用于减少近似最近邻搜索中共享内存组冲突的自定义 8 位浮点数值格式

1st Hiroyuki Ootomo东京工业大学计算机科学系日本东京
o
oto
mo.h@rio.gsic.titech.ac.jp

2nd Akira Naruse*NVIDIA*日本东京
anaruse@nvidia.com

arXiv:2301.06672v1 [cs.DC] 17 Jan 2023

摘要— **k最近邻搜索广泛应用于机器学习、计算机视觉、数据库搜索和信息检索等领域。精确最近邻搜索的计算成本极高，因此近似最近邻搜索（ANNS）备受关注。IVFPQ 是一种 ANNS 方法。虽然我们可以利用共享内存的高带宽和低延迟，在 NVIDIA GPU 上执行 IVFPQ 的搜索阶段，但共享内存组冲突会导致吞吐量下降。为减少存储体冲突并提高搜索吞吐量，我们提出了一种自定义 8 位浮点数值格式。该格式没有符号位，并且只需少量指令即可与 FP32 相互转换。我们将该格式用于 GPU 上的 IVFPQ，与 FP32 和 FP16 相比，在召回率没有显著损失的情况下实现了更好的性能。**

索引词—**近似最近邻搜索、IVFPQ、浮点值**

I. 引言

A. NVIDIA GPU架构与共享内存

在 NVIDIA GPU 中，一个线程束中的 32 个线程会并发执行。共享内存是一种片上内存，与设备内存相比，它具有更高的带宽、更低的延迟和更小的容量。共享内存包含 32 个存储体，每个存储体的数据宽度为 4 字节，线程通过这些存储体访问共享内存。当所有线程访问不同的存储体时，它们会并行执行。然而，当线程束中的某些线程访问不同地址、但使用同一存储体时，它们会依次执行。这称为“存储体冲突”，应当避免，以获得更高的性能。

B. 近似最近邻搜索

k-最近邻搜索（k-NNS）旨在为给定的查询向量 q，从数据集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ 中寻找 k 向量$V = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \cdots, \mathbf{x}_{i_k}\}$，其中

$$i_1, i_2, \cdots, i_k = \text{k-argmins}_i \left(|\mathbf{q} - \mathbf{x}_i|^2 \right).$$

每个向量都是 d 维的。 k 近似最近邻搜索会近似计算 k-NNS，这意味着结果向量不总是排名前 k 的最近邻。我们通过如下方式计算“召回率”，以评估 k-ANNS 的准确性：

$$\text{recall} = |V \cap V_{GT}|/|V_{GT}|,$$

其中 V_{GT} 是一组真实向量。

C. IVFPQ 算法

IVFPQ 算法 [1] 分为两个阶段：索引构建和搜索。索引构建阶段的流程如下。

1) 将数据集划分为若干簇，例如使用 k均值算法。 2) 在每个簇中，计算簇内每个向量与该簇质心之间的残差向量。 3) 将残差向量$\mathbf{r}$ 分组（乘积量化），并进行量化。通过此过程，将 d维向量$\mathbf{r}$ 压缩为 s维向量$\mathbf{y}$，其中每个元素被量化为 2^p 种模式（$s < d$ 和 p 称为“PQ 位”）。$s = d/2$ 的示例如下。

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_d \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} r_{d-1} \\ r_d \end{bmatrix} \end{bmatrix} \begin{matrix} \xrightarrow[\text{Quantize}]{} \\ \vdots \\ \xrightarrow[\text{Quantize}]{} \end{matrix} \begin{bmatrix} y_1 \\ \vdots \\ y_s \end{bmatrix} = \mathbf{y}$$

4) 保存簇归属信息、压缩后的数据集$\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$，以及 $[r_t \cdots r_u]^\top \underset{\text{Quantize}}{\rightarrow} y_v$的对应关系，该对应关系称为 PQ 码本。

然后，给定一个 d 维查询向量$\mathbf{q}$ 时，搜索阶段的过程如下。

1) 选取一个或多个质心与查询向量相对较近的簇。2) 在每个簇中，计算查询向量与质心之间的残差向量$\mathbf{r}^q$ 。3) 计算 PQ 码本中每个子向量与$\mathbf{r}^q$ 对应位置上的子向量之间的 L2 范数平方。通过这一过程，我们可以查阅

(*) 本研究是 Hiroyuki Ootomo 在 NVIDIA 实习期间参与研究内容的一部分。

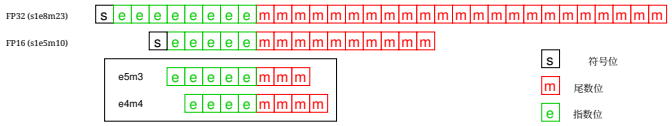


图 1：我们提出的 8 位浮点数值格式与 IEEE binary32/16 的比较。

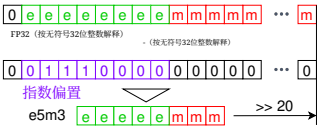


图 3：从 FP32 转换为 e5m3。

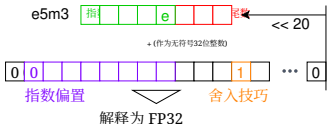


图 4：从 e5m3 转换为 FP32。

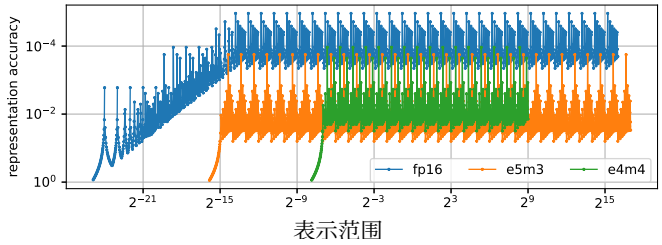


图 2：E5M3 和 E4M4 的表示范围与精度

$|[r_t \cdots r_u]^\top - [r_t^q \cdots r_u^q]^\top|^2$ PQ 索引 y_v。我们将这一查阅过程表示为函数 $F(y_v)$。

4) 对簇中的每个压缩数据集向量 $\mathbf{y}$ ，计算

$$R = \sum_{j=1}^{s} F(y_j). \tag{1}$$

由于残差向量$\mathbf{r}$ 和 $\mathbf{r}^q$的残差对应于原始向量 $\mathbf{x}$ 和 $\mathbf{q}$ 的残差，因此 R 是其 L2 范数的平方。

5) 找出 top-k 个最近向量，并对簇中每个向量 y 的 R 进行（部分）排序。

首先为 F（y_v）构建参考表后，只需从内存中加载片段并累加，即可计算 R 。我们将此参考表称为“norm2 片段查找表”。

D. 共享内存中的 *norm2* 片段查找表

在 GPU 上实现 IVFPQ 搜索阶段时，自然可以将范数片段表放在共享内存中，并让每个线程计算同一簇中不同 $\mathbf{y}$ 的 R 。当 PQ 位为 $p = 8$ 时， $F(y_l)$ 的模式数量为 $2^8 = 256$。当函数的返回数据类型为 FP32（4 字节）时，共享内存中的每个存储体保存256/32[=个存储体]/（4[=数据类型大小]/4[=存储体宽度]）8 $8 - 1 = 7$ 个通道中的元素。因此，在最坏情况下可能发生存储体冲突。我们分析了运行时存储体冲突的数量，发现其会降低吞吐量。当返回数据类型使用 FP16（2 字节）时，由于 1 个存储体可同时访问 2 个元素，因此只需 4 个通道。在这种情况下，最坏情况下只会发生 $4 - 1 = 3$ 存储体冲突。那么，使用 FP8（1 字节）如何?

II. 用于 GPU 上 IVFPQ 搜索的自定义 8 位浮点数值格式

我们设计了两种用于数据存储的 8 位浮点数值格式，以减少存储体冲突：e5m3 和 e4m4，其中指数位数分别为 5 和 4，尾数位数分别为 3 和 4，如图 1 所示。由于 F (y_v) 的返回值始终为零或正数，因此不需要符号位。这一特性使我们能够增加指数位或尾数位，并通过两条指令轻松地将其与 FP32 格式相互转换：移动浮点数值位串以及调整指数偏置。我们在图 2 中展示了这些格式的表示范围和精度。e5m3 可以表示 FP16 规格化数的完整范围，但精度较低。这些格式的特性和实现如下：

•仅用于数据存储。两种格式之间不进行计算。

• 由于要存储的值始终为正数，因此无需符号位。• 从 FP32 转换或转换为 FP32 的开销较小。• 不支持非规格化数和 Inf/NaN。

我们在图 3 和图 4 中展示了从 FP32 转换及转换为 FP32 的操作。

III. 评估

我们评估了在共享内存中将 e5m3 用作 norm2 片段查找表的数据类型时的召回率和吞吐量。我们的基础实现为 LIBCUANN [2]。用于评估的数据集为 BIGANN-100 M 和 DEEP-100M，其中 $d = 128$ 分别为 96。我们在图 5 和图 6 中展示了召回率和吞吐量。与 FP32 和 FP16 相比，在 BIGANN-100M 和 DEEP-100M 上使用 e5m3 均实现了更高的吞吐量，同时召回率略有下降。我们也评估了 e4m4，其召回率和吞吐量与 e5m3 处于同一水平。

IV. 结论

我们设计了定制的 8 位浮点数值格式，通过减少共享内存组冲突来提高 IVFPQ 的吞吐量。我们将其应用于 IVFPQ，在召回率略有下降的情况下提升了吞吐量。

参考文献

[1] Herve J´egou、Matthijs Douze 和 Cordelia Schmid。《用于最近邻搜索的乘积量化》。*IEEE* 模式分析与机器智能汇刊，33(1):117–128，2011 年 1 月。会议名称：IEEE 模式分析与机器智能汇刊。

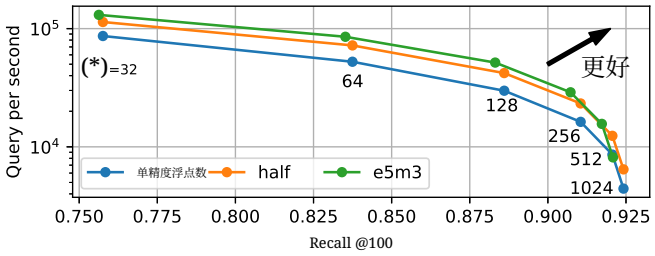


图 5：使用单精度浮点数、半精度浮点数和 e5m3 作为 norm2 片段查找表的数据类型时，BIGANN-100M 数据集上的召回率和吞吐量。(*) 表示搜索阶段第一阶段选取的簇数量。

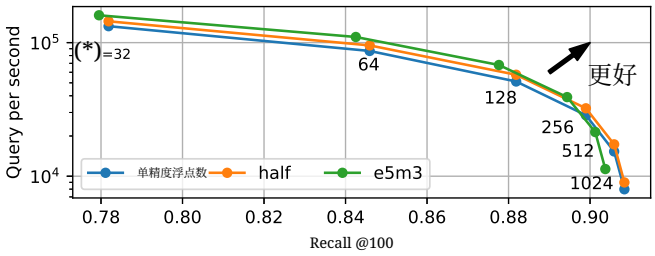


图 6：使用单精度浮点数、半精度浮点数和 e5m3 作为 norm2 片段查找表的数据类型时，DEEP-100M 上的召回率和吞吐量。(*) 表示搜索阶段第一阶段选取的簇数量。

[2] Harsha Vardhan Simhadri, George Williams, Martin Aum¨uller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subra- manya, and Jingdong Wang. 关于十亿规模近似最近邻搜索的 NeurIPS'21挑战赛结果, 2022年5月。编号：arXiv:2205.03763 arXiv:2205.03763 [cs]。