献，因而我把大部分空闲时间都花在了GitHub上。我现在在GitHub上拥有超过1.5k粉丝和超过550个个人项目的star。同时，我喜欢学习新知识，开源给予了我许多新的知识和能力，这些一点一滴的知识极大地拓宽了我的视野。

在2020年暑假期间，我同时参与了两份在线开源实习，一个是在中国科学院软件研究所开源软件供应链点亮计划——暑期2020的资助下于Emacs（中国）社区，另一个是在阿里巴巴编程之夏的资助下于阿里巴巴集团最热门的开源项目（Star数最多）——Arthas社区。项目期间，我很快从两个社区的新手变成了资深贡献者，并通过详细的计划出色地完成了任务。我还积极参与社区讨论，并且对即使在任务要求之外的问题需求也进行解决，及时响应，落实导师提出的修改建议。来自这两个社区的导师都对我在实习期间所做的工作给予了很高的评价。通过这两次实习，我锻炼了我的团队合作和沟通能力，获得了参与和贡献大项目丰富的经验和技能。在Emacs社区暑期2020导师的评价：https://isrc.iscas.ac.cn/gitlab/summer2020/students/proj-2012153/-/issues/25#note_175301，项目Wiki: https://isrc.iscas.ac.cn/gitlab/summer2020/students/proj-2012153/-/wikis/home

在2021年暑假，我参加了谷歌编程之夏，在OpenSUSE社区从事自定义IBus扩展的研究，并且同时参与了暑期2021的项目，在openEuler社区，使用C++和QT开发了一个ISO镜像刻录程序。此外，通过在2022年暑假再次参加谷歌编程之夏，我有幸加入到麻省理工学院计算机科学与人工智能实验室 (CSAIL)的 MIT App Inventor社区，创建了一个插件，从而允许在 Google Blockly（一个基于块编程的低代码平台）中一次选择、拖动和操作多个块。我的插件解决困扰 Google 工程师十年的问题（自 Blockly 项目启动以来）：https://github.com/google/blockly-samples/issues/267 。在2023 年 Google Blockly 开发者峰会期间我发表了演讲，阐述了我的工作，视频链接：https://www.youtube.com/watch?v=4OFU9D1Y2DI

2022年我还通过参加 Microsoft 强化学习开源节，与微软研究院纽约实验室的真实世界强化学习团队一起开展了为期四个月的研究项目。我的工作是为 Vowpal Wabbit（一个用 C++ 构建的机器学习平台）实现原生的CSV 解析器。我按照 CSV 官方规范实现了解析器，同时还与特定于上下文老虎机的命名空间和多行示例兼容。该项目达到了100%的测试和代码覆盖率，解析速度几乎与他们的原始文本解析器相同。我想提的一点是，我尽力优化解析器，性能实际上比我最初的代码提高了 10 倍！ 我也很自豪，我是所有参与者中唯一一个代码被合并到上游并最终合并到 v9.4.0 的人：https://github.com/VowpalWabbit/vowpal_wabbit/releases/tag/9.4.0

2022年，我还参与了openEuler开源实习，帮助解决共计6个议题（issue），涉及技术包括D-Bus、Glib、制作云原生容器镜像、Rust编程、Chrome开发者工具远程调试协议、测试编写、rpm软件包的打包制作。我在所有实习生中首先达到实习积分150，被评为优秀实习生并获得“开源之星”称号，其中一项成就 mdbook-pdf 现在平均每周下载量超过

100 次 https://github.com/HollowMan6/mdbook-pdf。2022年夏天，我另外还参与了腾讯犀牛鸟开源人才培养计划，在KonaJDK社区研究使用 JDK 生成 SM2 的密钥对，比较算法的性能和安全性。我最终获得了全部荣誉称号（2/4000+）：腾讯开源贡献者证书（当时全球仅发出30+张）、优秀学生获得者、任务奖金获得者。

我还积极参与了 2023 年 Igalia 代码体验计划，为 VR 浏览器 Wolvic（前身是Mozilla Firefox Reality）的开发做出了贡献。在此期间，我参与了已弃用的 Android 方法的重构，增强了用户交互功能，并解决了 一些OpenXR 相关问题。

去年夏天，我完成了在芬兰 CSC — IT 科学中心的实习。 工作内容主要是通过Go和C++语言将GPU使用数据监控后端转移到时间序列数据库TimescaleDB。我还重新设计了指标收集服务，负责将来自各种 GPU 节点（包括 AMD 和 Nvidia）的数据聚合到上述数据库中。值得注意的是，我的工作成果被应用到了CSC—IT科学中心管理着的三台全球知名的超级计算机——Puhti、Mahti和LUMI。尤其是 LUMI，在最新超级计算机 TOP500 排行榜上名列全球第五、欧洲第一。目前我正在CSC编写基于该监控系统的报警服务并以此完成我的硕士毕业论文。

在项目申请期间，我搭建好了开发环境，并尝试在本地电脑编译了OceanBase已有vector_search分支中的代码，测试了相关IVF-Flat算法的功能，并提交了4个PR来修复编译过程中遇到的问题：

- https://github.com/oceanbase/oceanbase/pull/1983
- https://github.com/oceanbase/oceanbase/pull/1984
- https://github.com/oceanbase/oceanbase/pull/1994
- https://github.com/oceanbase/oceanbase/pull/1995

如果我的提案和努力被接受，最终成功地融入上游代码库，我会为自己感到骄傲。

2. 项目详细方案

2.1 背景/算法

IVFPQ 作为融合了倒排索引和乘积量化的向量索引结构，相比起 IVFFlat 有多种优势，包括减少存储空间、加快搜索速度等。本任务是在当前 OceanBase 的向量版本基础上，实现 IVFPQ 索引算法的支持，以提供更丰富的向量索引类型。任务的目标不仅仅是实现 IVFPQ 算法本身，还需要将其应用在 OceanBase 的内核代码中，以达到在数据库层面原生支持 IVFPQ 向量索引的效果。

乘积量化 **(PQ)**：https://www.pinecone.io/learn/series/faiss/product-quantization/

1. 将一个大的高维向量划分为大小相等的块，创建子向量。

2. 确定每个子向量的最近 K 均值质心，将其称为再现或重建值。

3. 将这些再现值替换为代表质心的唯一 ID。

根据IVFPQ算法原理，IVF-PQ算法具体实现可分为以下几种方式：

1. 先用IVF算法把n个点分成几个大类，在每个大类里面，再用聚类局部的PQ算法降维；
2. 先用IVF算法把n个点分成几个大类，在每个大类里面，再用全局唯一的PQ算法降维；

3. 先用IVF算法把n个点分成几个大类，在每个大类里面，计算点与其所在的聚类的中心的差（残差）之后，对所有残差做一次PQ算法；

方案1和方案2的唯一区别就在于PQ算法是聚类局部的（每个聚类都有自己的PQ）还是全局唯一的。第一种方案有一个明显的缺点，即需要执行非常多次的聚类算法，会消耗大量时间。而方案3和其他2种方案相比，区别在于使用残差做PQ，而不是原始向量。这样做，使得在执行PQ算法时，把所有的聚类中心都平移到了原点，使得所有点都聚焦到原点。由于点变得更加聚拢，所以同样分为ksub个聚类时，平均而言每个聚类的区域会更小，做距离近似时，误差会更小。

Faiss（Meta Research推出的用于高效相似性搜索和密集向量聚类的库）中的IVFPQ算法实现的默认方案采用的是方案3，因而本项目也应该使用方案**3**。

- https://github.com/facebookresearch/faiss/blob/main/faiss/IndexPQ.cpp
- https://github.com/facebookresearch/faiss/blob/main/faiss/IndexIVFPQ.cpp
- https://github.com/facebookresearch/faiss/blob/main/faiss/gpu/impl/IVFPQ.cu

为了方便理解索引和查询过程，以下是方案**3**一个具体的例子（参考自faiss 暴搜/ivf/ivfpq）：

对于索引，比如原始向量[1，2，3，4]，它离一级量化聚类中心向量[1.1，1.9，3.1，4.2]最近，这样原始向量[1，2，3，4]与这个一级量化聚类中心向量[1.1，1.9，3.1，4.2]的残差等于[-0.1，0.1，-0.1，-0.2]，假定残差训练分2段，且其中[-0.1，0.1]与第0段的细聚类中心向量[-0.07，0.15]最近邻，[-0.1，-0.2]与第1段的细聚类中心向量[-0.13，-0.17]最近邻，那么原始向量[1，2，3，4]就被量化为：

1. 它在一级量化聚类中心向量[1.1，1.9，3.1，4.2]的倒排链中，假定倒排链id = 10
2. 它的残差的乘积量化，分别是[-0.07，0.15]、[-0.13，-0.17]两个分段细聚类中心，假定两个细聚类中心id分别是15、25
3. 那么原始向量[1,2,3,4]，被归到"粗聚类[10][1525]"的向量集合中。IVFPQ索引不再存储[1，2，3，4]的原始向量，而是存：倒排链id(10)、所属的两个分段细聚类id(15、25)

对于查询，如向量[0.9，2.1，3.05，3.95](假定它确实是原始向量[1，2，3，4]的近邻)，以下是查询步骤：

1. 它和全部一级量化聚类中心向量做暴搜比较，找到了最近邻的一级量化聚类中心向量[1.1，1.9，3.1，4.2]，残差 = [-0.2，0.2，-0.05，-0.25]
2. 对残差分段，第0段是[-0.2，0.2]，第1段是[-0.05，-0.25]
3. 第0段残差[-0.2，0.2]，和第0段各个细聚类中心暴搜比较，得出最近邻的是[-0.07，0.15]
4. 第1段残差[-0.05，-0.25]，和第1段各个细聚类中心暴搜比较，得出最近邻的是[-0.13，-0.17]
5. 那么待查询向量[0.9，2.1，3.05，3.95]，与"以[1.1，1.9，3.1，4.2]为一级量化结果，且第0分段残差细聚类中心是[-0.07，0.15]，第1分段残差细聚类中心是[-0.13，-0.17]"代表的原始向量，distance = dist([0.9，2.1，3.05，3.95]，[1.1，1.9，3.1，4.2]) +dist([-0.2，0.2]，[-0.07，0.15]) + dist([-0.05，-0.25]，[-0.13，-0.17])，即：待查询向量与粗聚类中心距离 + sum各分段(dist(待查询向量与粗聚类中心残差分段，分段对应的最近邻细聚类中心向量)

   如果是L2度量：

distance = || x - y_C || ^ 2 + sum各分段(|| x - y_C - y_R || ^ 2)

如果是IP（内积）度量：

distance = x|y_C + sum各分段((x - y_C)|y_R)

对于优化，针对L2度量，distance = || x - y_C || ^ 2 + sum各分段(|| x - y_C - y_R || ^ 2) = || x - y_C || ^ 2 + sum各分段(|| x - y_C || ^ 2 + || y_R || ^ 2 + 2 * (y_C|y_R) - 2 * (x|y_R))，在这里，|| y_R || ^ 2 和 2 * (y_C|y_R)可以构建索引时提前算好。

更详细的算法流程描述可见ANN召回算法之IVFPQ，这里不再摘抄。

2.2 具体方案

目前由于IVF-Flat已经在OceanBase中实现（这里不再摘抄重复OceanBase向量数据库设计文档中的内容），IVF-PQ与其有很多重叠的地方，所以在项目期间，我会研究已有的oceanbase中IVF Flat的代码：
https://github.com/oceanbase/oceanbase/tree/vector_search/src/share/vector_index，

- ob_ivfflat_index_build_helper
- ob_ivfflat_index_sample_cache
- ob_ivfflat_index_search_helper
- ob_tenant_ivfflat_center_cache
- src/sql/das相关代码
- src/sql/engine/table/ob_table_scan_op
- 其他相关代码

对于该项目，我们应该可以对IVF-FLAT进行重构，通过提取KMeans算法实现，并抽象出IVF相关的build和search基类，IVF-Flat和IVF-PQ继承该基类并实现各自的算法独特的部分。同时，对Cache进行抽象，统一缓存管理。这样的重构可以提高代码的可维护性和复用性：

1. 定义KMeans算法的基类和具体实现：将KMeans算法抽象出来，创建一个基类KMeansBase，然后创建具体的KMeans算法实现类，如KMeansPlusPlus和ElkanKMeans。
2. 抽象IVF相关build和search的基类：创建IVFBase类作为IVF相关操作的基类。具体的IVF-Flat和IVF-PQ将继承IVFBase并实现各自的算法。
3. 重构Cache：保留现有的缓存逻辑，根据需要对接新的Cache结构，将IVF相关的Cache抽象出来，创建一个基类IVFCacheBase。

PQ算法相关配置项，这些可在已有**IVF-Flat**配置项上，额外作为租户级配置项，参考自RAFT：https://docs.rapids.ai/api/raft/nightly/pylibraft_api/neighbors/#ivf-pq

- pq_bits：int，默认值 = 8

  量化后向量元素的位长度。

- pq_dim：int，默认值 = 0

  乘积量化后向量的维度。当设置为0时，会使用启发式方法选择一个最佳值。注意，pq_dim * pq_bits 必须是8的倍数。提示：较小的‘pq_dim’会导致较小的索引大小和更好的搜索性能，但召回率较低。如果‘pq_bits’为8，‘pq_dim’可以设置为任何数字，但8的倍数对于良好的性能是可取的。如果‘pq_bits’不是8，‘pq_dim’应为

8的倍数。为了获得良好的性能，‘pq_dim’最好是32的倍数。理想情况下，‘pq_dim’也应该是数据集维度的约数。

注意：如果dim不是pq_dim的倍数，则总会对输入数据和查询应用一个随机旋转，将工作空间从dim转换为rot_dim，rot_dim可能比原始空间稍大，并且是pq_dim的倍数（rot_dim % pq_dim == 0）。但是，当dim是pq_dim的倍数时，这种转换是不必要的（dim == rot_dim，因此不需要添加“额外”的数据列/特征）。默认情况下，如果dim == rot_dim，旋转变换将初始化为单位矩阵。

对于**IVFPQ**索引，我们分为三个部分，第一个是IVF聚簇中心的列表，第二个是每个向量分段对应残差向量聚簇中心的列表，第二个是倒排索引表，记录了每个聚簇中心所分配到的每个向量分段对应聚簇中心id，我们不妨将第一个表称为主辅助表，第二个表称为次辅助表，第三个表称为索引表。

因为计算PQ算法过程需要依赖总体聚簇中心计算出来，然后计算残差，这部分残差是中间结果。对于这些中间结果，我们可以专门用一张临时表，在创建索引的时候用。这个表用三个列，分别是对应索引表名，索引表每个向量的id（主键），以及对应残差。当构建索引完成后，清除该临时表内对应该索引表的数据。

对于主辅助表，其与IVFFlat中辅助表的功能以及存储内容一致；对于次辅助表，其类似于主辅助表，不过增加segment_idx列记录该残差向量聚类中心属于原始向量的第几段，在该表中，我们还可以存储一些和检索相关优化部分提到的预计算值，从而进行加速。对于索引表，我们不再存储原始向量，而是存储每个向量分段对应聚簇中心id形成的压缩版向量。

对于检索，因为主次辅助表数据在索引构建完成之后也不会变化，因而对于IVF-PQ，我们应当继续采用IVF-Flat的辅助表数据的缓存功能，从而在大部分场景下主次辅助表数据可以直接通过缓存得到。当查询路径是IVFPQ索引时，为related_table_ids同时增加主辅助表和次辅助表的table_id，以形成主次辅助表和索引表之间的table以及tablet关联关系。对于DAS层的改动，抽象出ObIvfAnnScanOp基类，实现ivf算法，ObIvfflatAnnScanOp以及ObIvfpqAnnScanOp都继承该基类，实现各自算法特有的流程。

对索引过程相关优化，目前没有明确的优化技巧，在项目期间，我会进一步研究Faiss和Raft 的实现，尽力做到最优。

3. 时间安排

在项目申请期间，我已经阅读了OceanBase 开发者手册，熟悉了OceanBase的架构、开发规范以及代码风格要求，并且提交了3个PR修复了编译错误。

我知道遇到问题时优先应该在互联网上搜索并参考有关资料，因而我只会和导师讨论自己无法克服的困难。我会每周通过电子邮件向导师发送一份“周期总结”，告知我到目前为止的工作进展和这周取得的进展。

以下是时间表：

A. 第1-4周 (7.1 - 28, 2024)共计4周

● 熟悉已有IVF-Flat实现，尝试对IVF-Flat进行重构，提取出可复用的Kmeans算法实现，并抽象出IVF基类，IVF-Flat继承该基类实现算法。

B. 第 5-10周 (7.29 - 9.8, 2024) 共计6周

● 完善IVFPQ向量索引设计方案，主要包括建立IVFPQ索引和通过IVFPQ索引进行ANN查询。

● 编写向量数据库使用手册中关于IVFPQ的部分：

https://github.com/oceanbase/oceanbase/blob/vector_search/docs/vector-search-manual.md

● 根据设计方案完成编码工作。

● 编写可验证IVFPQ索引功能的mysqltest测试用例，并通过已有mysqltest测试集。

C. 第11-13周 (9.9 - 30, 2024) 共计3周

● 使用ann-benchmark得到fashion-minst-784、sift-128以及glove-100测试集的测试结果，产出与oceanbase向量分支已有IVFFlat索引的性能对比分析报告。

D. 未来 (10.1, 2024 -)

在成功完成此暑期2024项目后，我将继续为 Oceanbase做出贡献并尽我所能继续维护。如果可能的话，我会很高兴在未来的暑期开源活动中成为Oceanbase社区的导师。