ZARC：精确近邻 数秒内搜索十亿个向量的最近邻

赵东方 (dzhao@uw.edu)

arXiv:2611.29252v1 [cs.DC] 22 Nov 2026

摘要

精确的 k 近邻向量搜索是机器学习系统关键路径上的重要环节，应用涵盖最新大语言模型（LLM）中的稀疏注意力、长上下文记忆以及训练数据去重。GPU 穷举搜索会针对每个查询扫描整个数据集，而近似索引返回的邻居并不精确。近期方法通过廉价的半精度证明，判断向量距离过远，从而跳过几乎所有向量；但这类方法只能在一块能够容纳整个数据集的 GPU 上运行。我们的洞见是，只将这些证明中较短的行保留在 GPU 内存中，并从主机内存中获取少量剩余的完整向量。我们的系统 ZARC 可在更新期间精确回答欧几里得、余弦、内积和过滤查询。在四块 H100 GPU 上，它能在 1.1 秒内对十亿个向量执行 10,000 次精确查询，而穷举搜索需要五分钟。在 11 个开放式大语言模型上，它检索精确的前 10 个注意力键的速度比穷举搜索快 2.2× 到 13.1× 倍。在包含一百万个词元的稀疏注意力场景中，其运行速度比使用 FlashInfer 的 bf16 选择快 3.1× 倍；后者对大多数查询都无法找出精确的前 64 个结果。

1 引言

精确 k近邻向量搜索处于机器学习系统的关键路径上，每个遗漏的邻居都会导致模型或其数据出现错误。在大语言模型（LLM）中，DeepSeek-V4（DeepSeek-AI，2026年）等稀疏注意力机制，对于每个查询词元，只关注前文中排名前k的条目。kNN-LM（Khan- delwal 等人，2020 年）和记忆型 Transformer（Wu 等人，2022）等长上下文记忆，同样会关注内存中最近的键，而这份内存会随着每个生成的词元不断增长。除注意力机制外，SemDeDup（Abbas 等人，2023年）等语义去重方法也依赖近邻，CLIP（Radford 等人，2021年）和 DINOv2（Oquab 等人，2024年）等表征的评估以及近似搜索本身也同样如此

预印本。DOI：10.5281/zenodo.23674584。代码：https://github.com/hpdic/libcertex。版权所有 © 2026 赵东方。

本身（Aum¨uller 等人，2020 年；Simhadri 等人，2022）。这些工作负载以批次方式处理数亿至数十亿个向量，这些向量会随时间变化，并携带筛选器的元数据。

GPU 向量搜索系统要么计算每个距离，要么搜索近似索引。

Faiss 中的穷举搜索

Faiss（Johnson 等人，2021年）和 cuVS（NVIDIA，2026年）会针对每个查询扫描每个向量。因此，在四块 H100 GPU 上，对 BIGANN-1B（Jegou 等，2011年）执行 10,000 次查询约需五分钟（图 1(a)）。穷举搜索还会将原始向量保存在设备内存中，而模型权重和 KV 缓存也会占用这部分内存。仅十亿个 128 维向量以单精度存储就占用 512 吉字节（图 1(b)）。

HNSW（Malkov 与 Yashunin，2020 年）、DiskANN（Subramanya 等人，2019年）、CAGRA（Ootomo 等人，2024年）以及乘积量化倒排文件（J´egou 等，2011年；Babenko 与 Lempitsky，2012 年）等近似索引返回的邻居并不精确，且不提供针对单个查询的保证，其误差会影响模型及其数据。例如，在精度最高的设置下，CAGRA 对至多 45% 的查询返回了通义千问2.5-7B（Yang 等人，2024年）的精确的前 10 个注意力键。类似地，NN-Descent （ Dong 等人，2011 ）会使 DataComp（Gadre等人，2023年）去重图中 3% 至 56% 的行变得不精确（图 1(c) ）。在选择率为 0.1% 的 BIGANN-10M 数据集上，过滤图索引 ACORN（Patel等人，2024年）仅对 3.4% 的查询返回精确的邻居。

这两种设计在跳过向量所依据的证据上有所不同。穷举搜索不会跳过任何向量，因此每个距离都必须计算；而近似搜索会跳过一些向量，却无法证明它们位于第 k 个距离之外，因此会漏掉邻居。只有具备此类证明，即能够证明某个向量的距离下界高于第 k 个距离的经认证下界，才能在不损失精确性的前提下跳过该向量。可认证排除（Zhao，2026年）通过半精度计算下界，并将乘积的舍入误差纳入每个界中，从而以张量核心的速度获得这一证明。因此，只有少数未被跳过的向量需要计算精确距离。然而，率先引入可认证排除的 CertEx 引擎要求将整个数据集保存在一块 GPU 的内存中，并且不支持删除、并发更新、过滤器或内积。

这些证明只读取每个向量中一行简短的半精度数据。我们的关键思路是仅将这些行保留在 GPU

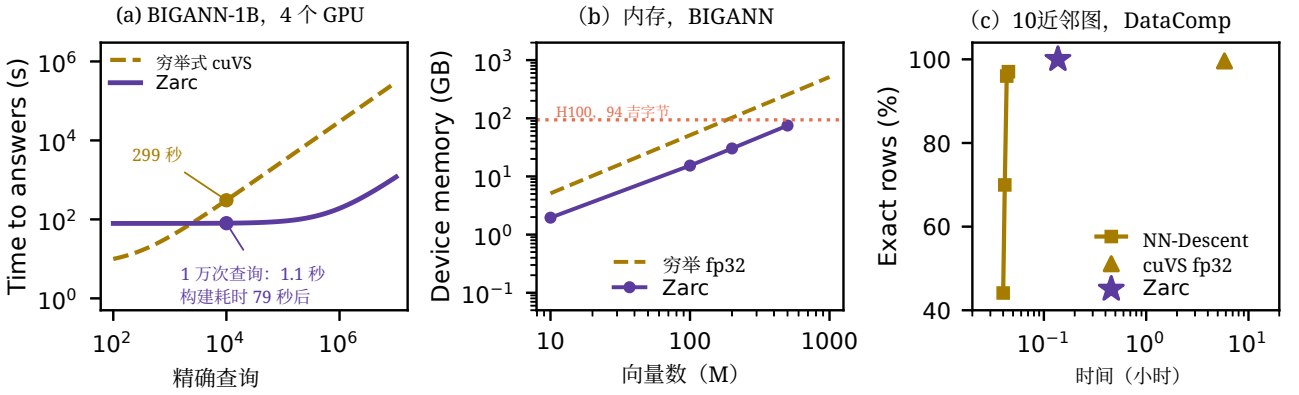

图 *1*。 精确搜索的时间、内存和精确性成本，以及 ZARC 的影响。

内存中，并将全精度向量存放在主机内存中，因为精确验证只需读取少数幸存向量的完整数据。本文将上述思路实现为 ZARC，这是一种精确向量搜索系统，采用 cuVS （NVIDIA，2026年）的接口，并以 Python 包的形式安装。ZARC 根据数据集的划分推导第 k 个距离的紧确上界，并按照经认证的下界顺序，从主机内存中验证幸存项；因此，即使面对插入、删除、并发查询、过滤器和内积查询，也能确保每个答案都精确无误。

我们在最多包含十亿个向量的数据集以及十一种开放式大语言模型的注意力键上评估 ZARC，并将其与 cuVS（NVIDIA，2026年）暴力搜索、CAGRA（Ootomo 等人，2024年）、IVF-PQ（J´egou 等，2011年）、DiskANN（Subramanya 等人，2019年）和过滤图索引进行比较。图 1 展示了 ZARC 对精确搜索的时间、内存和精确性的影响。在四块 H100 GPU 上，ZARC 用时 79 秒为 BIGANN-1B（Jegou 等，2011年）建立索引，并在 1.1 秒内对相同的 10,000 个查询给出精确结果（图 1(a)）。对于从 70 亿到 1.6 万亿参数的十一种大语言模型，ZARC 以穷举式 cuVS 搜索 2.2× 至 13.1× 倍的吞吐量返回精确的前 10 个注意力键，而大语言模型代码执行的 bf16 矩阵乘积会对 14% 至 86% 的查询漏掉精确的前 10 个结果（图 3）。在百万 token 上下文上的前-k 个稀疏注意力中，与使用 FlashInfer（Ye 等，2025年）的 top-k 内核进行 bf16 选择相比，ZARC 的精确选择使 Qwen2.5-7B-Instruct-1M（Yang 等，2025年）的前向传播缩短了 3.1×，且耗时低于密集注意力（图 5 ）。在一块 H100 上，ZARC 仅用 71 吉字节设备内存即可对 BIGANN-500M 执行精确搜索，而穷举搜索需要 256 吉字节（图 1(b)）。此外，在 0.1% 的选择率下，ZARC 的运行速度比过滤版 CAGRA 快 692×，并且为 DataComp-12.8M（Gadre等人，2023年）构建精确的 10近邻图的速度比穷举搜索快 42×（图 1(c)）。

本文的贡献如下：

- 我们设计了跨越内存层级的经认证筛选，层级结构，其中半精度筛选行存放在 GPU 内存中，全精度向量存放在主机内存中，并按照经认证的下界顺序验证幸存项。根据数据分区得出的阈值，以及针对每个分区拟合的投影，将查询通过 PCIe 读取的原始向量数量减少到 BIGANN-500M 上的 699（Jegou 等人，2011年）。更新、过滤器和内积复用相同的经认证的界。

- 我们将 ZARC 实现为一个库，采用 cuVS(NVIDIA，2026年) 的接口，提供面向 Hopper 和 Ampere GPU 的预先编译张量核、并发更新、经校准的成本模型，以及在单台服务器的多个 GPU 上进行的分片搜索。

- 我们在 11 个开放式大语言模型上评估 ZARC，模型参数规模从 7B 到 1.6T；评估场景包括对一百万 token 上下文进行稀疏注意力，以及十亿规模基准测试、过滤搜索、流式更新、语义去重和段落检索。在这些场景中，它对每个查询都能给出精确结果，速度最高可达 GPU 穷举搜索的 270× 倍；在大语言模型注意力键上的速度则快 2× 至 13.1× 倍。

2 动机与挑战

2.1 精确搜索工作负载

现代人工智能系统中的许多关键组件都依赖高效、准确的向量搜索能力，例如稀疏注意力、长上下文记忆和去重。这些应用需要在大型且随时间变化的数据集上获得精确答案。当查询的最大分数近似并列时，仅靠单精度距离并不足以保证精确性，而注意力分数往往正是如此（第4.2节）。例如，GLM-5.3（GLM-5 团队，2026年）的一次查询中，最大的十个注意力分数彼此相差不到 0.03%。这些数据集规模庞大，其原始向量会与模型权重和 KV 缓存争夺设备内存。例如，仅 BIGANN-500M（Jegou 等人，2011年）在单精度下就占用 256 吉字节。这些数据集

系统	超越高带宽内存的精确搜索	更新	精确过滤器	
cuVS 暴力搜索 (NVIDIA, 2026年)	部分	✗	✓	部分
cuVS IVF-PQ（J´egou 等人，2011年）	✗	✓	部分	✗
CAGRA（Ootomo 等人，2024年）	✗	✗	部分	✗
DiskANN（Subramanya 等人，2019年）	✗	✓	✗	✗
FreshDiskANN (Singh 等人，2021年)	✗	✓	✓	✗
ACORN(Patel等人，2024年)	✗	✗	部分	✗
Filtered-DiskANN(Gollapudi等人，2023年)	✗	✓	✗	✗
CertEx（Zhao，2026年）	✓	✗	部分	✗
ZARC	✓	✓	✓	✓

表 *1*。 GPU 和中央处理器向量搜索系统的属性。

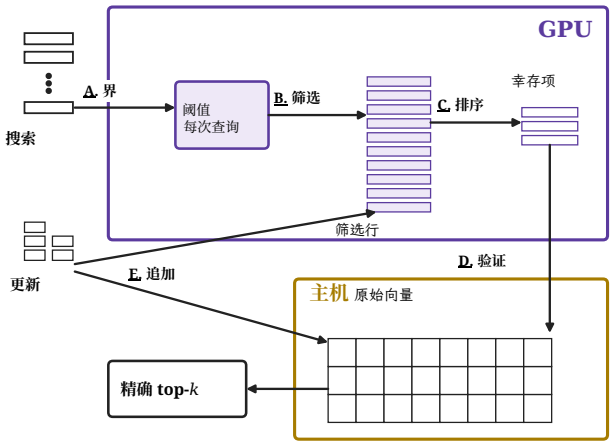

图 *2*。 ZARC：搜索路径（A 至 D）和更新（E）。

也会在查询运行期间发生变化，因此不能每次变化都重建索引。

查询的形式也各不相同，并以批次形式到达。查询可以将其邻居限制为某个元数据标签，或按内积对其进行排序，就像 TAS-B 等稠密检索器（Hofst¨atter 等，2021 年）所做的那样。去重、基准评测，以及 Orca（Yu 等，2022）和 vLLM（Kwon 等，2023年）等大语言模型服务系统的解码批次，都会一次提交许多查询。因此，对索引进行一次遍历即可服务整个批次。

2.2 现有系统的比较

表1 比较了向量搜索系统在这些属性上的表现，其中部分标记表示单精度精确性，或表示插入操作不包含删除。Faiss（Johnson 等人，2021年；Douze 等人，2026年）和 cuVS（NVIDIA，2026年）的穷举搜索虽然精确，但会将数据库保存在设备内存中，并针对每次查询扫描数据库。倒排文件（Babenko 与 Lempitsky，2012）和 HNSW 等近邻图（Malkov 与 Yashunin，2020 年）通过仅搜索数据集的一部分来避免全扫描，因此其召回率取决于探测数量或搜索宽度。图也只能通过扩展或重建来适应数据集的变化。例如，在 H100 上，对 880 万个文本段落重建 CAGRA（Ootomo 等人，2024年）需要 27.6 秒；对 BIGANN-10M 的 1000 万个向量重建 ACORN-γ（Patel等人，2024年）则需要 49 分钟（Jegou 等人，2011年）。

筛选器会增加图索引和穷举搜索的成本。在选择率较低时，图索引中的匹配顶点通常会形成非连通子图。因此，ACORN（Patel等人，2024年）和 Filtered-DiskANN（Gollapudi等人，2023年）等过滤图索引会添加与谓词无关或标签感知的边，但仍只能返回近似答案。穷举式 cuVS（NVIDIA，2026年）搜索借助位图预过滤仍能保持精确，但仍需承担全扫描的成本。

2.3 CertEx 的认证排除及其局限

CertEx（Zhao，2026年）会在某个向量与查询之间的距离下界超过 k阈值时跳过该向量，从而将第 k个近邻与查询之间的距离限定在上界以内。该阈值是查询与存储向量的均匀样本之间的第 k小精确距离，CertEx 将这些样本称为锚点。每个向量的下界只需要一条很短的筛选行即可表示；该筛选行以半精度保存向量在数据集主成分上的投影。一次张量核心乘积即可评估某个查询对应的所有筛选行的界。由于每个界都包含该乘积的最坏情况舍入误差，因此被跳过的向量不可能是真正的邻居。随后，仅对通过这一筛选的少数向量以全精度计算精确距离，CertEx 将这些向量称为幸存项。

生产工作负载暴露了 CertEx 的五项局限（Zhao，2026年），因为它会将整个数据集的原始向量和筛选行都保存在同一块 GPU 上。首先，面对聚类数据和近重复数据，均匀样本会使阈值变得宽松（第 3.2 节）。其次，十亿级数据集的原始向量无法与其筛选行一同装入内存（第 3.3 节）。第三，更新必须在不重建的情况下确保所有界仍然有效（第 3.4 节）。第四，精确配置的吞吐量最多可相差 5×，因此需要自动调优投影维度数量和锚点数量（第 3.5 节）。最后，过滤器和内积也需要各自经过认证的界（第 3.6 节）。

3 设计

3.1 系统架构概述

图2展示了 ZARC 如何在 GPU 上分四个阶段执行一批查询，同时将原始向量保留在主机内存中。阈值阶段（A）根据查询最近分区中采样锚点或候选向量的精确距离，为每个查询计算其第 k近距离的阈值。随后，筛选阶段（B）在张量核心上筛选所有向量的半精度行，并保留经认证的下界不超过该查询阈值的每一行（附录A）。每个幸存的查询-向量对（C）都携带其计算得到的乘积，ZARC 据此推导出经认证的下界，并对每个查询的幸存项排序。最后，验证阶段（D）会读取

按此顺序从主机内存中读取幸存项的原始向量，并以双精度计算其精确距离，直到下一个下界超过已找到的第 k近距离。更新（E）会追加新向量的筛选行和原始向量，并标记已删除向量的行，同时查询仍基于索引的一致视图继续运行（第3.4节）。

由于每次排除都有经认证的依据，并且每个可能成为邻居的幸存项都会经过验证，该流水线返回精确的前 k 个结果。这一保证建立在两个不变量之上：每个被排除的行都有高于其阈值的经认证的下界，并且每个阈值都从上方界定 k-th 距离（附录 B）。下面的每个机制都会保持这两个不变量，包括跨存储器和 GPU 的行放置以及并发更新。

ZARC 在持久化张量核心内核上执行筛选，并且从不将乘积矩阵写入内存。每个线程块将一组查询行保存在共享内存中，并通过由异步加载组成的流水线流式处理多组筛选行，其乘积以单精度累加。每个线程块在一个顺序循环中归约其行，而不采用标准矩阵乘法库中的 Split-K 调度，因为舍入误差界仅对这种累加顺序成立。随后，尾处理阶段将每个乘积与其查询的阈值进行比较，并在无需全局同步的情况下压缩幸存项。每个线程使用位掩码标记其幸存项，并通过一次共享内存原子操作预留输出空间。因此，一个空分块只需对每个乘积进行一次比较，无需任何内存流量（附录 C.1）。

3.2 通过划分获得更紧的界

ZARC 通过对数据集进行 k-均值划分来收紧阈值，并将各个划分存储为连续行块。对于每个查询，ZARC 首先依据 CertEx 的锚点阈值（Zhao，2026年），仅对质心距离查询最近的少数划分中的行进行筛选。筛选内核的路由变体通过一次内核启动完成此步骤，使用一个工作项列表，其中每个工作项将一个查询分块与某个划分中的一个行分块配对。在已筛选的行中，ZARC 计算具有最小经认证的下界的少量 k 候选项的精确距离。这些距离中第 k 小的值给出第二个阈值，我们称之为分区阈值。随后，所有行的筛选均在较小的阈值下进行。

无论如何选择分区和候选项，分区阈值都是安全的，因为它是与 k 个不同存储向量之间的精确距离，因此从上界限定了真实的第 k 个距离。因此，分区只会影响 ZARC 的运行速度，而倒排文件的分区则决定查询能够返回哪些向量。去重会执行自连接，其中每个存储向量都是一个查询。这样的自连接会跳过最近分区的筛选。相反，存储中按顺序排列的一个批次的查询来自一个或两个

pa个分区，它们直接给出阈值的 批次。

在聚类数据集上，向量距离一个全局主子空间较远，但接近其所属划分的子空间。因此，ZARC 为每个划分单独建立局部投影，将其投影到该划分以自身质心为中心的协方差矩阵的主特征向量上，从而收紧界。每个划分都使用自身的投影、尺度和舍入误差常数对其行进行编码，使经认证的界在该划分内成立（附录D）。搜索时，通过一次批量矩阵乘积，为每个划分分别对每个查询进行投影；随后，一次路由内核启动便使用相应的查询行筛选每个划分。锚点及后续插入操作保留全局投影和普通内核启动。这样，每个向量只需通过一次认证测试，而不会经过两次。

3.3 GPU 间的分层放置

ZARC 仅将半精度筛选行和每个查询都会读取的锚点样本保存在 GPU 内存中，并将全精度向量保存在固定主机内存中，由 GPU 通过零拷贝访问读取。由于行比全精度向量短数倍，相同的设备内存可以容纳数倍数量的向量；可选的单字节整数行还会使行大小再次减半（附录 E）。然而，主机内存位于 PCIe 之后，其带宽比设备内存的带宽低一个数量级以上。由于幸存项位于分散的位置，验证必须尽可能少地读取原始向量。

ZARC 根据经认证的下界本身对验证进行排序。筛选后，一个内核根据每个幸存项的存储的乘积计算其经认证的下界，并将查询和该下界打包成一个整数键。随后对这些键进行基数排序，按查询分组幸存项，并按照下界递增的顺序排列。验证以批次中所有查询为单位分轮执行：每个尚未完成的查询都会从其下一个幸存项中取出一个窗口，其中幸存项的下界低于其当前第 k 个精确距离。窗口逐轮加倍，而不是为每个查询分配一个 warp；后者会使最困难查询的长幸存项列表在 PCIe 上串行处理。一次扁平启动以双精度计算某一轮的精确距离，并使每个 warp 同时处理多次行读取（附录 C.2），随后第二次启动将这些结果合并到每个查询的前 k 个结果中。当窗口达到高于其第 k 个距离的下界时，查询即完成；这会将从主机内存读取的数据限制为幸存项的一小部分（附录 F）。

筛选行超过单个 GPU 内存容量的数据集会被拆分为连续分片，每个 GPU 对应一个索引，由 ZARC 并行构建。若各分片根据自身的第 k 个距离进行筛选，保留下来的幸存项会多得多。因此，ZARC 分两轮进行搜索。第一轮中，每个分片根据其

自身锚点，为每个查询计算一个阈值。这些分片阈值中的最小值构成全局 k 个距离的界，因为任一分片中任意 k 个不同向量都满足该界（附录 G）。第二轮中，每个分片都在这个共享阈值下进行筛选和验证，之后主机合并各分片的精确答案。每个查询只有一个阈值和 k 对向量会经由互连传输。

3.4 更新，无需重新建立索引

ZARC 在查询运行期间插入和删除向量，无需重建。原始向量和筛选行的主机与设备缓冲区会预先分配一定容量，插入操作可将行追加到其中。插入操作使用现有投影来投影新向量，因为对于未用于计算该投影的向量，下界仍然有效。然而，每次查询的舍入误差界都取决于所有存储行中的最大范数和误差项。因此，除非先将新行纳入界的范围，否则范数更大的新向量会使每个正在运行的查询的界失效。

ZARC 通过发布协议（附录 C.3）确保遵循这一顺序。更新操作无需加锁即可将行写入已发布行之后的位置，然后在排他锁下发布新的行数以及提高后的误差界。每个搜索批次都以共享模式持有同一把锁。因此，每个批次读取的都是同一个一致视图，并且绝不会从未经过筛选的锚点获取阈值。删除操作会标记已删除向量所在的行，使其无法通过筛选。它还会将该向量从锚点样本中移除。由于插入操作会保持投影不变，ZARC 会定期重新计算投影并重建这些行。

3.5 通过校准成本模型进行调优

ZARC 利用成本模型为每个分区选择投影维度数、每个锚点的向量数、查询批大小以及全精度向量所在的内存层级。该模型通过微基准测试针对每个设备校准一次，并根据已筛选行、锚点距离和每次查询的预期保留向量数预测一批查询的耗时（附录 H）。由于每种配置都能保持所有界有效，因此错误的预测只会影响速度。

每个查询的幸存项数量取决于数据，其规律无法由任何静态规则捕捉（附录I）。例如，最佳维度数会随着数据集增大而减少，并且取决于查询分布。因此，调优器会在数据集的两个较小前缀上运行搜索，并通过幂律将幸存项数量和每个查询的已验证行数外推到完整规模。然后，成本模型根据预测的批次耗时对配置进行排名，并保留速度最快的配置。

3.6 过滤查询与内积查询

筛选后的查询会将其邻居限制为带有指定标签的向量，例如某个租户或类别。筛选无需匹配向量之间存在任何连通性，这使得筛选操作归结为从筛选范围中移除行。只有阈值取决于标签，因为其他标签向量的阈值可能低于匹配向量中的第 k 个距离，从而排除真正的邻居。

ZARC 以两种方式支持为每个向量和每个查询分别指定一个标签。第一种方式中，筛选内核的尾处理阶段不会保留其他标签的行。这样，查询的锚点仅来自其所属标签。该方式支持任意逐行谓词，例如 cuVS 的位集过滤器（NVIDIA，2026年），这些过滤器只会排除行，而不会改变其他任何界。第二种方式中，ZARC将每个标签存储为由其自身锚点组成的连续行块，并按标签对每一批查询进行排序。随后，路由内核仅将同一标签的每个查询分块与该标签的行配对，因此路由内核的工作量会按选择率成比例缩减（附录 J）。

对于内积查询，ZARC 直接界定内积。将问题标准地归约为欧几里得搜索时，会为每个向量附加一个坐标，使其范数提升至数据集中最大的范数。然而，对这个坐标进行舍入会使范数产生至多一个末位单位的偏差，并可能改变近似并列项的顺序。ZARC 则在构建筛选行时不减去数据集的均值，从而保留每个向量的范数。随后，同一个张量核心乘积就能为每个向量的内积给出上界（附录B）。任何界低于任意 k个不同向量中精确内积第 k 大的向量，都不可能位于前 k名之列。阈值取自锚点的精确内积。随后，验证阶段会以双精度按精确内积对幸存项排序。余弦查询会筛选归一化副本，并针对舍入误差留出余量（附录 K）。

4 评估

我们使用约 15,000 行 CUDA C++和 Python 实现了 ZARC，并配有测试和基准测试（附录 C）。表 2 列出了它的接口，每个接口都保留了现有库的调用方式，涵盖 cuVS 的索引模块（NVIDIA，2026年）、Faiss 的平坦索引（Johnson 等人，2021年），以及 PyTorch（Paszke 等人，2019年）和 FlashInfer （Ye 等人，2025 年）中的 top-k 选择。因此，将代码迁移到精确搜索，所需改动几乎只有调整导入语句。Python 绑定也能直接接收 PyTorch、CuPy 和 NumPy 数组，无需复制。为计算分区阈值，ZARC 会筛选最近的两个分区，并为每个请求的邻居验证四个候选项，这一过程涉及约 0.2% 的行。其张量核心内核会预先编译

库	即插即用组件，适用于	使用场景
cuVS、C++	索引模块（构建、搜索）	向量数据库
cuVS, C	cuvsBruteForce 调用	向量数据库
cuVS、Python	暴力搜索模块	检索增强生成、训练数据去重
Faiss	IndexFlatL2、IndexFlatIP	检索增强生成、基准测试真实值
PyTorch	基于注意力分数执行 torch.topk	稀疏注意力、长上下文记忆
FlashInfer	top-k 页表和不规则变换	大语言模型服务中的稀疏注意力

表 *2.* ZARC 的接口。

以适配 Hopper 和 Ampere GPU，因此批次形状变化时，不会有查询因等待内核调优而延迟。相比之下，CertEx 的 Triton（Tillet 等人，2019年）引擎（Zhao，2026年）会在运行时调优每个内核，而查询只能等待。

4.1 实验设置

开放式大语言模型。该大语言模型工作负载涵盖来自五家开发者的 11 个开放式大语言模型，参数量从 7B 到 1.6T，分别属于 DeepSeek-V4 （ DeepSeek-AI，2026 年；2025 年）、通义千问（Yang 等人，2024 年；2025a；Qwen 团队，2026年）、gpt-oss（OpenAI，2025年）、GLM（GLM-5 团队，2026年）和 Hy3（腾讯 Hun-yuan 团队，2026年）系列。对于每个模型，内存中存放的是 PG-19（Rae 等人，2020 年）训练书籍上某一层在旋转位置嵌入（RoPE）（Su 等人，2024 年）之前的键；在一张 H100 上，前四个 KV 头各存放 1000 万个键。每个注意力头从测试书籍中接收 10,000 个解码查询。对于 DeepSeek 和 GLM 的多头潜在注意力（ DeepSeek-AI，2024 ），内存中存放所有注意力头共享的吸收后的潜在键；它与某个注意力头的吸收后的查询的内积，等于该头在应用 RoPE 之前的注意力 logit。内存不包含每个块的前四个位置，即部署模型会常驻内存的注意力汇（Xiao 等人，2024b）。真实值根据与所有键的 fp32 距离计算，再以 fp64 重新评分，并根据其舍入误差进行核验（附录 L）。

平台。 主要平台是一台服务器，配备四块 94 GB H100 GPU，这些 GPU 两两之间通过 NVLink 连接；此外还配有 192 核 Intel Xeon Platinum 8468 和 960 GB 主机DRAM，用于运行十亿规模和大语言模型实验。另一台服务器配备一块通过 PCIe 5.0 x16 连接的 94 GB H100、同款处理器的 48 个核心以及 240 GB 主机DRAM，用于在一块 GPU 上运行其他实验。两台服务器均运行 Ubuntu 24.04、CUDA 12.6（驱动程序版本为 560.35.05）、CUTLASS 3.9.2、PyTorch 2.14.1 和 cuVS 26.10。在更早一代 Ampere 架构的 A100 上，每个查询仍保持精确，而 ZARC 的运行速度11× 比 26× 穷举搜索更快（见附录 M）。

数据集。向量工作负载采用 SIFT1M 和 BI- GANN（Jegou 等人，2011年）中的 SIFT描述符（Lowe，2004）、ImageNet（Oquab 等人，2024年）的 DINOv2 嵌入（Deng 等人，2009）、DataComp（Radford 等人，2021年）的 CLIP 嵌入（Gadre等人，2023年）、Deep1B（Babenko 与 Lempitsky，2016年）以及 MS MARCO（Nguyen 等人，2016年）的两种嵌入，分别用于余弦和内积检索。BIGANN 和 Deep1B 均达到十亿个向量，而 DINOv2 和 DataComp 包含聚类数据和近重复数据，在这些数据上，统一阈值会放宽。每个数据集都附带所有查询的精确真实值，采用双精度计算得出（附录N）。

基线方法。穷举搜索使用 cuVS（NVIDIA，2026）的单精度暴力搜索；对于过滤查询，则使用位图预过滤器，或仅搜索匹配向量。cuVS 提供的近似 GPU 基线方法包括带位集筛选器的 CAGRA（Ootomo 等人，2024）、带有和不带有精确精炼的 IVF-PQ（J´egou 等人，2011），以及用于构建 10近邻图的 NN-Descent（Dong等人，2011）。CPU基线方法包括在块设备上运行的 DiskANN（Subramanya 等人，2019）、FreshDiskANN（Singh 等人，2021）、ACORN（Patel等人，2024）和 Filtered-DiskANN（Gollapudi等人，2023）。对于大语言模型，基线方法包括使用 torch.topk 的大语言模型代码中的 bf16 和 fp16 矩阵乘积、fp16 和 fp32 下的 cuVS 穷举搜索、CAGRA，以及使用 4,096 个列表的 IVF-Flat。在稀疏注意力中，最强的基线方法是先进行 bf16 矩阵乘积，再使用 FlashInfer（Ye 等，2025）的 top-k 内核。CertExengine（Zhao，2026）用作不采用分区和层级时的筛选参考。

度量指标。 我们在固定批次大小下，以每秒查询数报告吞吐量、每个层级的内存占用，以及构建和更新吞吐量。对于近似基线，精确比例统计其 k 距离在双精度下与精确距离相等的查询数量；在存在并列时，这一标准比 Recall@10 更严格。ZARC 的每个答案都会与真实值进行核对，并且都与之匹配，因此其精确比例始终为 100%，即使在存在精确并列、重复项和极端范数的对抗性输入上也是如此（附录O）。

4.2 开放式大语言模型上的注意力检索

我们评估开放式大语言模型注意力键上的精确 top-k 检索：首先，将其作为算子，在 11 个模型上进行评测，每个 KV 头配备一块 H100；随后，在稀疏注意力中进行端到端评测。长上下文推理会为每个解码查询检索这些键（Wu 等人，2022；Xiao 等人，2024a；Tang 等人，2024；Liu 等人，2025）。大语言模型代码通过 bf16 或 fp16 矩阵乘积后接 torch.topk 来实现检索，其中的舍入会打乱最大分数中近似并列项的顺序。图 3 显示，ZARC 在每个模型的每个查询上都能做到精确，并且速度快于所有精确基线。在 Qwen2.5-7B（Yang 等人，2024）上，近似的 CAGRA（Ootomo 等人，2024）和 IVF-Flat 索引只有在许多查询中漏掉精确 top-10 的情况下才能提升吞吐量，而 ZARC 既精确，又比 bf16 和 fp16 矩阵乘积更快（图 3(a)）。在 Qwen（Yang 等人，2025a；Qwen 团队，2026）、OpenAI（OpenAI，2025）、腾讯（腾讯混元团队，2026）、DeepSeek（DeepSeek-AI，2026；2025）的 11 个模型上，

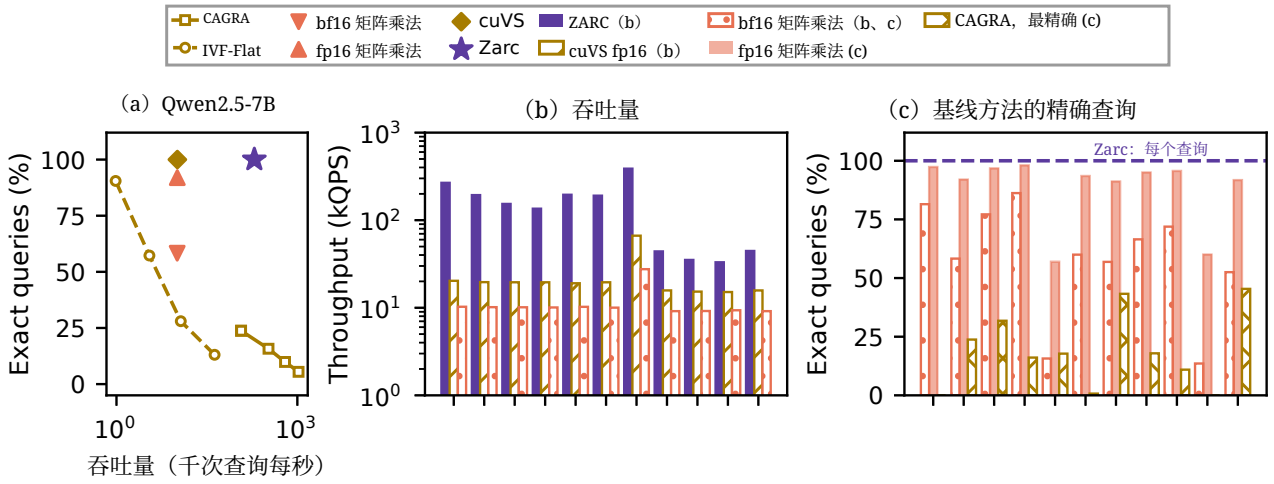

图 *3*。 在 11 个开放式大语言模型上使用每个 KV 头对应一块 H100 得到的精确的前 10 个注意力键。

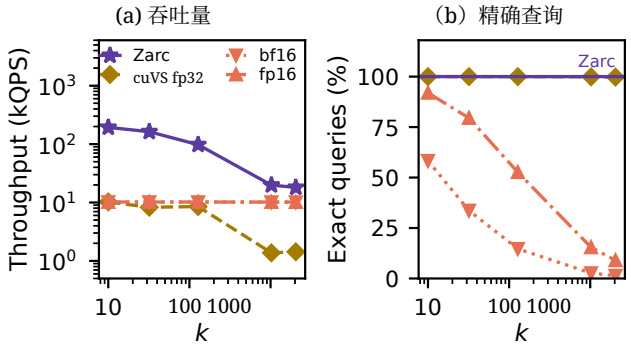

图 *4*。 通义千问2.5-7B 上的精确前-k 与 k 的对比。

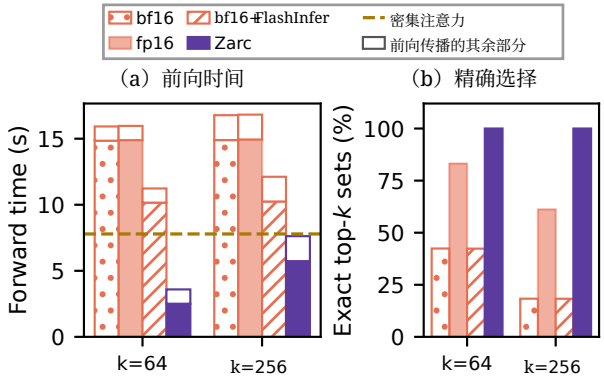

图 *5*。 在超过一百万个词元上端到端运行稀疏注意力。

和智谱（GLM-5 团队，2026年），ZARC 在 gpt-oss、通义千问和 Hy3 的 64 至 256 维键上，每秒可处理 135k 至 388k 个查询，比8× 穷举式 cuVS1× 搜索快NVIDIA，2026年，在 DeepSeek 和 GLM 的 512 和 576 维键上，每秒可处理 33k 至 45k 个查询，比2× 相同搜索快8×（图 3(b)）。然而，bf16 矩阵乘积仅对 14% 至 86% 的查询结果精确，fp16 乘积则对 57% 至 98% 的查询结果精确，而 CAGRA 至多仅对 45% 的查询结果精确（图3(c)）。

图4显示，在通义千问2.5-7B（Yang 等人，2024年）上，ZARC 在不超过 $k = 2,048$的范围内始终保持精确，并且速度快于 bf16 矩阵乘积和 cuVS 穷举搜索；DeepSeek-V3.2（DeepSeek-AI，2025年）的稀疏注意力使用的正是这一范围。在 $k = 2,048$时，ZARC 每秒可处理 18.1k 次精确查询，比 bf16 矩阵乘积快 1。8× 倍，比使用 fp32 的 cuVS 穷举（NVIDIA，2026年）搜索快 12。7× 倍（图 4（a））。在 bf16 矩阵乘积下，这些查询中只有 1.4% 是精确的，而在 fp32 的 cuVS 下则有 99.7% 至 99.8% 是精确的（图 4（b））。

图 5 在四张 H100 GPU 上，对 Qwen2.5-7B-Instruct-1 M（Yang 等人，2025b）端到端评估了 top-k 稀疏注意力。对于两篇 PG-19（Rae 等人，2020 年）文本中的每一篇，模型先填充一百万个词元，然后评估接下来的 2,048 个词元；这些词元的查询头会关注注意力汇、此前已评估的词元，以及某种选择方法选出的 top-k 上下文键。使用 $k = 64$时，ZARC 为每个查询精确选择 top-64，并将前向传播时间缩短至 3.6 秒；相比之下，使用 bf16 矩阵乘积和 FlashInfer（Ye 等人，2025年）的 top-k 内核需要 11.2 秒，使用 torch.topk 需要 15.9 秒，而密集注意力需要 7.8 秒（图5（a））。两种 bf16 选择方法仅对 42% 至 43% 的查询精确选出 top-64（图 5（b））。使用$k = 256$时，前向传播耗时 7.6 秒，而另外两种方法分别需要 12.1 秒和 16.8 秒。在两篇文本上，四种选择方法的困惑度彼此相差均不超过 0.3%。因此，ZARC 在不牺牲质量的情况下，以更短的时间实现了精确选择。精确选择还使所关注的键仅由模型本身决定，而 bf16 和 fp16 矩阵乘积会针对同一查询关注不同的键。

4.3 基于分区的更紧界

我们在 BIGANN（Jegou 等人，2011年）上评估分区阈值和局部投影，此时筛选是主要耗时环节

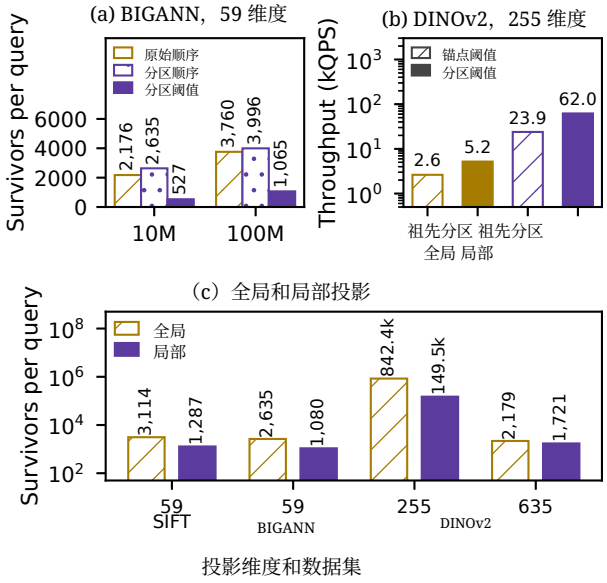

图 *6.* 分区阈值和局部投影。

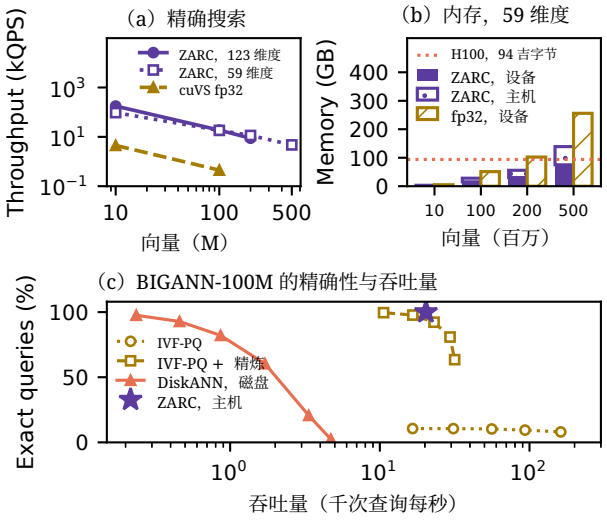

图 *7.* 使用单个 H100 的 BIGANN 分层内存。

每个批次的耗时；在 DINOv2（Oquab 等人，2024年）上，验证则是主要耗时环节。这两种机制只会改变幸存项，因此每种变体都能保持精确。

图6 显示，在幸存项代价较高时，这两种机制能够减少幸存项并提高吞吐量。在 BIGANN（Jegou 等人，2011年）上，采用 59 维投影时，分区阈值在 1000万和 1亿向量处分别减少了 5。0× 和 3。8×个幸存项（图 6（a）），但由于筛选占主导地位，此处吞吐量变化不大。在 DINOv2（Oquab 等人，2024年）上，采用 255 维投影时，局部投影与分区阈值共同提高了吞吐量 24×，从每秒 2.6k 次查询提高到每秒 62.0k 次查询（图6（b））。仅使用局部投影，就将 DINOv2 上每次查询的幸存项从 84.2 万减少到 15 万，并将 SIFT1M 和 BIGANN-10M 上的幸存项减少了 2。4× 倍（图6（c））。

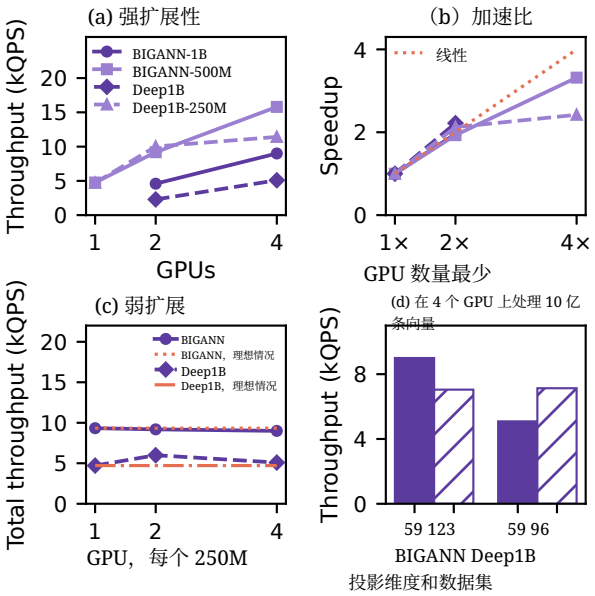

图 *8*。 四个 H100 GPU 上的分片精确搜索。

4.4 分层内存扩展

我们评估了在 BIGANN（Jegou 等人，2011年）的 1000万至 5亿向量前缀上，将原始向量置于主机内存中的方案：在设备内存中存放筛选行以及每 16 个向量取一个的锚点样本。在同一 GPU 上进行穷举 fp32 搜索时，向量数量达到约 1.9亿后便会停止。

图 7 显示，主机内存使单个 GPU 上的精确搜索能够扩展到 5 亿个向量。在 1000万和 1亿个向量的规模下，穷举 fp32 搜索仍可容纳于 GPU 内存中，而 ZARC 运行速度39× 和 42× 比它更快，并且在超过这一规模后仍能持续响应查询：在 5 亿个向量和 59 维投影下，每秒可处理 5,233 个查询（图 7（a））。每个查询只读取其 5,593 个幸存项中的 699 个（附录 F）。在 5 亿个向量的规模下，ZARC 需要 71 吉字节的设备内存，而穷举 fp32 搜索则需要 256 吉字节（图 7（b））。压缩索引和磁盘驻留索引只有牺牲精确性，才能达到这一容量（图 7（c））。cuVS（NVIDIA，2026年）IVF-PQ（J´egou 等人，2011年）需要使用 512 次探测进行精确细化，才能在每秒 1.06 万个查询的速率下，对 99.6% 的查询实现精确结果；相比之下，ZARC为每秒 2.03 万个查询。块设备上的 DiskANN（Subramanya 等人，2019年）则最多只能对 97.6% 的查询实现精确结果，速度为每秒 236 个查询。

4.5 多 GPU 可扩展性

我们在主服务器的四个 H100 GPU 上评估分片搜索，将每个分片的原始向量存放在固定主机内存中，并以穷举式 cuVS（NVIDIA，2026年）在半精度分片上的搜索作为基线。共享阈值确保在任意分片数量下，每个分片的结果都保持精确。

图 8显示，共享阈值可将精确搜索扩展至

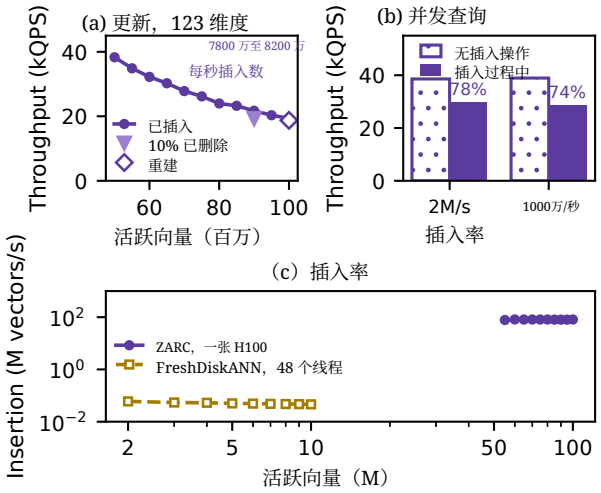

图 *9*。 BIGANN 上的更新与并发查询。

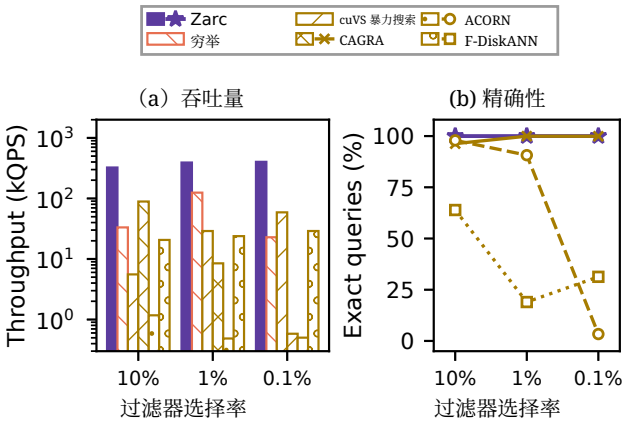

图 *10*。 BIGANN-10M 上的精确过滤搜索。

十亿个向量。在 BIGANN-500M（Jegou 等人，2011年）上，ZARC 在一、两和四块 GPU 上的每秒查询数分别为 4,759、9,188 和 15,796（图 8（a））。在 BIGANN-1B 上，四块 GPU 的吞吐量达到 1。96× 最初容纳数据集的两块 GPU 的吞吐量（图 8（b）），每秒查询数为 8,997， 269× 穷举式 cuVS（NVIDIA，2026年）搜索（图 1（a））。在每块 GPU 处理 2.5 亿个向量的弱扩展测试中，所有 GPU 的总吞吐量在数据集规模达到十亿个向量前始终相差不超过 3.6%（图8（c））。这条平坦曲线代表理想情况，因为每个查询筛选的数据集会随 GPU 数量增加而扩大。最佳投影维度取决于数据集（图 8（d））。在 BIGANN-1B 上，59 个维度的每秒查询数为 8,997，而 123 个维度为 7,046，因为更宽的行会使筛选的字节数翻倍。在 Deep1B（Babenko 与 Lempitsky，2016年）上，完整的 96 个维度每秒查询数为 7,133，而 59 个维度为 5,087，因为完整维度能排除幸存项；否则，验证过程需要从主机内存中读取这些幸存项，每个都是 384 字节的向量。

4.6 流式动态更新

我们通过在十个插入步骤中将 BIGANN（Jegou 等人，2011年）索引中的向量数量从 5000 万扩展到 1 亿，然后删除 10% 的向量来评估更新性能，并将每个查询都与针对活跃向量执行的穷举搜索结果进行核对。以 48 个中央处理器线程运行的 FreshDiskANN（Singh 等人，2021年）作为向图索引执行插入操作的基线。

图 9 显示，更新能够在与重建索引相当的吞吐量下保持每个答案的精确性。插入以每秒 7800 万至 8200 万个向量的速度运行时，吞吐量会随数据集大小变化：从 5000 万向量时每秒 3.83 万次查询下降到 1 亿向量时每秒 1.94 万次查询，这与在 1 亿向量上重建索引时每秒 1.87 万次查询的结果一致（图 9（a））。当一个后台线程以每秒 200 万或 1000 万个向量的速度插入 2500 万个向量时，查询仍能保持其吞吐量的 78% 和 74%，并且核对的 17 万个答案全部精确（图 9（b））。在 48 个中央处理器线程上运行的 FreshDiskANN（Singh 等人，2021年）每秒插入约 1，300× 更慢（图 9（c）），因为它会搜索图，并修复每个新向量周围的边。

4.7 应用工作负载

过滤搜索。 图 10显示，在 BIGANN-10M（ZARC 是每种选择率下最快的精确方法，每秒可处理 32.4 万至 40.3 万次查询（Jegou 等人，2011年），比针对匹配向量进行的穷举式 cuVS（8×快 58×（NVIDIA，2026年）搜索（图 10（a））。在最精确的设置下，CAGRA（Ootomo 等人，2024年）的运行速度 46× 且 692× 比 ZARC 慢，在选择率为 1% 和 0.1% 时尤为如此，因为每个标签都需要单独调用一次 CAGRA。相比之下，过滤图索引 ACORN（Patel等人，2024年）和 Filtered-DiskANN（Gollapudi等人，2023年）分别只能为至多 97.9% 和 63.9% 的查询返回精确邻居（图10（b））。

语义去重。ZARC 仅用 8.2 分钟端到端构建出 DataComp-12.8M 的精确10近邻图（Gadre等人，2023年）（图1（c）和附录P）， 42× 比穷举式 cuVS（NVIDIA，2026年）搜索预计需要的 5.8 小时快 58×，也比 CertEx 引擎（Zhao，2026年）快。近似 cuVS NN-Descent（Dong等人，2011年）至少有 3.0% 的行结果不精确，且未被标记。分区阈值将每个查询的幸存项从 10,486 个减少到 114 个，而验证会读取 41 个。

段落检索。检索增强生成（RAG）（Lewis 等人，2020 年）能够容忍近似邻居。因此，精确搜索在这类场景中的速度必须与近似索引相当。图11显示，ZARC 在 MSMARCO 的 884 万个文本段落上，无论采用余弦相似度还是内积，都能实现精确检索，且速度比精度最高的 CAGRA 设置更快（Nguyen 等人，2016年）。采用余弦相似度时，ZARC 每秒可处理 3.59 万至 5.17 万次查询， 60× 穷举式 cuVS（NVIDIA，2026年）搜索，

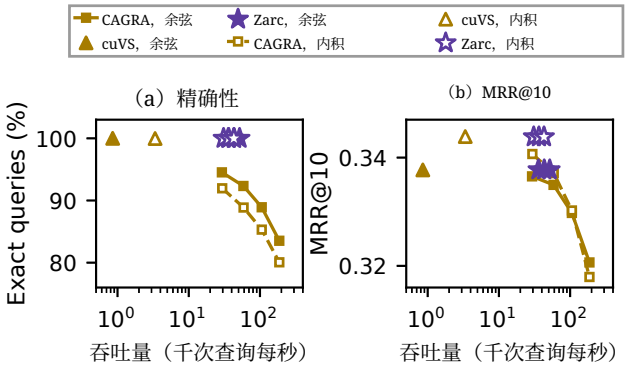

图 *11*。 MS MARCO 上的精确检索。

而精度最高的 CAGRA 设置每秒处理 2.92 万次查询时，会漏掉 5.5% 的查询（Ootomo 等人，2024年）。采用内积时，ZARC 每秒可处理 3.08 万至 4.30 万次查询， 13× cuVS 穷举搜索，而 CAGRA 会漏掉 8.0% 的查询（图11（a））。精确性也体现在检索质量上，MS MARCO 使用前 10 条结果中的平均倒数排名（MRR@10）进行衡量。MRR@10 是对各个查询中排名前 10 的结果里第一个相关文本段落的倒数排名取均值。采用余弦相似度时，ZARC 的 MRR@10 达到穷举搜索的 0.338，而 CAGRA 为 0.321 至 0.337（图11（b））。

5 相关工作

精确搜索。 AESA（Vi-dal Ruiz，1986年）、覆盖树（Beygelzimer 等人，2006）、VP树（Yianilos，1993）、M树（Ciaccia 等人，1997）和 FLANN（Muja & Lowe，2014）等度量访问方法，以及 GEMINI（Faloutsos 等人，1994）、VA文件（Weber 等人，1998）和 iDistance（Jagadish 等人，2005年）等基于界的过滤器，通过指针追踪和标量比较来剪枝候选项，这会使 GPU 的矩阵单元处于空闲状态。从 Garcia 等人（2008）到 Faiss（Johnson 等人，2021年）和 cuVS（NVIDIA，2026年）的 GPU 穷举搜索使用了这些矩阵单元，但会以单精度计算每个距离，这可能导致近似并列的结果排序错误。计算几何中的浮点过滤器（Fortune &Van Wyk，1993；Shewchuk，1997）和验证数值计算（Higham，2002）通过先使用快速浮点数进行计算，并仅在误差界要求时改用精确算术，从而以较低成本解决此类情况。ZARC 将这种舍入误差界应用于筛选每个向量的半精度张量核心乘积。

近似索引与驻盘索引。倒排文件（Johnson 等人，2021年；Babenko 与 Lempitsky，2012；Baranchuk 等人，2018）处理经 OPQ（Ge 等人，2013）、ScaNN（Guo 等人，2020）或 RaBitQ（Gao 与 Long，2024）压缩的向量；而 HNSW（Malkov 与 Yashunin，2020）、NSG（Fu 等人，2019）、CAGRA（Ootomo 等人，2024）、SONG（Zhao 等人，2020）等近邻图，以及 GGNN（Groh 等人，2023年）通过探测数量或搜索宽度来权衡召回率与吞吐量。HM-ANN（Ren 等人，2020 年）、BANG（Venkatasubba 等人，2025年）和 DeepStore（Mailthody 等人，2019年）将十亿规模索引置于异构内存、单个 GPU 或存储中，而 BaM（Qureshi 等人，2023年）则使 GPU 能够直接访问这些索引。DiskANN（Subramanya 等人，2019年）、Starling（Wang 等人，2024年）、SPANN（Chen 等人，2021年）和 FreshDiskANN（Singh 等人，2021年）将完整向量保存在固态存储中，Rummy（Zhang 等人，2024年）和 FusionANNS（Tian 等人，2025年）则让读取操作与中央处理器和 GPU 上的距离计算并行进行。PilotANN（Gui 等人，2026年）、SVFusion（Peng 等人，2026年）、HAVEN（Hsu 等人，2026年）和 IVF-RaBitQ（Shi 等人，2026年）将近似搜索分布在 GPU 服务器的内存层级中，向量所在层级决定其精度；而 ZARC 的层级仅决定向量从何处读取。

过滤搜索。Milvus（Wang等人，2021年）、Manu（Guo等人，2022）、AnalyticDB-V（Wei等人，2020 年）和 VBase（Zhang等人，2023年）等向量数据库将关系谓词与向量索引相结合。在索引层面，ACORN（Patel等人，2024年）和 Filtered-DiskANN（Gollapudi等人，2023年）在图索引中添加与谓词无关或标签感知的边。这两类系统都依赖匹配顶点的连通性。相比之下，ZARC 仅筛查查询标签对应的行。

可认证排除。RaBitQ（Gao 和 Long，2024年；Gao等人，2025年）的误差界以及 ConANN（Horchidan等人，2025年；Vovk等人，2005年）的保形召回保证适用于旋转分布或查询分布，而 ZARC 的保证则适用于每个查询和每个向量。这些界源自 CertEx（Zhao，2026年），该方法针对可放入单个 GPU 的数据提出了可认证排除。ZARC 通过分区阈值、内存层级、分片、并发更新、过滤器和内积对其进行了扩展。

6 结论

本文介绍了 ZARC，使得在十亿级向量数据集和百万词元上下文的规模下进行精确向量搜索成为现实。其核心发现是，经过认证的排除只需在 GPU 内存中保存短小的半精度筛选行，而少数剩余向量则可留在主机内存中。在十亿级基准测试、十一个开放式大语言模型以及超过一百万个词元的稀疏注意力场景中，ZARC 对每个查询都能返回精确答案，同时速度最高可达 270× 倍于 GPU 穷举搜索；在检索方面，其速度也快于处于最精确设置的近似索引。由于 ZARC 保留了 cuVS 、 Faiss 、 PyTorch 和 FlashInfer 的调用方式，采用它所需的改动几乎只有更换导入语句。

参考文献

Abbas, A.、Tirumala, K.、Simig, D.、Ganguli, S. 和 Morcos, A. S. SemDeDup：通过语义去重实现 Web 规模下的数据高效学习。*arXiv* 预印本 *arXiv:2303.09540*，2023年。网址：https://arxiv.org/abs/2303.09540。

Adinets, A. 和 Merrill, D. Onesweep：面向 GPU 的更快速最低有效位基数排序。*arXiv* 预印本 *arXiv:2206.01784*，2022年。doi：10.48550/arXiv.2206. 01784。网址 https://arxiv.org/abs/2206.01784。

Aum¨uller, M., Bernhardsson, E. 和 Faithfull, A. ANN-Benchmarks：用于评测近似最近邻算法的基准测试工具。《信息系统》，87:101374，2020年。网址：https://doi.org/10.1016/j.is.2019.02.006。Babenko, A. 和 Lempitsky, V. 倒排多重索引。载于*2012*年*IEEE* 计算机视觉与模式识别会议，第 3069–3076 页，2012年。网址：https://doi.org/10.1109/CVPR.2012.6248038。Babenko, A. 和 Lempitsky, V. 深度描述符十亿规模数据集的高效索引。载于*IEEE*计算机视觉与模式识别会议（*CVPR*）论文集，第2055–2063页，2016年。网址：https://doi.org/10.1109/CVPR.2016.226。Baranchuk, D., Babenko, A. 和 Malkov, Y. 重新审视十亿规模近似最近邻搜索的倒排索引。载于欧洲计算机视觉会议（*ECCV*）论文集，第209–224页，2018年。网址：https://doi.org/10.1007/978-3-030-01258-813。_Beygelzimer, A., Kakade, S. 和 Langford, J. 用于最近邻搜索的覆盖树。载于第*23*届国际机器学习会议论文集，第97–104页，2006年。网址：https://doi.org/10.1145/1143844.1143857。Blanchard, P., Higham, N. J., Lopez, F., Mary, T. 和 Pranesh, S. 混合精度分块融合乘加：误差分析及其在GPU张量核心上的应用。《*SIAM*科学计算杂志》，42(3):C124–C141，2020年。网址：https://doi.org/10.1137/19M1289546。Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M. 和 Wang, J. SPANN：高效的十亿规模近似最近邻搜索。载于神经信息处理系统进展（*NeurIPS*），第34卷，第5199–5212页，2021年。网址：https://proceedings.neurips.cc/paper/2021/hash/299dc35e747eb77177d9cea10a802da2-Abstract.html。

Ciaccia, P.、Patella, M. 和 Zezula, P. M树：一种用于度量空间中相似性搜索的高效访问方法。载于第*23*届超大型数据库国际会议（*VLDB*）论文集，第426–435页，1997年。网址 http://www.vldb.org/conf/1997/P426.PDF。

DeepSeek-AI。DeepSeek-V2：一种强大、经济且高效的混合专家语言模型。*arXiv* 预印本 *arXiv:2405.04434*，2024年。网址https://arxiv.org/abs/2405.04434。

DeepSeek-AI. DeepSeek-V3.2：推动开放式大语言模型的前沿发展。*arXiv* 预印本 *arXiv:2512.02556*，2025年。网址https://arxiv.org/abs/2512.02556。DeepSeek-AI. DeepSeek-V4：迈向高效的百万词元上下文智能。*arXiv* 预印本 *arXiv:2606.19348*，2026年。网址 https://arxiv.org/abs/2606.19348。Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet：大型分层图像数据库。发表于*IEEE*计算机视觉与模式识别会议（*CVPR*），第248–255页，2009年。网址 https://doi.org/10.1109/CVPR.2009.5206848。Dong, W., Charikar, M., and Li, K. 面向通用相似性度量的高效 *k*近邻图构建。发表于第*20*届万维网国际会议（*WWW*）论文集，第577–586页，2011年。网址 https://doi.org/10.1145/1963405.1963487。Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazar´e, P.-E., Lomeli, M., Hosseini, L., andJ´egou, H. Faiss 库。*IEEE*大数据汇刊，12(2)：346–361，2026年。网址 https://doi.org/10.1109/TBDATA.2025.3618474。Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. 时间序列数据库中的快速子序列匹配。发表于*1994*年*ACM SIGMOD*数据管理国际会议论文集，第419–429页，1994年。网址 https://doi.org/10.1145/191839.191925。Fortune, S. and Van Wyk, C. J. 面向计算几何的高效精确算术。发表于第九届计算几何年度研讨会（*SoCG*）论文集，第163–172页，1993年。网址 https://doi.org/10.1145/160985.161015。Fu, C., Xiang, C., Wang, C., and Cai, D. 基于导航式扩展图的快速近似最近邻搜索。《*VLDB*基金会会刊》，12(5)：461–474，2019年。网址 https://doi.org/10.14778/3303753.3303754。

Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., Orgad, E., Entezari, R., Daras, G., Pratt, S., Ramanu- jan, V., Bitton, Y., Marathe, K., Mussmann, S., Vencu, R., Cherti, M., Krishna, R., Koh, P. W., Saukh, O., Ratner, A., Song, S., Hajishirzi, H., Farhadi, A., Beaumont, R., Oh, S., Dimakis, A., Jitsev, J., Carmon, Y., Shankar, V., and Schmidt, L. DataComp：探寻新一代多模态数据集。发表于神经信息处理系统进展（*NeurIPS*），2023年。网址 https://doi.org/10.52202/075280-1179。

Gao, J. 和 Long, C. RaBitQ：使用理论误差界量化高维向量，以进行近似最近邻搜索。*ACM* 数据管理会议录，2(3)：1–27，2024年。URL https://doi.org/10.1145/3654970。

Gao, J.、Gou, Y.、Xu, Y.、Yang, Y.、Long, C. 和 Wong, R. C.-W. 欧几里得空间中用于近似最近邻搜索的高维向量实用且渐近最优的量化。*ACM* 数据管理会议录，2025年。URL https://doi.org/10.1145/3725413。

Garcia, V.、Debreuve, E. 和 Barlaud, M. 使用 GPU 进行快速 k 近邻搜索。载于 *2008 IEEE* 计算机学会计算机视觉与模式识别会议研讨会，第 1–6 页，2008。URL https://doi.org/10.1109/cvprw.2008.4563100。

Ge, T., He, K., Ke, Q. 和 Sun, J. 面向近似最近邻搜索的优化乘积量化。载于 *IEEE*计算机视觉与模式识别会议（*CVPR*），第2946–2953页，2013年。URL: https://doi.org/10.1109/CVPR.2013.379。

GLM-5 团队。GLM-5：从氛围编程到智能体工程。 *arXiv*预印本 *arXiv:2602.15763*，2026年。URL: https://arxiv.org/abs/2602.15763。

Gollapudi, S.、Karia, N.、Sivashankar, V.、Krishnaswamy, R.、Begwani, N.、Raz, S.、Lin, Y.、Zhang, Y.、Mahapatro, N.、Srinivasan, P.、Singh, A. 和 Simhadri, H. V. Filtered-DiskANN：带过滤器的近似最近邻搜索图算法。载于 *2023*年*ACM Web*会议论文集（*WWW*），第3406–3416页，2023年。URL: https://doi.org/10.1145/3543507.3583552。

Groh, F., Ruppert, L., Wieschollek, P. 和 Lensch, H.P. A. GGNN：基于图的GPU最近邻搜索。*IEEE*大数据汇刊，9(1):267–279，2023年。网址: https://doi.org/10.1109/tbdata.2022.3161156。

Gui, Y.、Yin, P.、Yan, X.、Zhang, C.、Zhang, W. 和 Cheng, J. PilotANN：面向向量搜索的内存受限的GPU加速。载于第*32*届*ACM SIGKDD*知识发现与数据挖掘会议论文集（第*1*卷），KDD '26，第348–358页。计算机协会，2026年。网址: https://doi.org/10.1145/3770854.3780192。

Guo, R.、Sun, P.、Lindgren, E.、Geng, Q.、Simcha, D.、Chern, F. 和 Kumar, S. 使用各向异性向量量化加速大规模推理。载于国际机器学习会议（*ICML*），《机器学习研究论文集》第 119 卷，第 3887–3896 页，2020 年。网址: https://proceedings.mlr.press/v119/guo20h.html。

Guo, R.、Luan, X.、Xiang, L.、Yan, X.、Yi, X.、Luo, J.、Cheng, Q.、Xu, W.、Luo, J.、Liu, F.、Cao, Z.、Qiao, Y.、Wang, T.、Tang, B. 和 Xie, C. Manu：云原生向量数据库管理系统。《*VLDB*基金会会刊》，15(12)：3548–3561，2022 年。网址：https://doi.org/10.14778/3554821.3554843。

Higham, N.J. 数值算法的精度与稳定性。工业与应用数学学会（SIAM），美国宾夕法尼亚州费城，第二版，2002。doi：10.1137/1.9780898718027。URLhttps://doi.org/10.1137/1.9780898718027。

Hofst¨atter, S., Lin, S.-C., Yang, J.-H., Lin, J., and Hanbury, A. 通过平衡主题感知采样，高效地训练有效的稠密检索器。载于 第*44*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集，第113–122页，2021年。URLhttps://doi.org/10.1145/3404835.3462891。

Horchidan, S., Zeiher, F., Bostr¨om, H., and Carbone, P. Co-nANN：保形近似最近邻搜索。《*VLDB*基金会会刊》，19(1):29–42，2025年。URL https://www.vldb.org/pvldb/vol19/p29-horchidan.pdf。

Hsu, P.-K., Xu, W., Liu, Q., Rosing, T., 和 Yu, S. HAVEN：面向大规模近似最近邻搜索加速的高带宽闪存增强型向量引擎。*arXiv* 预印本 *arXiv:2603.01175*，2026年。网址: https://arxiv.org/abs/2603.01175。

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., 和 Zhang, R. iDistance：一种基于自适应 b+-树的最近邻搜索索引方法。*ACM* 数据库系统汇刊 *(TODS)*，30(2):364–397，2005年。网址 https://doi.org/10.1145/1071610.1071612。

Jegou, H.、Tavenard, R.、Douze, M. 和 Amsaleg, L. 搜索十亿个向量：使用源代码重新排序。载于*2011*年 *IEEE* 国际声学会议，

语音与信号处理（*ICASSP*），第 861–864 页，2011年。网址 https://doi.org/10.1109/ICASSP.2011.5946540。

Johnson, J., Douze, M. 和 Jegou, H. 使用 GPU 进行十亿规模的相似性搜索。*IEEE*大数据汇刊，7 (3):535–547，2021年。网址 https://doi.org/10.1109/TBDATA.2019.2921572。

J´egou, H., Douze, M. 和 Schmid, C. 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。网址 https://doi.org/10.1109/TPAMI.2010.57。

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., 和 Lewis, M. 通过记忆实现泛化：最近邻语言模型。载于 国际学习表征会议 *(ICLR)*，2020 年。网址 https://openreview.net/forum?id=HklBjCEKvH。Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., 和 Stoica, I. 借助 PagedAttention 高效管理大语言模型服务中的内存。载于第 *29* 届操作系统原理研讨会论文集 *(SOSP)*，第 611–626 页，2023年。网址https://doi.org/10.1145/3600006.3613165。Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., K¨uttler, H., Lewis,M., Yih,W., Rockt¨aschel, T., Riedel, S., 和 Kiela, D. 面向知识密集型自然语言处理任务的检索增强生成。载于 神经信息处理系统进展 *(NeurIPS)*，第 33 卷，第 9459–9474 页，2020 年。网址 https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html。Liu, D., Chen, M., Lu, B., Jiang, H., Han, Z., Zhang, Q., Chen, Q., Zhang, C., Ding, B., Zhang, K., Chen, C., Yang, F., Yang, Y., 和 Qiu, L. RetrievalAttention：通过向量检索加速长上下文大语言模型推理。载于 神经信息处理系统进展 *(NeurIPS)*，2025年。网址https://papers.nips.cc/paper_files/paper/2025/hash/4e36d4049fb0fea195a8267c8dcd0824-Abstract-Conference.html。Lowe, D. G. 基于尺度不变关键点的独特图像特征。 国际计算机视觉杂志，60(2):91–110，2004 年。网址 https://doi.org/10.1023/B:VISI.0000029664.99615.94。Mailthody, V. S., Qureshi, Z., Liang, W., Feng, Z., Garcia de Gonzalo, S., Li, Y., Franke, H., Xiong, J., Huang, J., 和 Hwu, W. DeepStore：面向智能查询的存储内加速。载于第 *52* 届 *IEEE/ACM* 国际微体系结构研讨会论文集 *(MICRO)*，第 224–238 页，2019年。网址 https://doi.org/10.1145/3352460.3358320。

Malkov, Y. A. 和 Yashunin, D. A. 使用分层可导航小世界图进行高效且稳健的近似最近邻搜索。*IEEE*模式分析与机器智能汇刊，42(4):824–836，2020 年。网址 https://doi.org/10.1109/TPAMI.2018.2889473。

Muja, M. 和 Lowe, D. G. 面向高维数据的可扩展近邻算法。*IEEE*模式分析和机器智能汇刊，36(11)：2227–2240，2014。网址 https://doi.org/10.1109/tpami.2014.2321376。

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. MS MARCO：人工生成机器阅读理解数据集。载于认知计算研讨会论文集：神经与符号方法融合，*NeurIPS 2016*同期举办，第1773卷 *CEUR*研讨会论文集，2016年。网址：https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf。NVIDIA。NVIDIA H100张量核心GPU架构。白皮书，2022年。网址：https://resources.nvidia.com/en-us-hopper-architecture/nvidia-h100-tensor-c。NVIDIA。CUTLASS：用于线性代数子程序的CUDA模板。GitHub代码仓库，2025年。网址：https://github.com/NVIDIA/cutlass。NVIDIA。cuVS：GPU上的向量搜索与聚类。GitHub代码仓库，2026年。网址：https://github.com/rapidsai/cuvs。版本26.10。Ootomo, H., Naruse, A., Nolet, C., Wang, R., Feher, T., and Wang, Y. CAGRA：面向GPU的高度并行图构建与近似最近邻搜索。载于*IEEE*第*40*届国际数据工程会议（*ICDE*）论文集，第4236–4247页，2024年。网址：https://doi.org/10.1109/ICDE60146.2024.00323。OpenAI。gpt-oss-120b和gpt-oss-20b模型卡片。*arXiv*预印本 *arXiv:2508.10925*，2025年。网址：https://arxiv.org/abs/2508.10925。Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma,

V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. DINOv2：无需监督的鲁棒视觉特征学习。《机器学习研究汇刊》，2024年。网址：https://openreview.net/forum?id=a68SUt6zFt。

Paszke, A., Gross, S., Massa, F., Lerer,A., Bradbury, J.,Chanan, G., Killeen, T., Lin,Z.,Gimelshein, N., Antiga,L.,Desmaison, A., K¨opf, A., Yang, E., DeVito, Z., Rai- son, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., andChintala, S. PyTorch：一种命令式风格的高性能深度学习库。载于神经信息处理系统进展（*NeurIPS*），第 32 卷，2019年。网址：https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html。

Patel, L., Kraft, P., Guestrin, C. 和 Zaharia, M. ACORN：针对向量嵌入和结构化数据的高性能、与谓词无关的搜索。 *ACM* 数据管理会议录，2(3):1–27，2024 年。网址 https://doi.org/10.1145/3654923。Peng, Y., Yang, D., Xie, Z., Sun, J., Shou, L., Chen, K. 和 Chen, G. SVFusion：面向大规模实时向量搜索的中央处理器-GPU 协同处理架构。《*VLDB*基金会会刊》，19(5):1074–1087，2026 年。网址 https://www.vldb.org/pvldb/vol19/p1074-yang.pdf。Qureshi, Z., Mailthody, V. S., Gelado, I., Min, S., Masood, A., Park, J., Xiong, J., Newburn, C. J., Vainbrand, D., Chung, I.-H., Garland, M., Dally, W. 和 Hwu, W. GPU 在 BaM系统架构 中发起的按需高吞吐存储访问。载于 第 *28* 届 *ACM* 编程语言与操作系统架构支持国际会议（*AS-PLOS*），第 *2* 卷，第 325–339 页，2023 年。网址 https://doi.org/10.1145/3575693.3575748。Qwen 团队。Qwen3.8-Flash-Next。Hugging Face 模型仓库，2026 年。网址 https://huggingface.co/Qwen/Qwen3.8-Flash-Next。Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. 和 Sutskever, I. 从自然语言监督中学习可迁移视觉模型。载于国际机器学习会议（*ICML*），第 139 卷，收录于《机器学习研究会议录》，第 8748–8763 页，2021 年。网址 https://proceedings.mlr.press/v139/radford21a.html。Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C. 和 Lillicrap, T. P. 面向长程序列建模的压缩型Transformer。载于国际学习表征（*ICLR*），2020 年。网址 https://openreview.net/forum?id=SylKikSYDH。

Reimers, N. and Gurevych, I. Sentence-BERT：使用孪生 BERT 网络的句子嵌入。载于*2019*年自然语言处理实证方法会议暨第九届自然语言处理国际联合会议论文集（*EMNLP-IJCNLP*），第 3982–3992 页，2019年。doi：10.18653/v1/D19-1410。网址：https://aclanthology.org/D19-1410/。

Ren, J., Zhang, M., and Li, D. HM-ANN：在异构内存上实现十亿级点的高效近邻搜索。发表于 神经信息处理系统进展（*NeurIPS*），第 33 卷，第 10672–10684 页，2020 年。网址：https://proceedings.neurips.cc/paper/2020/hash/788d986905533aba051261497ecffcbb-Abstract.html。Shewchuk, J. R. 自适应精度浮点运算与快速稳健的几何谓词。发表于 离散与计算几何，18(3)：305–363，1997 年。网址：https://doi.org/10.1007/PL00009321。Shi, J., Gao, J., Xia, J., Feh´er, T. B., and Long, C. 基于 GPU 的近似最近邻搜索与 IVF-RaBitQ：快速索引构建与搜索。发表于《*VLDB*基金会会刊》，19(11)：3454–3467，2026年。网址：https://doi.org/10.14778/3836663.3836701。Simhadri, H. V., Williams, G., Aum¨uller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S. J., and Wang, J. NeurIPS'21 十亿规模近似最近邻搜索挑战赛结果。发表于 *NeurIPS 2021* 竞赛与演示专题论文集，第 176 卷，收录于 机器学习研究论文集，第 177–189 页，2022 年。网址：https://proceedings.mlr.press/v176/simhadri22a.html。Singh, A., Subramanya, S. J., Krishnaswamy, R., and Simhadri, H. V. FreshDiskANN：用于流式相似性搜索的快速、准确的基于图的 ANN 索引。*arXiv* 预印本 *arXiv:2105.09613*，2021年。网址：https://arxiv.org/abs/2105.09613。Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. RoFormer：采用旋转位置嵌入的增强型 Transformer。发表于 神经计算，568：127063，2024年。doi：10.1016/j.neucom.2023.127063。网址：https://doi.org/10.1016/j.neucom.2023.127063。Subramanya, S. J., Devvrit, Simhadri, H. V., Krishnaswamy, R., and Kadekodi, R. DiskANN：在单节点上实现快速、准确的十亿级近邻搜索。发表于

神经信息处理系统进展（*NeurIPS*），第 32 卷，2019年。网址：https://papers.nips.cc/paper_files/paper/2019/hash/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Abstract.html。

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., 和 Han, S. Quest：面向高效长上下文大语言模型推理的查询感知稀疏性。载于 国际机器学习会议 *(ICML)*，第 235 卷，第 47901–47911 页，2024年。网址：https://proceedings.mlr.press/v235/tang24l.html。

腾讯混元团队。Hy3。Hugging Face 模型仓库，2026年。网址：https://huggingface.co/tencent/Hy3。

Tian, B., Liu, H., Tang, Y., Xiao, S., Duan, Z., Liao, X., Jin, H., Zhang, X., Zhu, J., and Zhang, Y. 借助 CPU/GPU 协同过滤与重排序，实现高吞吐、低延迟的十亿级向量搜索。载于第 *23* 届 *USENIX*文件与存储技术会议（*FAST 25*），第 171–185 页，2025年。网址：https://www.usenix.org/conference/fast25/presentation/tian-bing。

Tillet, P., Kung, H. T., and Cox, D. Triton：一种用于分块神经网络计算的中间语言与编译器。载于第 *3* 届 *ACM SIGPLAN* 机器学习与编程语言国际研讨会论文集，第 10–19 页，2019年。网址：https://doi.org/10.1145/3315508.3329973。

Venkatasubba, K., Khan, S., Singh, S., Simhadri, H. V., and Vedurada, J. BANG：使用单个GPU进行十亿级近似最近邻搜索。*IEEE*大数据汇刊，11(6):3142–3157，2025年。网址：https://doi.org/10.1109/TBDATA.2025.3581085。

Vidal Ruiz, E. 一种以（近似）恒定平均时间查找最近邻的算法。模式识别快报，4(3):145–157，1986年。网址：https://doi.org/10.1016/0167-8655(86)90013-9。

Vovk, V., Gammerman, A., and Shafer, G.随机世界中的算法学习。Springer，马萨诸塞州波士顿，2005年。doi：10.1007/b106715。网址：https://doi.org/10.1007/b106715。

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., Yu, K., Yuan, Y., Zou, Y., Long, J., Cai, Y., Li, Z., Zhang, Z., Mo, Y., Gu, J., Jiang, R., Wei, Y., and Xie, C. Milvus：专为向量数据管理打造的系统。载于*2021*年 *ACM SIGMOD*数据管理国际会议论文集，第 2614–2627 页，2021年。网址：https://doi.org/10.1145/3448016.3457550。

Wang, M., Xu, W., Yi, X., Wu, S., Peng, Z., Ke, X., Gao, Y., Xu, X., Guo, R., and Xie, C. Starling：一种面向数据段上高维向量相似性搜索的高效 I/O 磁盘驻留图索引框架。*ACM* 数据管理会议录，2(1): 1–27，2024年。网址：https://doi.org/10.1145/3639269。

Weber, R., Schek, H.-J., 和 Blott, S. 高维空间中相似性搜索方法的定量分析与性能研究。载于 第 *24* 届超大规模数据库国际会议论文集（*VLDB*），第 194–205 页，1998 年。网址：http://www.vldb.org/conf/1998/p194.pdf。

Wei, C., Wu, B., Wang, S., Lou, R., Zhan, C., Li, F., and Cai, Y. AnalyticDB-V：面向结构化和非结构化数据查询融合的混合分析引擎。 *VLDB* 基金会论文集，13(12):3152–3165，2020 年。doi: 10.14778/3415478.3415541。URL https://doi.org/10.14778/3415478.3415541。

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. 记忆化 Transformer。载于 国际学习表征会议 *(ICLR)*，2022年。URL https://openreview.net/forum?id=TrjbxzRcnf-。

Xiao, C., Zhang, P., Han, X., Xiao, G., Lin, Y., Zhang, Z., Liu, Z., and Sun, M. InfLLM：面向大语言模型的高效上下文记忆与无需训练的长上下文外推。载于神经信息处理系统进展 *(NeurIPS)*，2024a。URL https://neurips.cc/virtual/2024/poster/94480。

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. 具有注意力汇的高效流式语言模型。载于国际学习表征会议 *(ICLR)*，2024b。URL https://arxiv.org/abs/2309.17453。

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. 通义千问2.5 技术报告。 *arXiv* 预印本 *arXiv:2412.15115*，2024年。URL https://arxiv.org/abs/2412.15115。

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang,

T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. 通义千问3 技术报告。 *arXiv* 预印本 *arXiv:2505.09388*，2025a。URL https://arxiv.org/abs/2505.09388。

Yang, A., Yu, B., Li, C., Liu, D., Huang, F., Huang, H., Jiang, J., Tu, J., Zhang, J., Zhou, J., Lin, J., Dang, K., Yang, K., Yu, L., Li, M., Sun, M., Zhu, Q., Men, R., He, T., Xu, W., Yin, W., Yu, W., Qiu, X., Ren, X., Yang, X., Li, Y., Xu, Z. 和 Zhang, Z. Qwen2.5-1m 技术报告。 *arXiv* 预印本 *arXiv:2501.15383*，2025年b。URLhttps://arxiv.org/abs/2501.15383。

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A. 和 Ceze, L. FlashInfer：面向大语言模型推理服务的高效且可定制的注意力引擎。在机器学习与系统会议论文集（*MLSys*），第 7 卷，2025年。URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/dbf02b21d77409a2db30e56866a8ab3a-Abstract-Conference.html。

Yianilos, P. N. 一般度量空间中的近邻搜索数据结构与算法。载于第四届*ACM-SIAM*离散算法研讨会（*SODA*）论文集，第311–321页，1993年。网址：https://dl.acm.org/doi/10.5555/313559.313789。

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S. 和 Chun, B.-G. Orca：面向基于Transformer的生成模型的分布式服务系统。载于第*16*届*USENIX*操作系统设计与实现研讨会（*OSDI 22*），第521–538页，2022年。网址：https://www.usenix.org/conference/osdi22/presentation/yu。

Zhang, Q., Xu, S., Chen, Q., Sui, G., Xie, J., Cai, Z., Chen, Y., He, Y., Yang, Y., Yang, F., Yang, M. 和 Zhou, L. VBASE：通过宽松单调性统一在线向量相似性搜索与关系查询。载于第*17*届*USENIX*操作系统设计与实现研讨会（*OSDI 23*），第377–395页，2023年。网址：https://www.usenix.org/conference/osdi23/presentation/zhang-qianxi。

Zhang, Z.、Liu, F.、Huang, G.、Liu, X. 和 Jin, X. 通过重排序流水线处理超出 GPU 内存的大型数据集中的快速向量查询。在第 *21* 届 *USENIX*网络系统设计与实现研讨会（*NSDI 24*），第 23–40 页，2024年。网址 https://www.usenix.org/conference/nsdi24/presentation/zhang-zili-pipelining。

Zhao, D. CertEx：通过认证排除法实现大尺度下的精确视觉检索。*arXiv* 预印本 *arXiv:2610.21325*，2026年。网址 https://arxiv.org/abs/2610.21325。

Zhao、W.、Tan、S. 和 Li、P.《SONG：GPU 上的近似最近邻搜索》。载于 *2020 IEEE* 第 *36* 届数据工程国际会议（*ICDE*），第 1033–1044 页，2020 年。网址：https://doi.org/10.1109/icde48307.2020.00094。

经认证的筛选行

本附录中的命题建立在筛选的一项保证之上，第 2.3节对此进行了描述，而第 3.1 节的筛选阶段也依赖于该保证。本节使用 ZARC 的记号重新表述该保证。考虑一个由向量组成的数据集 $\mathbf{x} \in \mathbb{R}^D$、一个中心 $\mathbf{c} \in \mathbb{R}^D$以及一个矩阵 $\mathbf{V} \in \mathbb{R}^{m\times D}$，其各行正交归一；ZARC 将该矩阵计算为数据集样本协方差矩阵的主特征向量。CertEx（Zhao，2026年）定义提升映射

$$\ell(\mathbf{x}) = \big[\mathbf{V}(\mathbf{x}-\mathbf{c});\ \|(\mathbf{I}-\mathbf{V}^\top\mathbf{V})(\mathbf{x}-\mathbf{c})\|\big] \in \mathbb{R}^{m+1}, \quad (1)$$

其平方范数 $a(\mathbf{x}) = \|\ell(\mathbf{x})\|^2$ 等于 $\|\mathbf{x} - \mathbf{c}\{v14>$。因此，每个查询 $\mathbf{q}$ 和向量 $\mathbf{x}$ 都满足 $\|\mathbf{q}-\mathbf{x}\|^2 \geq a(\mathbf{q}) + a(\mathbf{x}) - 2\langle \ell(\mathbf{q})\ \ell(\mathbf{x})\rangle$，因为柯西-施瓦茨不等式将两个残差的内积界定在其范数乘积之内。在本附录中，$U(\mathbf{q})$ 和 $r_k(\mathbf{q})$ 表示距离平方，这不会改变论文中的表述，因为平方运算保持距离的顺序不变。

ZARC 使用 CertEx 的舍入证书（Zhao、2026年），以半精度计算该界。每个存储向量都变为一个筛选行，由 $m+1$ 半精度提升坐标组成，并通过一个公共因子进行缩放 $1/\alpha$，使每个提升后的行的范数至多为 64，随后附加四个证书坐标。其中两个证书坐标存储行范数及其舍入误差的向上舍入界，另外两个存储 $-a(\mathbf{x})/(2\alpha^2)$，表示为两个半精度数之和。查询行中与之匹配的坐标通过一次带 fp32 累加的张量核心乘积 $v(\mathbf{q},\ \mathbf{x})$，结合其误差项，计算缩放后的界。向量存活当且仅当 $v(\mathbf{q},\ \mathbf{x}) \geq \theta(\mathbf{q})$，其中

$$\theta(\mathbf{q}) = \mathrm{down}\Big(\frac{a(\mathbf{q}) - U(\mathbf{q})}{2\alpha^2} - \mu(\mathbf{q})\Big) \quad (2)$$

且裕度 $\mu(\mathbf{q})$ 通过数据集上证书坐标的最大值涵盖剩余误差项。对于幸存项，计算得到的乘积给出经认证的下界

$$\lambda(\mathbf{q},\mathbf{x}) = \big(a(\mathbf{q})/\alpha^2 - 2(v(\mathbf{q},\mathbf{x}) + \epsilon(\mathbf{q}))\big)\alpha^2, \quad (3)$$

其中 $\epsilon(\mathbf{q})$ 汇总了累积误差和裕度。

命题 1（经认证筛选）。 如果认证上界涵盖每个存储行，并且每个乘积都以单精度沿一条顺序链累加其各项，那么每个未通过测试 $v(\mathbf{q},\ \mathbf{x}) \geq \theta(\mathbf{q})$ 的向量都满足 $\|\mathbf{q} - \mathbf{x}\|^2 > U(\mathbf{q})$，而每个幸存项都满足$\lambda(\mathbf{q},\ \mathbf{x}) \leq \|\mathbf{q} - \mathbf{x}\|^2$。

命题 1重述了 CertEx（Zhao，2026 年）的保证，其误差分析遵循浮点点积（Higham，2002）和张量核心运算（Blanchard 等人，2020 年）的标准分析。其累加条件约束了第C.1节的内核。具体而言，误差常数 $\gamma = 16\,L\,u$ with $u = 2^{-24}$ 界定了对 L 个填充半精度数的一行进行的顺序 fp32 累加。因此，每个乘积都会在一条张量核心指令链中累加，因为 ZARC 从不会将跨越整行长度的归约拆分到多个线程块中。

B 阈值、更新和内积的正确性

论文以文字形式陈述了该设计的三项保证，本节将对其进行证明。第一项涉及第 3.2 节的分区阈值，第二项涉及第 3.4 节的插入和删除，第三项涉及第 3.6 节的内积筛选。每个证明仅使用第 A 节的下界和命题1 的证书。

命题 2（分区阈值）。 对于查询 $\mathbf{q}$ 以及任意k 个互不相同的存储向量 $\mathbf{x}_1$、.....、$\mathbf{x}_k$，真实的第 k 近距离满足 $r_k(\mathbf{q}) \leq \max_i \|\mathbf{q} - \mathbf{x}_i\|$。因此，在 $\min(U_0$、$U_1)$ 下进行筛选，绝不会排除 $\mathbf{q}$ 的真实第 k 近邻。

证明。第 k 近距离 $r_k(\mathbf{q})$ 是以 $\mathbf{q}$ 为中心、包含 k 个存储向量的球的最小半径。半径为 $\max_i \|\mathbf{q} - \mathbf{x}_i\|$ 的球包含这 k 个互不相同的向量 $\mathbf{x}1$、.....、$\mathbf{x}_k$，因此 $r_k(\mathbf{q}) \leq \max_i \|\mathbf{q} - \mathbf{x}_i\|$。阈值$U_0$ 是互不相同的有效锚点之间的精确距离中第 k 小的距离，对于 k 个最近锚点而言，它等于上述最大值。同样，U_1 对于 k 个最近的ρk 个候选项而言，也等于该最大值。两个阈值及其最小值因此从上方界定了 $r_k(\mathbf{q})$。于是，每个真实邻居 $\mathbf{x}$ 都满足 $\|\ell(\mathbf{q}) - \ell(\mathbf{x})\| \leq \|\mathbf{q} - \mathbf{x}\| \leq \min(U_0, U_1)$，因为提升距离不会超过真实距离。CertEx（Zhao，2026）的舍入证书会保留所有提升距离不超过该阈值的向量，其中包括真实邻居 $\mathbf{x}$。

□

命题 3（更新正确性不变量）。对于固定的正交归一投影和中心，提升映射后的下界对每个存储向量均成立。如果舍入证书的最大值覆盖每个存储行，则认证测试绝不会排除活跃向量中的真实邻居。因此，使用现有投影进行的插入，以及禁用行和锚点的删除，都能保持精确性。

证明。 CertEx 的提升下界 $\|\ell(\mathbf{q}) - \ell(\mathbf{x})\| \leq \|\mathbf{q} - \mathbf{x}\|$（Zhao，2026年）对每个正交归一投影和每个中心都成立，因为其证明仅使用了将 $\mathbf{q}$–$\mathbf{c}$ 和 $\mathbf{x}$–$\mathbf{c}$ 分解到 $\mathbf{V}$ 的张成空间及其补空间中的正交分解。插入操作使用现有投影和中心对新向量进行提升映射，此时该界同样成立。舍入证书通过证书坐标的数据库最大值界定半精度乘积的误差。由于插入操作会在任何搜索读取这些行之前，将这些最大值提升到覆盖其各行的范围，认证测试便能像在静态数据集上一样，使每个活跃向量都保持在阈值以内。已删除的行会与每个查询的常数坐标组合出最负的半精度值。因此，其乘积低于每个阈值，从而在所有筛选中排除该行。已删除的锚点不再进入阈值，因此根据命题 k- 个活跃向量中的第 $\mathbf{k}$ 小个精确距离，并将 k-个活跃向量中的距离从上界定为命题 2。

□

对于内积查询，第3.6节通过以下方式将每个向量围绕原点进行提升映射：$\ell(\mathbf{x}) = [\mathbf{V}\mathbf{x};\ \|(\mathbf{I} - \mathbf{V}^\top\mathbf{V})\mathbf{x}\|]$ 并使用下界筛选分数 $s(\mathbf{q},\mathbf{x}) = \|\mathbf{q}\|^2 - 2\langle\mathbf{q},\mathbf{x}\rangle$。

$$s(\mathbf{q}, \mathbf{x}) \geq \|\ell(\mathbf{q})\|^2 - 2\langle\ell(\mathbf{q}), \ell(\mathbf{x})\rangle. \tag{4}$$

命题 4 (精确内积筛选)。设任意正交归一的$\mathbf{V}$ 以及任意 k 个互不相同的向量，其 k小分数为 T。任何公式（(4)中的界超过 T 的向量，都不在所有存储向量中与 k 内积最大的$\mathbf{q}$个向量之列。

证明。定义残差 $\mathbf{r_x} := (\mathbf{I} - \mathbf{V}^\top\mathbf{V})\mathbf{x}$，由此得到$\ell(\mathbf{x}) = [\mathbf{V}\mathbf{x};\ \|\mathbf{r_x}\|]$，同理可得 $\ell(\mathbf{q})$。由于 $\mathbf{V}$ 的各行正交归一，$\mathbf{x} = \mathbf{V}^\top\mathbf{V}\mathbf{x} + \mathbf{r_x}$ 将 $\mathbf{x}$ 分解为正交的两部分，由此得到 $\langle\mathbf{q}, \mathbf{x}\rangle = \langle\mathbf{V}\mathbf{q}, \mathbf{V}\mathbf{x}\rangle + \langle\mathbf{r_q}, \mathbf{r_x}\rangle$，以及$\|\mathbf{q}\|^2 = \|\ell(\mathbf{q})\|^2$。柯西-施瓦茨不等式给出$\langle\mathbf{r_q}, \mathbf{r_x}\rangle$ 的上界 $\|\mathbf{r_q}\|\|\mathbf{r_x}\|$，因此有 $\langle\mathbf{q}, \mathbf{x}\rangle \leq \langle\ell(\mathbf{q})\ \ell(\mathbf{x})\rangle$，从而得到公式（4）。所给的 k 个向量表明，数据集上的第 k小分数至多为 T。界超过 T的向量，其分数高于 T，因此不属于分数最小的 k 个向量。这些向量就是内积最大的 k个向量，因为内积分数会随内积增大而降低。

□

第3.6节的内积界不包含任何仅依赖向量的项。因此，筛选行保留其布局和舍入证书，并将 $-a(\mathbf{x})/2$ 的两个坐标设为零。阈值 T来自锚点的精确分数，与欧几里得情形相同。随后，验证以双精度计算 $s(\mathbf{q},\mathbf{x})$ 对幸存项进行排序，并返回内积最大的精确前 k 项。

C 实现细节

C.1 GPU筛选内核

在 Hopper GPU 上（NVIDIA，2022年），第 3.1 节筛选内核的每个线程块都会将 128 个查询行保存在共享内存中，并遍历数据库行的每 G个分块，其中 G 是共享查询分块的线程块数量。网格中每个流式多处理器至多驻留两个线程块，因为再增加一个线程块就会启动第二轮，几乎使遍历时间翻倍。张量内存加速器（TMA）通过三级流水线加载行分块，随后 warpgroup 矩阵乘法指令按照 CUTLASS（NVIDIA，2025年）的布局，将半精度乘积以单精度累加。对这些行的归约在该线程块中通过一个顺序循环执行，从而使每个乘积都采用命题 1所假设的累加顺序。

尾处理阶段的每个线程都会将其累加器片段中的幸存项收集到位掩码中，并通过一次共享内存原子操作，为这些幸存项在其块的输出段中预留空间。因此，各块无需争用全局计数器。对于 128 和 256 个元素的行，ZARC 运行 warp 专用变体：其中，一个生产者 warpgroup 使用 TMA 流式传输行分块，两个或三个消费者 warpgroups 轮流获取这些分块。随后，一个消费者的尾处理阶段会与其他消费者的张量核心计算重叠。对于 64 个元素的行，每个消费者获得的乘积过短，无法隐藏繁重的尾处理阶段，因此协作内核最多为其提供 2^{25} 行。

路由变体接收一个工作项列表，每个工作项将查询的一个分块与来自某个连续分区的行分块配对，而 TMA 可以从任意行偏移加载该分区。经过筛选的搜索也使用同一个路由内核来筛选标签分区。在 A100 等 Ampere GPU 上，ZARC 运行具有相同尾处理阶段的内核：这些内核发出 mma.sync 指令，并使用异步复制加载分块，以替代 TMA。

C.2 从主机内存验证

为验证第 3.3 节，GPU 通过统一虚拟寻址从固定主机内存读取原始向量，因此验证过程无需在设备上暂存行。第 F 节的排序键将查询索引存放在高位，将下界的保序位模式存放在低位，基数排序（Adinets 和 Merrill，2022）可通过一次遍历完成排序。批次的每个临时缓冲区都由流顺序分配器分配，因为普通的设备内存释放会使整个 GPU 同步，并在每个缓冲区处阻塞批次流水线。

在验证的第 r 轮中，每个尚未完成的查询都会选取接下来的 $32 \cdot 2^r$ 个幸存项，其下界低于当前第 k 个精确距离。一次扁平启动会以双精度计算所有这些配对的精确距离，同时每个 warp 有八次行读取在途。第二次内核启动会将每个查询的距离合并到其前 k 个结果中。当查询的窗口停在高于其第 k 个距离的下界处，或到达最后一个幸存项时，该查询即完成。

C.3 更新与接口

对于第 3.4 节中的更新，删除操作会将筛选行中与每个查询行的常数坐标相对应的坐标设为最负的半精度值，从而使该行的乘积低于所有阈值。删除操作还会标记该向量，使其不再作为锚点设定阈值。搜索会在整个批次期间以共享模式持有读写锁，而更新仅在发布新的行数和证书最大值时以独占模式持有该锁。

ZARC （表2）的接口遵循 cuVS 的构建、扩展、移除和搜索调用（NVIDIA，2026年）。搜索操作接收一批查询、邻居数量 k，以及可选的每个查询一个标签或一个位集过滤器。索引按欧氏距离、余弦相似度或内积对向量进行排序。对于稀疏注意力，键缓存会为每个 KV 头保留一个索引，并返回精确的 top-k，其格式可以是 `torch.topk` （Paszke 等人，2019年）的布局，也可以是 top-k 的 FlashInfer 变换（Ye 等人，2025年）所用的 int32 位置。引擎会在构建时根据第 3.5 节的成本模型选择投影维度数量和锚点步长。

D 每分区局部投影

第 3.2 节的局部边界取代了每个分区内的全局提升映射，因此证书必须对每个分区都成立。每个分区 $\mathcal{P}_j$ 都有自己的中心 $\mathbf{c}_j$、自己的正交归一方向 $\mathbf{V}_j$、自己的尺度 α_j以及自己的证书最大值，并据此对其中的行进行编码。搜索会通过批量矩阵乘积为每个分区各提升一次查询。随后，搜索会根据公共的 $U(\mathbf{q})$，结合该分区的尺度和最大值，设置每个查询与分区对的阈值；之后通过一次路由遍历，利用各分区自己的查询行筛查所有分区。

命题 5 （局部投影）。 如果每个分区的行都使用该分区的投影和证书最大值进行编码，并且该分区路由遍历的查询使用相同的投影进行提升映射，那么命题 *1* 对该路由遍历中的每个查询以及该分区中的每一行均成立。

证明。 式（1）的提升界对所有正交 $\mathbf{V}$ 和所有中心都成立，尤其对 $(\mathbf{c}_j, \mathbf{V}_j)$也成立。一对的舍入证书用查询所经过的行的最大值来界定其乘积的误差，而这些最大值就是分区 $\mathcal{P}_j$ 的最大值。因此，命题 1

因此在每个分区内都成立，在每个分区中，所有配对都与同一个 $U(\mathbf{q})$ 相交。 □

锚点会保留全局投影的行，因为锚点阶段会将每个查询与每个锚点进行比较。构建完成后添加的向量同样会保留全局投影。路由遍历覆盖各个划分，普通遍历则处理剩余的行，从而使每个向量都接受一次认证测试。在筛选器下，每个被纳入的邻居仍会作为幸存项保留，因为阈值来自被纳入的锚点。随后，验证会丢弃筛选器拒绝的幸存项，剩余的前-k 项就是精确过滤答案。

E 紧凑型 INT8 筛选行

半精度筛选行每个向量占用 $2(m+5)$ 字节，而在相同宽度下，每行使用一个尺度的 int8 编码仅占其约一半，这扩大了单个 GPU 在第 3.3 节分层放置中能够容纳的数据集规模。每个提升向量都会将其投影坐标 $\mathbf{z}_x = s_x(\mathbf{c}_x + \mathbf{e}_x)$ 存储为 int8 编码 $\mathbf{c}_x$，其尺度为$s_x = \max_j |z_{x,j}|/127$，从而保持 $|e_{x,j}| \le 1/2$。每行还会存储 $a(\mathbf{x})/2$, s_x, $\|\mathbf{e}_x\|$, $\|\mathbf{c}_x\|$，以及单精度的残差范数。张量核心会将 $\langle \mathbf{c}_q, \mathbf{c}_x \rangle$ 精确累加为 int32，并且由于 $127^2 m < 2^{24}$，该数值可以精确转换为单精度。

柯西-施瓦茨不等式对投影坐标乘积的其余项给出界，因此可得

$$\langle \mathbf{z}_q, \mathbf{z}_x \rangle \le s_q s_x \big(\langle \mathbf{c}_q, \mathbf{c}_x \rangle + \|\mathbf{e}_x\| (\|\mathbf{c}_q\| + \|\mathbf{e}_q\|) + \|\mathbf{e}_q\| \|\mathbf{c}_x\| \big). \tag{5}$$

考虑将式 ($\beta(\mathbf{q},\mathbf{x})$ 中的式 (5) 代替 $\langle \mathbf{z}_q, \mathbf{z}_x \rangle$后，通过提升界得到的下界；并对其进行单˜精度评估 β。

命题 *6*（*INT8* 行）。如果筛选过程保留所有满足 $\tilde{\beta} \le U(q) + \eta(q)$ 的数对，其中 $\eta(q) = 64u\big(\sqrt{a(q)} + \sqrt{\max_x a(x)}\big)^2$, ，那么每个被排除的向量都有 $\|q-x\|^2 > U(q)$，而 $\beta - \eta(q)$ 则给出每个幸存项平方距离的下界。

证明。式 (5) 给出 $\langle \mathbf{z}_q, \mathbf{z}_x \rangle$ 的上界，因此$\beta(\mathbf{q}, \mathbf{x}) \le \|\ell(\mathbf{q}) - \ell(\mathbf{x})\|^2 \le \|\mathbf{q} - \mathbf{x}\|^2$。对 β 的单精度评估每项至多进行 16 次舍入运算，各项绝对值之和至多为 $2.6(\|\ell(\mathbf{q})\| + \|\ell(\mathbf{x})\|)^2$。根据这些计数，浮点求和的标准误差分析（Higham, 2002）˜给出 $|\beta - \beta| \le \eta(\mathbf{q})$。因此，被排除的向量满足 ˜有 $\beta > \beta - \eta(\mathbf{q}) > U(\mathbf{q})$，而每个幸存项均满足 $\tilde{\beta} - \eta(\mathbf{q}) \le \beta \le \|\mathbf{q} - \mathbf{x}\|^2$。

□

INT8 行返回与半精度行完全相同的邻居，另外还包括量化所接纳的幸存项。锚点保留半精度行，这些行仅用于选择锚点，其精确距离构成 $U(\mathbf{q})$。由于 INT8 界只会接纳更多幸存项，命题 7 仍然适用，搜索会返回精确的前 k。

F 有序验证

有序验证（第 3.3 节）仅在幸存项仍可能进入前 k 项时读取其原始向量；这使得从主机内存中的读取量受到经认证的下界紧密程度的限制。一个内核为每个幸存项计算式（ λ（$\mathbf{q}$, $\mathbf{x}$）（式（3））的距离，并将其向下舍入为单精度。随后，该内核生成一个键，将查询索引与下界的保序位模式连接起来；一次基数排序据此先按查询、再按下界对幸存项排序。二分搜索找出每个查询的第一个幸存项。该实现以轮次方式，对一个批次中的所有查询执行验证（第C.2 节）。一轮开始时的第 k 个距离绝不会低于顺序验证在同一幸存项处所使用的距离。因此，这些轮次会验证顺序验证所验证的每个幸存项，并返回相同的前 k 项。

命题 7（有序验证）。如果 U（$\mathbf{q}$） $\ge r_k$（$\mathbf{q}$）以及命题 *1*成立，则在第一个下界超过当前 k 个精确距离的幸存项处停止的验证，会返回距离最小的 k 个向量。

证明。每个满足 $\|\mathbf{q} - \mathbf{x}\|^2 \le r_k(\mathbf{q})$ 的向量 $\mathbf{x}$ 都满足$\|\mathbf{q} - \mathbf{x}\|^2 \le U(\mathbf{q})$，因此根据命题 1 成为幸存项。考虑验证在当前第 k 个精确距离 b 处停止的时刻。每个未经验证的幸存项的界都至少等于触发停止的界，而该界超过 b。命题 1 随即表明，它的距离高于 b。因此，未经验证的幸存项不可能替代当前的 k 个向量，因为这些向量的距离至多为 b。所以，当前的 k 个向量是所有幸存项中距离最小的 k 个向量，因而也是所有向量中距离最小的向量。

□

命题 7 将精确距离的数量限定为：在最后一轮开始时，其界低于 k 个距离的幸存项数量。因此，对于在前 32 个幸存项内确定答案的查询，读取的原始行数至多为 32；而界远高于 k 个距离的幸存项只需在排序中使用一个键，无需读取其向量。这一特性解释了为什么第 3.3 节的分层布局能够保持其吞吐量：在 BIGANN-500M（Jegou 等人，2011年）上，每次查询的验证只读取 5,593 个幸存项中的 699 个。

G 多 GPU 共享阈值

数据集超过一块 GPU 的容量时，会被拆分为连续分片，每个设备对应一个索引，搜索分两轮进行（第 3.3 节）。第一轮中，每个分片都会根据自身的锚点计算每个查询的阈值 $U_s(\mathbf{q})$。

第二轮中，每个分片都在所有分片中的最小阈值下进行搜索，并返回自身数据集中位于该阈值范围内的精确k 最近向量，随后由主机按精确距离合并结果。由于每个分片在第二轮中都会跳过自身的阈值阶段，因此两轮搜索的开销只比单 GPU 搜索略高。每个分片针对每个查询只需通过互连传输 k 个标识符与距离对，因此合并开销与分片大小无关。ZARC 并行构建分片索引，每个设备由一个主机线程驱动；在四块 GPU 上，主机合并 10,000 个查询的结果耗时 16.7 毫秒。

命题8 (共享阈值)。 阈值 $\min_s U_s(\mathbf{q})$ 下分片答案的合并结果将返回所有分片中距离最小的 k 个向量。

证明。 每个 $U_s(\mathbf{q})$ 都是数据集中 k个不同向量的第 k 个精确距离，因此 $\min_s U_s(\mathbf{q}) \geq r_k(\mathbf{q})$ 是所有分片中第 k个距离 $r_k(\mathbf{q})$，由命题 2得出。因此，所有分片中 k 个最近向量都位于共享阈值以内，并且属于其所在分片的 k 个最近向量。根据命题 7，每个分片都会返回阈值以内精确的 k 个最近向量。因此，合并这些集合后包含 k 个最近向量。

□

H 成本模型的评估

第 3.5节中的成本模型预测一个批次所需的时间，计算方式如下

$$T(m, s, B) = \alpha_{\text{scr}}\, B\, N_{\text{p}}(m+5) + \alpha_{\text{anc}}\, B\, \frac{N_{\text{p}}}{s} + \alpha_{\text{ver}}\, B\, S(m, s), \tag{6}$$

其中， N_{p} 是探测的分区中的向量数量，$S(m,\ s)$ 是每次查询的预期幸存项数量，而α_{ver} 取决于全精度向量所在的内存层级。

我们在 BIGANN（3.5节）数据集上评估第 Je- gou 等人，2011年）中的成本模型，使用分层布局处理从 1000万到 5 亿个向量的数据集。基于 100万和 400万个向量前缀进行调优时，每种数据集规模耗时 4.9 到 5.2 秒。由于每种候选配置都是精确的，错误选择只会影响吞吐量。

图 12 显示，在这一 50× 数据集规模范围内，模型对每个查询的幸存项数量预测误差在 2.7% 以内（图 12（a）），并且能够从 1亿 到 5亿 个向量中选出实测速度最快的配置（图 12 （b））。选定宽度从 1000 万 和 1 亿 时的 123 缩小到 2 亿 和 5 亿 时的 59；在 2 亿 时，这使每秒查询数达到 11.4k，而宽度 123 时为 10.2k，尽管宽度为 123 的行仍能装入设备内存。在 1000万 时，选定的 16 锚点步长与更快的 32 步长相差不到 4%。选定配置的预测吞吐量比实测值高出 1% 至 11%。

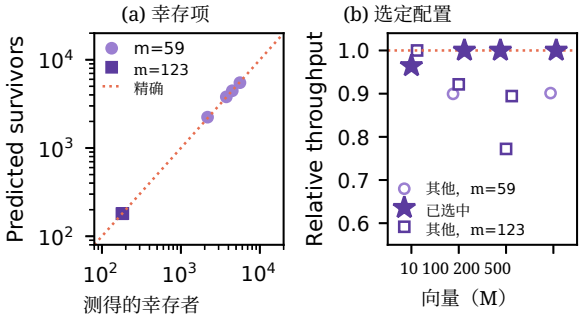

图 *12*。BIGANN 上的成本模型预测与选择。

I 配置效果

CertEx 引擎的测量结果（Zhao，2026年）涵盖 21 种数据集配置下评估的每一种宽度，揭示了静态选择无法捕捉的三个效应，这些效应促使我们设计了第3.5节的调优器。首先，吞吐量关于 m 呈单峰形，并且峰顶较为平坦。低于最佳宽度时，由于较宽松的界会增加幸存项数量，吞吐量最多会下降 5 倍；高于最佳宽度时，吞吐量仅下降 5% 至 35%。一种静态规则选择能够覆盖数据库 95% 方差的最小、且与瓦片对齐的宽度，其吞吐量至少达到最佳吞吐量的 0.87，并在 21 种配置中的 12 种上选出最佳宽度。其次，随着数据集增大，最佳宽度会缩小，因为筛选阶段的规模取决于 N（$m+5$），而每个查询的验证成本只取决于幸存项数量。在 BIGANN（Jegou 等人，2011年）上，宽度 123 在不超过 2000万 个向量时速度最快，而从 5000万 个向量起，宽度 59 的速度最快；仅凭方差的规则无法预测这一点。第三，数据库的方差无法描述来自另一分布的查询。对于 CLIP（Radford 等人，2021年）的文本查询针对 CLIP 图像嵌入时，95% 规则所选宽度使每批次 10,000 个查询产生超过 2×10^8 个幸存项，并导致幸存候选缓冲区溢出，而下一个宽度的速度最快。当某一批次的幸存项超过暂存缓冲区时，系统会将该批次的查询分成两半分别重新执行；即使低估了幸存项数量，也能确保每个答案都是精确的。重新执行的阶段保持完整批次的形状，并将非活动查询的阈值设为无穷大，因为采用新的批次形状会在查询期间重新调优内核，并耗费数秒。

J 过滤搜索和标签分区

过滤搜索会针对每个查询，仅纳入谓词接受的向量。因此，精确过滤答案就是这些获准向量中的精确 top-k，第3.6 节介绍的两种方法都会返回这一结果。第一种方法仅对通过阈值的向量对应用谓词，而锚点阶段已对锚点应用相同的谓词。因此，每个查询的阈值都来自获准向量

本身。命题 2和命题 7结合每个查询的获准向量，即可得出精确过滤答案；其中，获准向量之后的每个条目都以 ID−1 代替邻居。

第二种方法会按标签对筛选行的副本排序，并将副本中的每一行映射到其向量的 ID。带有查询标签的搜索会按标签对每一批查询排序，并将每个查询分块路由到对应标签的行。这个路由内核只筛选这样的分块：其中查询与行具有相同标签；同时，它会屏蔽已加载分块中标签不匹配的行。分区后插入的向量构成一个尾部，由第一种方法进行筛选。尾部中的幸存项随后与路由遍历中的幸存项合并。

命题 9(标签分区)。 在标签分区及其尾部上进行带有查询标签的搜索，会为每个查询返回 k 个距离最小的同标签活跃向量。

证明。 路由遍历会筛选分区中每个查询所属标签的所有行，而第一种方法会在标签谓词下筛选尾部的所有行。因此，每个通过查询认证测试的向量都恰好满足该测试一次。掩码也确保不会保留标签不同的向量对。阈值取自查询所属标签的有效锚点，并根据命题 k界定第 个通过测试的距离2。由于幸存项在验证前会映射到向量 ID，因此，对每个查询所属标签的活跃向量应用命题 3 和 7 即可得出结论。

□

K 精确余弦相似度

余弦相似度（第 3.6 节）需要单独论证，因为 ZARC 返回的是给定向量的精确余弦值，而筛选行提升映射的是经过舍入、因此发生偏差的归一化副本。一对向量的分数为 $s(\mathbf{q},\ \mathbf{x}) = 2 - 2\langle \mathbf{q},\ \mathbf{x}\rangle/(\|\mathbf{q}\|\|\mathbf{x}\|)$，即精确单位向量的平方距离；ZARC 使用存储向量以双精度计算该值。相比之下，筛选行提升映射的是经过 32 位浮点数归一化的副本$\mathbf{q}'$ 和 $\mathbf{x}'$，舍入使它们各自与精确单位向量的距离最多为$\eta_1 = (D + 16)u$。

三角不等式将 $\|\mathbf{q}' - \mathbf{x}'\|$ 与精确单位距离之间的差界定为 $\eta = 2\eta_1$。由于 $\|\mathbf{q}' - \mathbf{x}'\| \le 2 + \eta$，精确分数至少为 $\|\mathbf{q}' - \mathbf{x}'\|^2 - \delta$，其中$\delta = 2\eta(2 + \eta)$。因此，ZARC 在式（2）中将 δ 加到 $U(\mathbf{q})$ 上，并在式（3）中从 $\lambda(\mathbf{q}, \mathbf{x})$ 中减去它。

命题 10 (精确余弦)。 对于松弛量 δ，所有未通过认证测试的向量，其余弦得分都高于 $U(\mathbf{q})$；而每个幸存项都满足 $\lambda(\mathbf{q}, \mathbf{x}) - \delta \le s(\mathbf{q},\mathbf{x})$。

证明。 未通过测试的向量满足 $\|\mathbf{q}' - \mathbf{x}'\|^2 > U(\mathbf{q}) + \delta$根据命题 1 应用于阈值为 $U(\mathbf{q}) + \delta$的归一化副本。其精确分数至少为

模型	层	Dim	注意力头	块	每个注意力头的键
gpt-oss-120b	36 个中的 17 个	64	4	8,192	10.0M
Qwen2.5-7B	28 个中的 14 个	128	4	8,192	250.0M
Qwen2.5-32B	64 个中的 32 个	128	4	8,192	40.0M
Qwen2.5-72B	80 个中的 40 个	128	4	8,192	40.0M
Hy3	80 个中的 20 个	128	4	8,192	10.0M
Qwen3-235B-A22B	94 个中的 47 个	128	4	8,192	10.0M
Qwen3.8-Flash-Next	第 48 个中的 23 个	256	2	8,192	2.5M
DeepSeek-V4-Flash	43 个中的 21 个	512	4	4,096	10.0M
DeepSeek-V3.2	61 个中的 20 个	576	4	4,096	10.0M
GLM-5.3	78 个中的 24 个	576	4	4,096	10.0M
DeepSeek-V4-Pro	61 项中的第 12 项	512	4	2,048	10.0M

表 3。 提取的各个开放式大语言模型的层数和内存。

$\|\mathbf{q}' - \mathbf{x}'\|^2 - \delta$，因此超过 $U(\mathbf{q})$。每个幸存项都满足 $\lambda(\mathbf{q},\ \mathbf{x}) \le \|\mathbf{q}' - \mathbf{x}'\|^2$，根据同一命题，结合精确分数的下界可得 $\lambda(\mathbf{q},\ \mathbf{x}) - \delta \le s(\mathbf{q},\ \mathbf{x})$。

□

命题 10提供了命题 2 和 7 所用的两项性质，即排除高于阈值的项，并为每个幸存项提供下界。由于余弦搜索的阈值来自锚点和候选向量的精确分数，因此这两个命题都适用，只需将平方距离替换为 s。内积情形无需松弛，因为第 3.6 节中的提升映射不会创建归一化副本。

L 开放式大语言模型

表 3 列出了第 4.2 节中的 11 个开放式大语言模型。每份内存都存放一层中经过旋转位置嵌入（RoPE）（Su 等人，2024 年）处理前的键，这些键是在 PG-19 训练书籍（Rae 等人，2020 年）的块上计算得出的，并排除了前四个位置，因为这些位置是注意力汇（Xiao 等人，2024b）。每份内存的查询由从留出测试书籍中抽取的 10,000 个 RoPE 前查询组成；这些查询从被搜索的 KV 头中的查询头均匀抽取。对于能够装入四块 H100 GPU 的层，使用其中间层；对于较早的层，则使用最大的检查点。对于多头潜在注意力（DeepSeek-AI，2024 年），内存存放吸收键 $[\mathbf{c};\ \mathbf{k}_{\text{rope}}]$，所有注意力头共享该键；注意力头 j 的查询为 $[\mathbf{W}_{\text{uk},j}^{\top}\mathbf{q}_{\text{nope},j};\ \mathbf{q}_{\text{rope},j}]$。键与查询的内积等于 RoPE 前注意力 logit。DeepSeek-V4（DeepSeek-AI，2026年）由所有查询头共享一个键头。随后，四个被搜索的注意力头各自从四分之一的查询头中获取查询。Qwen3.8-Flash-Next（Qwen 团队，2026年）的内存存放 250 万个键，因为其线性注意力层在没有融合内核时的运行速度约为每秒 340 个词元；其查询则来自每个 40 个测试块中各 256 个位置。每种对比方法都读取同一份包含每个头 1000 万个键的内存；若内存中的键更少，则读取整份内存。

图 13 将通义千问Qwen2.5-7B（Yang 等人，2024年）扩展到十亿个键，并支持不断增长的内存。每个注意力头有 2.5 亿个键时，四块 H100 GPU 可容纳十亿个键，

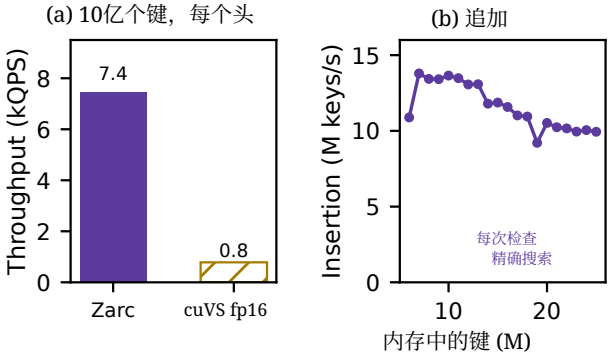

图 *13*。 通义千问2.5-7B，包含十亿个键，并支持追加。

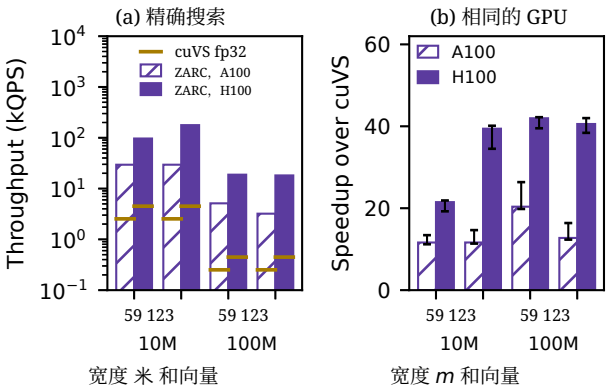

图*14*。 A100 和 H100 GPU 上的精确吞吐量。

在此基础上，ZARC 使用存储在主机内存中的原始键，每个头每秒可回答 7.4 千次精确查询；而采用半精度进行穷尽式 cuVS（NVIDIA，2026年）搜索时，每个头每秒可回答 9。5× （图13（a））。内存以每次增加 100 万个键的步幅，从 500 万增长到 2500 万时，每个头每秒可吸纳 1040 万至 1280 万个键，同时每次经过检查的搜索都保持精确（图13（b））。

M 代 GPU

我们评估了 ZARC 在较早一代 GPU 上是否仍能保持精确性，并继续优于穷举搜索。评估将配备 80 GB HBM2e 内存的 NVIDIA A100 与其他实验（第 4.1 节）中的 H100 进行比较，使用 BIGANN-10M（Jegou 等人，2011年）和 BIGANN-100M 数据集，向量宽度为 59 和 123，批次包含 512 至 8,192 个查询。A100 实验在第三台服务器上运行，该服务器配有两颗 64 核 AMD EPYC 7763 处理器、503 GB 主机 DRAM，以及一块 NVIDIA A100 80 GB PCIe GPU；该 GPU 以 7g.80gb 配置的一个 MIG 实例运行，使用其 108 个流式多处理器中的 98 个。此服务器运行相同的驱动程序和软件栈，但该库使用 CUDA 12.9 和 CUTLASS 4.1 编译（NVIDIA，2025年）。两款 GPU 使用相同的行和阈值，因为舍入证书对乘积的 fp32 累加误差给出了界，而无需假设求和顺序。在 H100 上，ZARC 在此批次范围内，针对 BIGANN-10M 每秒可处理 86.6k 至 98.5k 个查询（$m = 59$，每秒 155k 至 181k 个查询（$m = 123$；相比之下，穷举式 cuVS 搜索每秒为 4.5k 个查询（NVIDIA，2026年）。图 14将 A100 的结果与这些数值进行了比较。

在 A100 上，对于两种规模、两种宽度以及全部五种批次大小，每次查询都能得到精确结果；所有配置的每次查询幸存项数量都与 H100 上相同。图 14(a) 显示，A100 在 BIGANN-10M（Jegou 等人，2011年）上每秒可处理 2.84 万至 3.72 万次查询，在 BIGANN-100M 上每秒可处理 3100 至 6600 次查询；相比之下，同一 GPU 上穷举式 cuVS（NVIDIA，2026年）搜索的吞吐量分别为 2500 次和 251 次。因而，ZARC 在 BIGANN-10M 上比穷举搜索快 11×至 15×倍，在 BIGANN-100M 上快 12× 至 26× 倍（图 14(b)），而 H100 上的相应数值为 19× 至 42× 倍。

H100。A100 上的加速比较小，因为在 BIGANN-10M 上，筛选阶段占批次耗时的 90%；此外，其 Ampere 内核会发出 mma.sync 指令，所达到的张量核心吞吐量峰值占比低于 H100 的 TMA 和 WGMMA 内核。出于同样的原因，在 A100 上处理 1亿个向量时，较窄的宽度 59 更快，每秒可处理 5000 至 6600 次查询；而在 H100 上，两种宽度的吞吐量相同。

N 评估详情

数据集。表4总结了第4.1节中的数据集及其各自服务的工作负载。评估涵盖局部图像描述符、视觉基础模型嵌入和网络规模多模态嵌入。SIFT1M 和 BIGANN（Jegou 等，2011）包含采用欧氏距离的 128 维 SIFT 描述符（Lowe，2004）。BIGANN 中从 1000万到 10 亿个向量的各个前缀均提供官方精确真实值。DINOv2-1.28M 包含由 DINOv2（Oquab 等，2024）ViT-B编码器对 ImageNet-1k（Deng 等，2009）的训练图像生成的 768 维嵌入。DataComp-12.8M 包含 DataComp（Gadre 等人，2023）小型数据池中的 1279 万个 768 维 CLIP（Radford 等人，2021）ViT-L/14 图像嵌入，这些嵌入已归一化为单位长度；在这些嵌入上，欧氏距离与余弦相似度对邻居的排序相同。过滤搜索会以 10%、1% 和 0.1% 的选择率为 BIGANN-10M 分配随机标签。MS MARCO（Nguyen 等人，2016）包含 884 万个文本段落，我们使用两个模型对其进行嵌入。Sentence-BERT（Reimers & Gurevych，2019）模型 msmarco-distilbert-cos-v5 会生成用于余弦检索的单位向量，而 TAS-B（Hofst¨atter 等人，2021）会生成用于内积检索的未归一化向量。Deep1B（Babenko 与 Lempitsky，2016）包含 10 亿个 96 维 float32 图像描述符、10000 个查询及官方真实值；其 2.5 亿和 5 亿个向量的前缀，其精确真实值由双精度暴力搜索得出。

数据集	向量维度		度量指标	工作负载
SIFT1M	1,000,000	128	L_2	局部边界
DINOv2-1.28M	1,281,167	768	L_2	局部边界
BIGANN	1000万至10亿	128	L_2	层级、过滤器、更新、调优
DataComp-12.8M	12,790,000	768	余弦	10近邻图
Deep1B	1,000,000,000	96	L_2	多GPU扩展
MS MARCO	8,841,823	768	余弦、内积	检索、内积

表*4.* 评估的数据集和工作负载。

输入	度量	精确查询
高斯向量	L2, IP, cos	100%
秩 4 数据	L2	100%
每个向量的 50 个副本	L2, IP	100%
$\{0, 1, 2\}$中的坐标	L2, IP	100%
相差一个32位浮点数末位单位的成对数据	L2	100%
按 10^{-6}缩放的高斯	L2、余弦	100%
按 10^{4} 缩放的高斯	L2、内积	100%
范数 10^{-3} 和 10^{3} 混合	L2, IP	100%
高斯，10% 已删除	L2	100%

表*5.* 对抗性输入下的精确查询。

Deep1B 的真实值。精确答案还揭示了 Deep1B 官方真实值中的三处错误（Babenko 与 Lempitsky，2016年）。在 10000 个查询中，有三个查询的 ZARC 答案与官方真实值不同。对全部 10 亿个向量进行独立的双精度暴力搜索，证实了这三种情况下 ZARC 给出的答案。数据集中有两个查询分别存在 16863 个和 23515 个距离为零的精确重复项，但官方列表各自只保留了六个。第三个查询的官方第 10 个邻居比 ZARC 返回的向量距离更远。因此，Deep1B 的结果使用了修正这三行后的官方真实值，这表明即使向量规模达到 10 亿，基准测试真实值本身也需要通过精确搜索来确定。

对抗性输入下的精确性

我们测试 ZARC 在专门构造用于破坏浮点数界限的输入上的精确性，第4.1节对此进行了概述。表5 列出了九种输入，每种包含一百万个 128 维向量，涵盖从精确并列和重复项，到低于半精度正常范围下限以及远高于其上限的数值；同时列出了每种输入所针对的度量：欧氏距离（L2）、内积（IP）和余弦相似度（cos）。对于每种输入和度量，我们使用 $k = 10$ 和$k = 100$ 进行 1,000 次查询，并与双精度下的穷举搜索进行比较；当查询 k 返回的 ID 互不相同，且其双精度距离等于 k 最小距离时，该查询即计为精确。

对于 k 的两种取值，以及输入、度量指标和 k 的全部 32 种组合，每个查询在每个输入上都是精确的。这些输入涵盖了浮点数搜索可能失败的两个环节。在第 k 个距离处出现的并列和重复项，用于测试双精度验证能否确定原本一半精度无法做到，而经缩放的混合范数测试

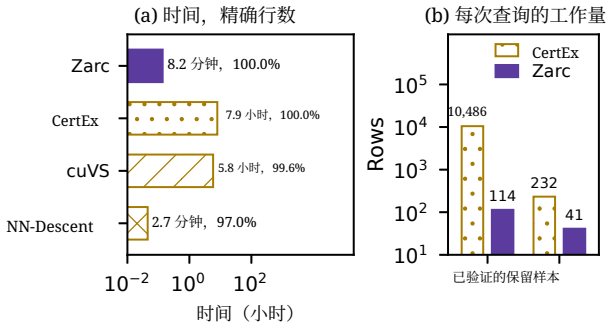

图*15.* DataComp-12.8M 的精确10近邻图。

每个筛选行的舍入误差界是否仍然覆盖其乘积的误差。

P 语义去重

图15详细介绍了第4.7节的去重工作负载，包括 DataComp-12.8M 的 10近邻图（Gadre 等人，2023年）中各种方法的端到端时间和精确行数，以及 ZARC 和 CertEx 引擎（Zhao，2026年）的幸存项数量和每个查询的已验证行数。ZARC 用时 8.2 分钟，42× 比穷举式 cuVS（NVIDIA，2026年）搜索预计的 5.8 小时快得多；而 NN-Descent（Dong 等人，2011年会使 3.0% 的行不精确（图15（a））。分区阈值将每个查询的幸存项从 10,486 个减少到 114 个，其中验证会读取 41 行（图15（b））。