**Task Overview**

**Task:** Build an approximate K-NN Graph for a set of vectors.

**Input/Output:**
- **Inputs:** dataset contains 10M 100 dimension vector data.
- **Output:** 100-nearest neighbors for each vector in given dataset.

**Performance Metric:**

\[
Recall = \frac{\text{number of true top 100 nearest neighbors}}{100}
\]

**Solution Overview**

- **Random Initialization**
- **Local Join**
- **Update**

**NN-Descent Framework**

- **Random Initialization:** randomly initialize the neighbor lists \(N[v]\) of each node \(v\).
- **Local Join:** for each node \(v\), and \(p, q \in N[v]\), update \(N[p]\) and \(N[q]\) based on the similarity between \(p\) and \(q\) if one of them is new to the \(N[v]\).
- **Update:** update the information of \(N[v]\) to determine which neighbors are new to \(N[v]\).

**Distance Computation by SIMD**

- **Dim** = 100

\[
Y \quad 16 \text{ floats} \quad \cdots \quad 16 \text{ floats} \quad 4 \text{ floats}
\]

\[
RX \quad 6 \times 16 \text{ floats} \quad 4 \text{ floats}
\]

\[
RY \quad 16 \text{ floats} \quad \cdots \quad 16 \text{ floats} \quad 4 \text{ floats}
\]

- 512 bits SIMD Register
- Process 16 floats at once

**Heap v.s. Sorted Array**

**Observation:**

- In the first few iterations, heap is a nice data structure to represent neighbor lists due to frequent update operations.
- After several iterations, the update of neighbor lists becomes less frequent. On the contrary, many update attempts fail since there they have been in the neighbor list.
- In that case, using sorted array can be more efficient than heap due to less traverse cost as well as low insertion cost.

- **Less traverse cost for duplicate value insertion**

- **Less traverse cost for new value insertion**

- **Low Insertion memory copy cost**

**Experimental Evaluation**

- **Experiment Environment:** Intel(R) Xeon(R) Gold 5318Y CPU @ 2.10GHz and 512GB memory.
- The experiment is conducted on the released dataset of the contest, which consists of 10M float vectors of 100 dimensions.
- Optimized NN-Descent is 3 times faster than original one.

- **Plot:**
  - \(Recall\) vs. \(Time (s)\)
  - Optimized vs. NN-Descent

**References**
