Gemini® APU: Enabling High Performance Billion-Scale Similarity Search

www.gsitechnology.com

Introduction

Similarity search plays a key role in many applications, such as e-commerce, drug discovery, Natural Language Processing (NLP), and visual search.

One of the key challenges in the era of big data is managing similarity search in applications where databases scale to 1B items and beyond, doing so with low latency and cost-effectiveness. Low latency is critical for many online applications.

In this paper, we introduce the Gemini® Associative Processing Unit (APU) and present its role in an Approximate Nearest Neighbor (ANN) similarity search pipeline. Latency and recall numbers for query-by-query ANN searches using the DEEP1B dataset will be provided. DEEP1B consists of 1 billion, 96-dimensional vectors with each dimension being a 32-bit floating point (FP32) number. The Gemini APU can efficiently handle either batch mode or query-by-query requests, but in this paper we present query-by-query results rather that batch mode numbers because in many real-world online applications, requests arrive one by one.

The Gemini® APU Similarity Search Engine

The Gemini APU complements existing similarity search hardware, such as the CPU, by offloading a significant portion of the similarity search pipeline. The Gemini APU performs vector management and partitions the database across multiple cards in a distributed architecture. The result is a similarity search solution that scales to databases of billions of items.

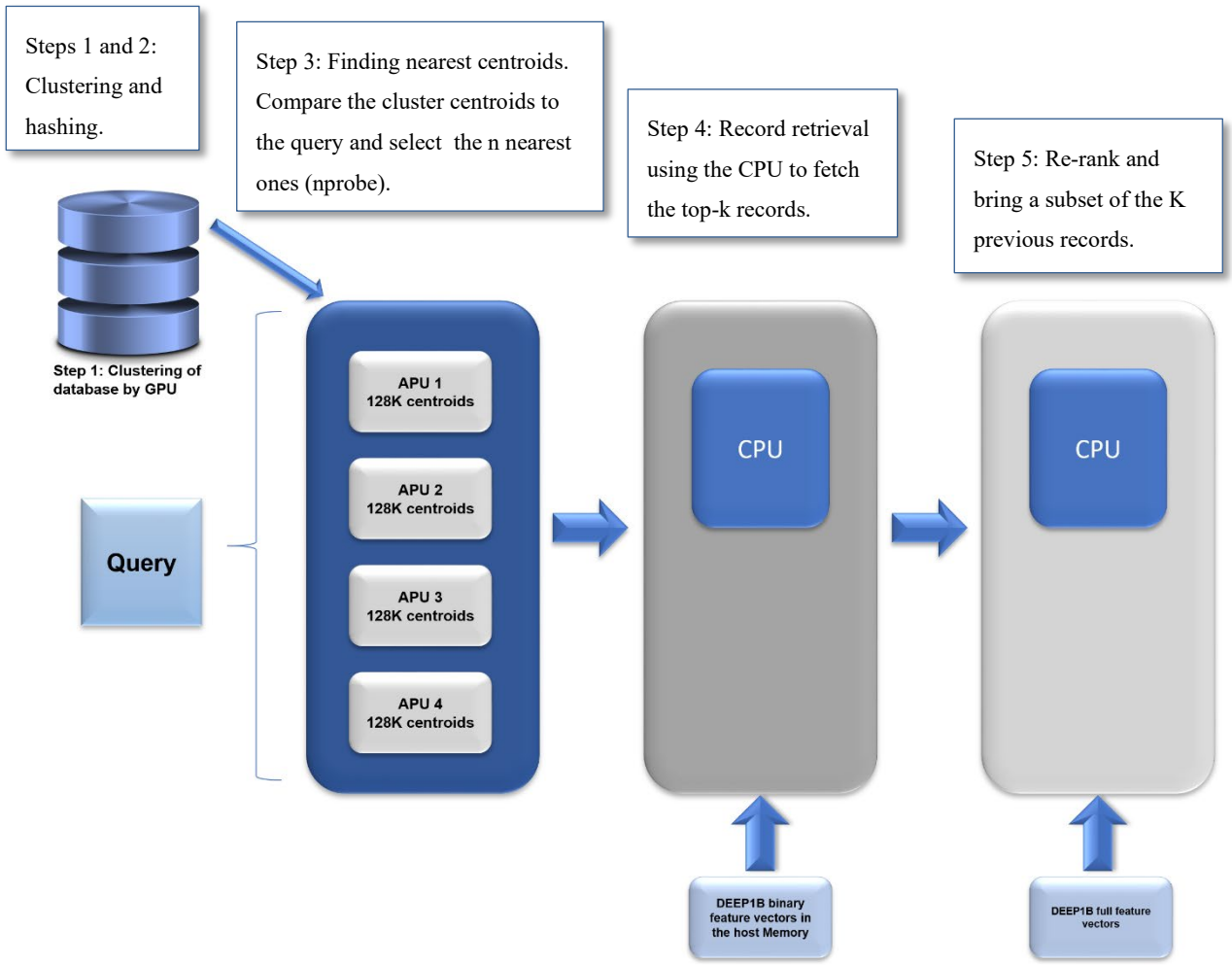
To efficiently perform similarity search at scale, the Gemini APU leverages techniques such as database clustering and data compression in such a way that yields low latency and high quality results (in terms of recall).

Locality Sensitive Hashing (LSH) is used to compress the data. LSH hashes similar data points into the same hash buckets with high probability. High dimensional vectors are compressed into vectors that are much smaller and require much less computation to calculate relative distances.

Overview of the Approximate Nearest Neighbor Search Pipeline

We used the FAISS LSH algorithm to convert the Deep1B dataset from floating point feature vectors to binary feature vectors. The 96-dimensional feature vectors (each dimension being a FP32 number) were converted to binary feature vectors of length = 768 bits. The Gemini APU can very efficiently perform nearest neighbor searches on binary feature vectors by computing the hamming distance between them.

Next, the binary database is clustered into 512K clusters with an average of 2000 records per cluster. The result is a database memory size of 96GB for 1 billion records (1B records x 768 bits) and a centroid memory size 49MB for the 512K clusters (512K records x 768 bits).



1. In the first step, we cluster the database records and compute the centroid of each cluster. This is performed offline and uses known methods such as K-means. The number of clusters (or the expected average cluster size) is predetermined. This step is typically performed using a GPU.

2. In the second step, we convert the centroid feature vectors into binary vectors using LSH. The binary centroid vectors are then loaded into Gemini APU's L1 buffer(s).

3. The third step is performed on the Gemini APU and takes less than 100 microseconds.

   This step compares the cluster centroids to the query and selects the n nearest centroids. The number n, or nprobe, is chosen in advance. We found the optimum nprobe to be 192. It represents the best number for recall while maintaining low latency. Recall is improved with larger n values, but with a tradeoff of greater latency.

   128K centroids (768 bits each) can be loaded into one Gemini APU. Therefore, 4 Gemini APU boards were used to support 512K centroids. The Gemini APU searches directly, in-place (no I/O required), allowing it to perform this step in less than 100 microseconds.

   The values for nprobe and the number of bits representing each centroid are programmable in the Gemini APU. We found nprobe = 192 and centroids of 768 bits to be optimal and thus chose those values. The output of this step is a pool of 384K (192 nprobe x 2000 candidates per nprobe) top candidates to be the nearest neighbors.

4. In the fourth step, we search for nearest neighbors to the query. As mentioned previously, the search in this step is not performed on the full database but rather on the n clusters (nprobe), whose centroids were determined to be the nearest neighbors in **Step 2** above. The number of records that the Gemini APU returns from the nprobe selected clusters is a small fraction of the entire database. This results in a significant reduction in the number of items to be searched, which provides a tremendous offload for the CPU.

This step uses a hashed representation for the dataset, such as LSH. Hashing can have a slight effect on the ordering (ranking) of the results, but this can be offset by selecting not a single record that is nearest to the query with respect to the hashed representation, but a set of the k nearest neighbors to the query.

5. To address this slight effect on ordering mentioned in **Step 4** above, the set members from **Step 4** and the query are compared with their original representations (non-hashed, full representation) and distance metric to produce the approximate nearest neighbor. This step is called re-ranking or refinement.

DEEP1B Performance Results

1B DB Records:

Hardware profile: A 1U server containing 4 Gemini APU boards running at a rate of 400 MHz. nprobe = 192 with centroid / records nbit = 768.

CPU profile: 2 Intel Xeon Gold 5115 - 2.40 GHz Deca-Core Processors.

We use recall@R, which is defined as the fraction of queries where the actual nearest neighbor is ranked in the first R results. For example, recall@160 checks to see if the actual nearest neighbor is within the first 160 results returned.

Here are the tests results for the Deep1B dataset:

- Recall@160 = 92.2%,
- Queries per second (QPS): 800
- Latency: 1.25 ms

Scaling to 40 Billion Items

Microsoft recently presented performance results for a database of 40 billion items.
The table below shows a comparison between a system using Gemini APU, and the Microsoft solution.

Features	Microsoft	Gemini APU
DB size	40B	40B
Number of servers	166	40
QPS	1200–1800	1600
Latency (ms)	5–8	1.25

The Gemini APU solution compares favorably to the state-of-the-art Microsoft solution. As can be seen in the table, GSI Gemini provides lower latency and does so using a smaller footprint. Additionally, based on power numbers published for industry-standard servers, the Gemini APU solution is anticipated to provide a 6x power improvement when compared with the Microsoft solution.

Conclusion

Gemini APU serves as the cornerstone of an ANN similarity search solution that addresses the billion-scale problem and does so with the lowest latency, even for the difficult query-by-query task. It also provides high-quality results with the smallest system footprint and lowest power usage.