Custom 8-bit floating point value format for reducing shared memory bank conflict in approximate nearest neighbor search

Hiroyuki Ootomo[1], Akira Naruse[2]

[1]Tokyo Institute of Technology [2]NVIDIA

Approximate Nearest Neighbor Search (ANNS)

- (Exact) nearest neighbor search is computationally enormous.
- ANNS is used in information retrieval, machine learning, etc.
  - The search spaces are getting larger in recent years.
  - There are some ANNS libraries for many-core CPUs and GPUs.
- Several methods are proposed: Graph-based, **IVFPQ**, etc.

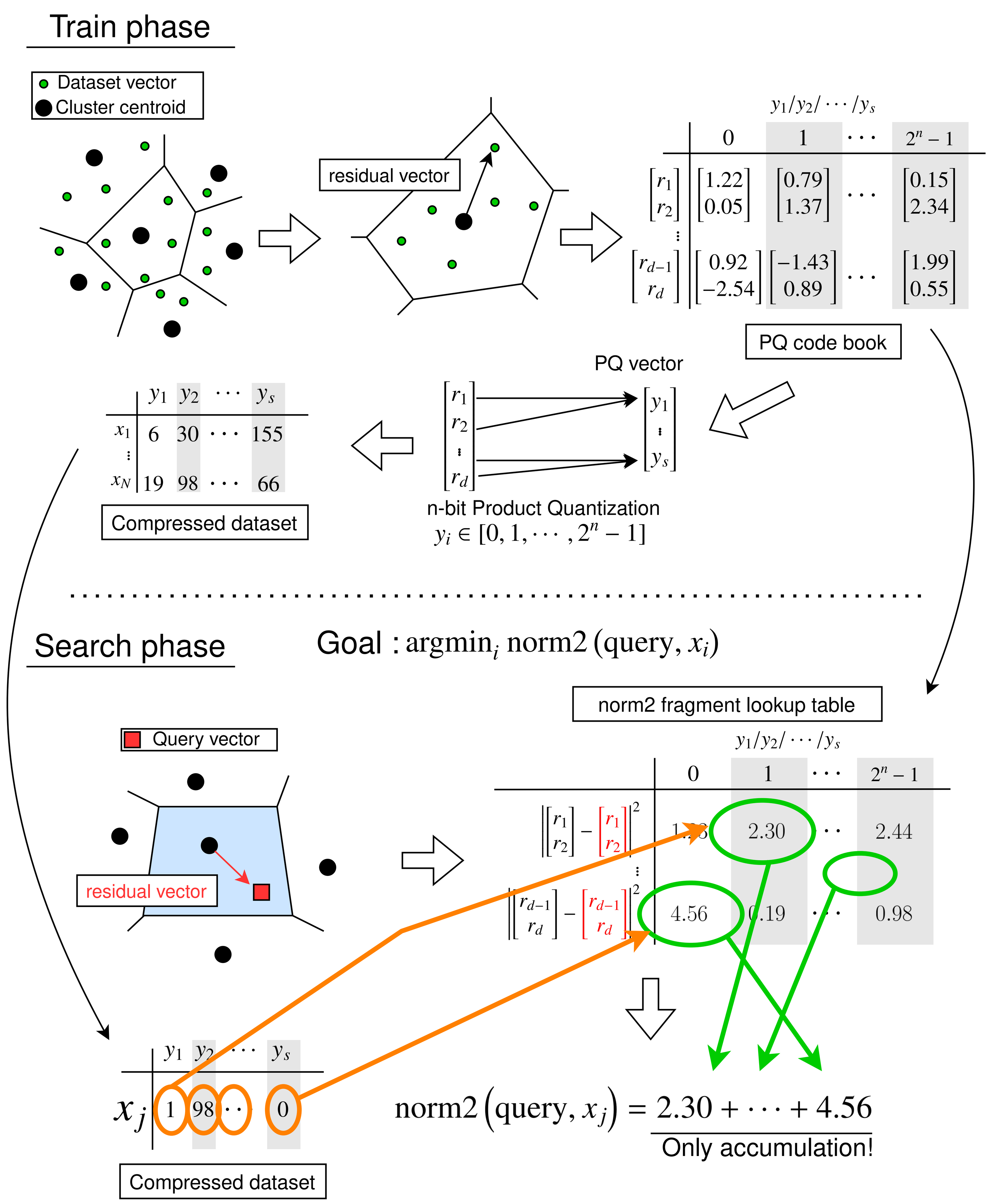
Goal

For a given a d-dimensional query vector $\mathbf{q}$, obtain k vectors $\mathbf{V} = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \cdots, \mathbf{x}_{i_k}\}$ from dataset vectors $\mathbf{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ where

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

IVFPQ

- Can compress the dataset.
  - $\Rightarrow$ Can treat a large dataset on a single device.
- Can compute the norm between a query vector and a dataset vector only by addition instructions once a norm2 fragment lookup table is made.
- Algorithm



IVFPQ search on GPU

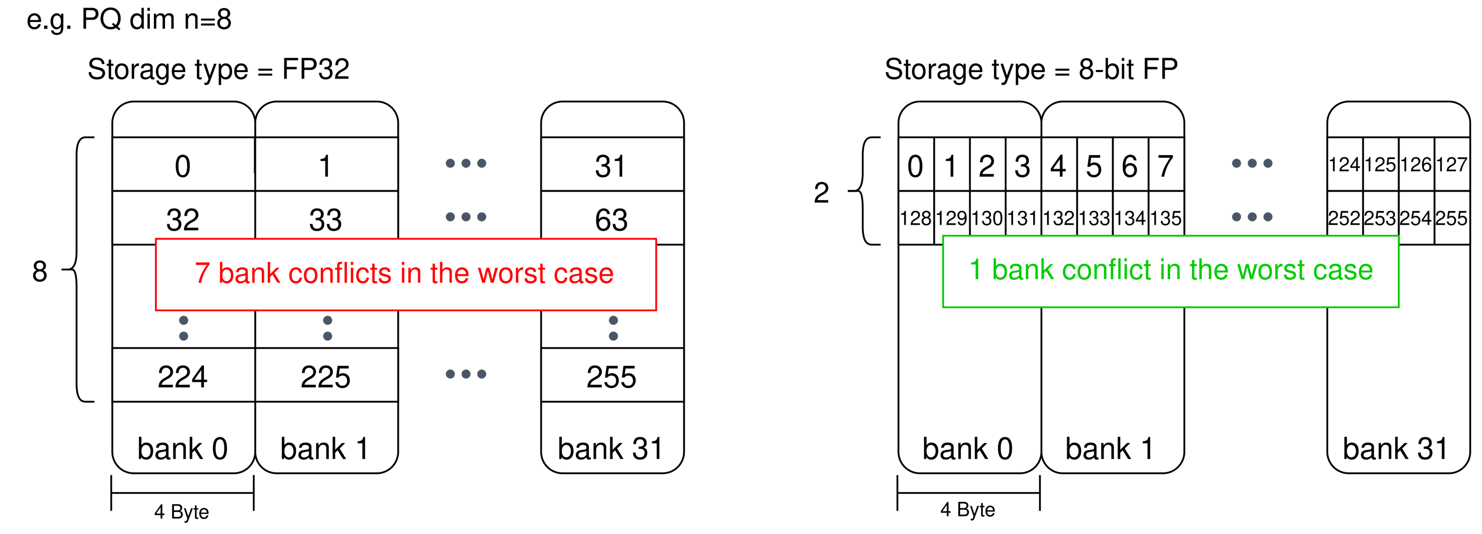
- The norm2 fragment lookup table is stored in shared memory.
- To compute the norm between a query vector and a dataset vector, we load the norm2 fragments from the table and accumulate them.

Problem

- Loading a norm2 fragment from shared memory is a random access through 32 threads in a warp.
- $\Rightarrow$ **Shared memory bank conflict occurs and the throughput degrades.**

Goal: Reducing shared memory bank conflict

Reduce shared memory bank conflict using low-bit floating point representation of the norm fragments

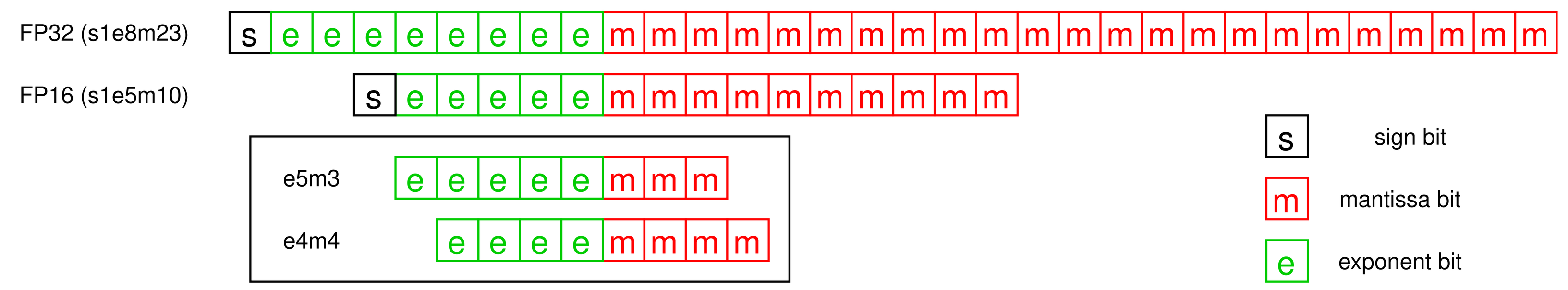


Custom 8-bit floating point format for norm representation

Requirements

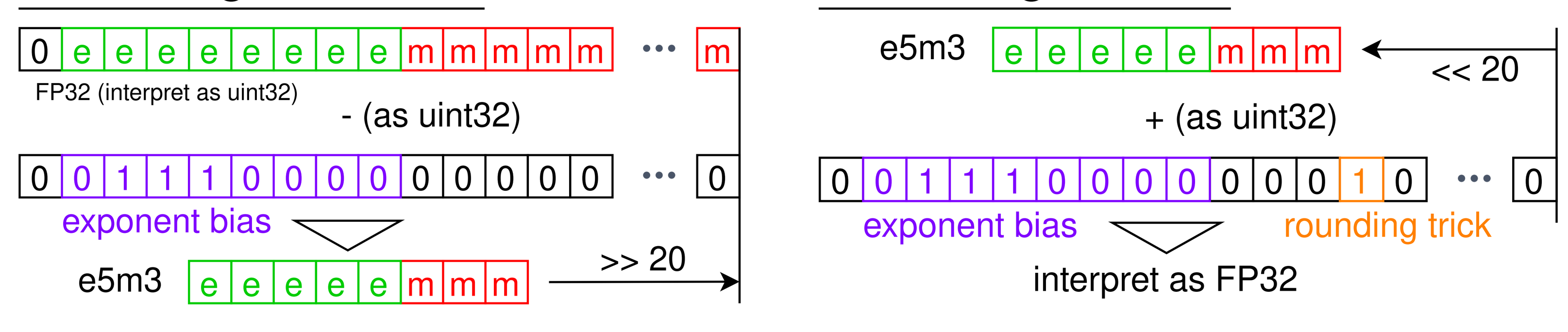
- Only for data storage. No computation between the formats.
- No sign bit is needed since L2-norm is always positive.
- Low overhead for converting from/to FP32.

Proposed format: e5m3 and e4m4



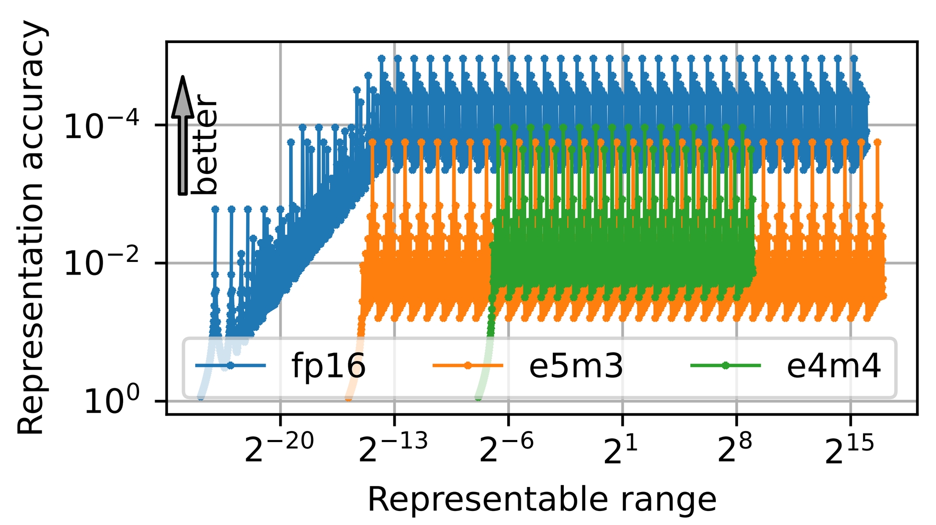
Converting from FP32

Converting to FP32



(*) Check underflow before converting

Representation range and accuracy



While the representation accuracy of e4m4 is not significantly better than e5m3, representable range is only half.
$\Rightarrow$ We use e5m3.

The accuracy of ANNS

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

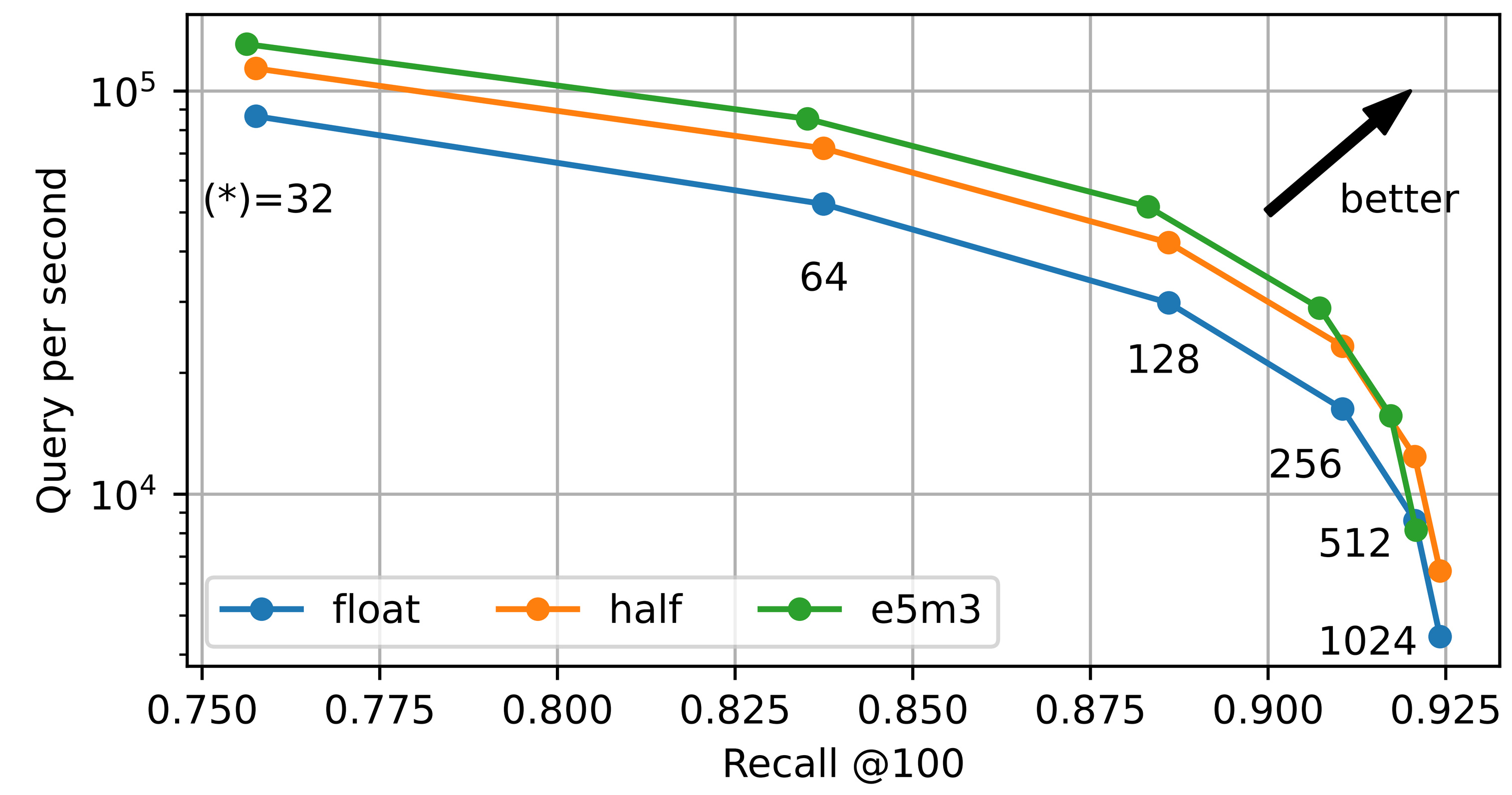
where $\mathbf{V}_{\text{GT}}$ is the set of the ground truth for the query.

Environment

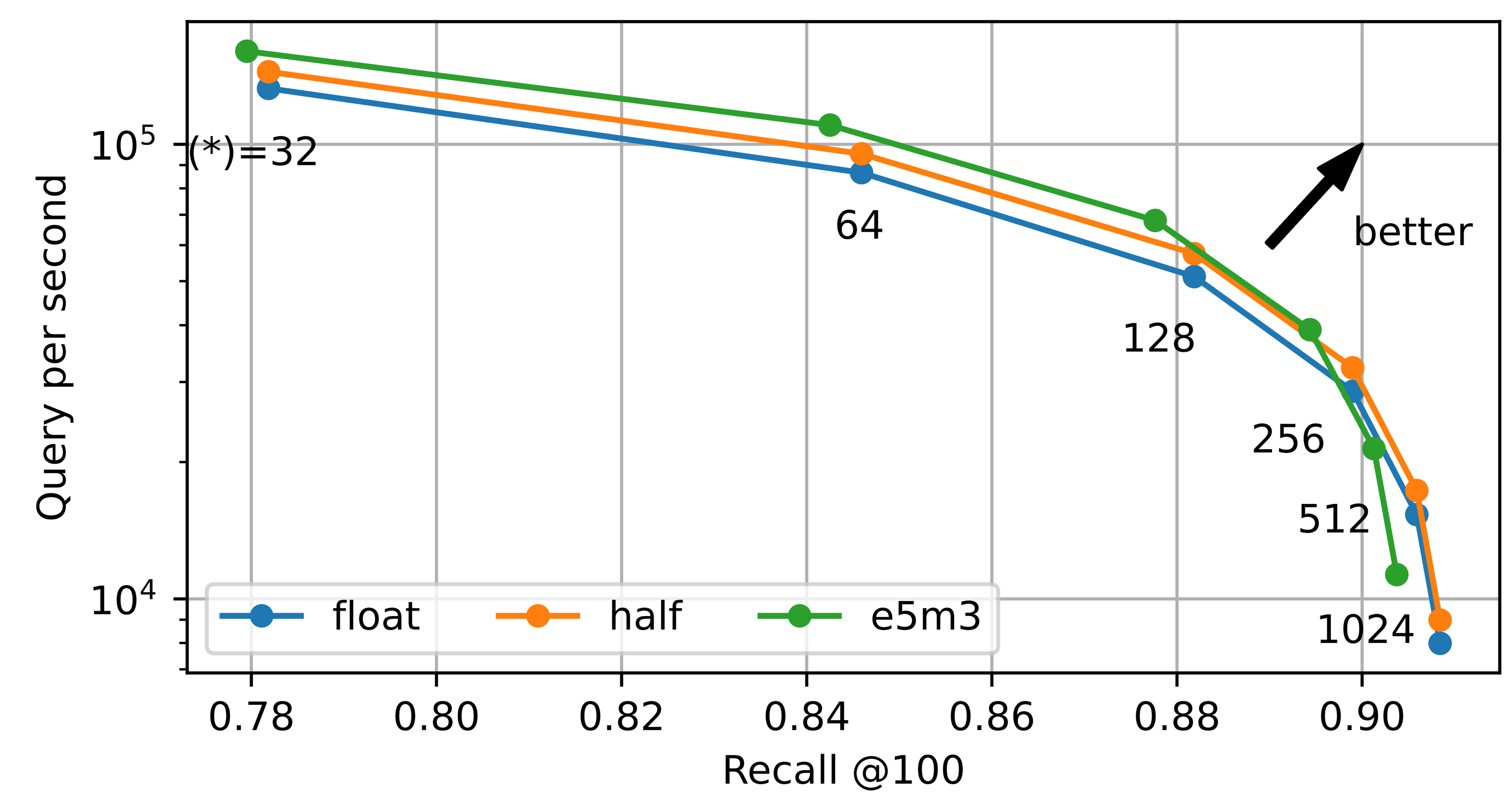
- AMD Ryzen 9 5900X
- NVIDIA GeForce RTX 3080Ti
- CUDA 11.7

Evaluation

- BIGANN 100M dataset ($d = 128$, $s = 64$, data_type = `uint8`)



- Yandex DEEP 100M dataset ($d = 96$, $s = 48$, data_type = `float`)



Parameters

- (*) `num_probes` : The number of clusters picked up in the first stage of the search phase.
- num clusters : 100,000
- batch size : 10,000
- PQ bit : $n = 8$

Conclusion

- We have developed custom 8-bit floating point formats for reducing bank conflict in IVFPQ on GPU.
  - The sign bit is omitted.
  - It can be converted from/to FP32 with a few operations.
- We have applied it to IVFPQ and improved the throughput with a little recall degradation.