Final Project report

Group Members

110062313 陳楷寰
110062317 陳柏丞
110062311 林奕安
110030014 蔡茗鈞

Implementation

1. Index implement

IVFIndex 的record schema分成兩種，一種是用於儲存cluster centroid以及其對應的cluster_id, 共會有NUM_CLUSTERS個record.
另一種是用於儲存cluster底下的sift record block和id

```
private static Schema centroid_schema(SearchKeyType keyType){
    Schema sch = new Schema();
    sch.addField("centroid", keyType.get(0));
    sch.addField("cluster_id", INTEGER);
    return sch;
}

private static Schema index_schema(SearchKeyType keyType) {
    Schema sch = new Schema();
    sch.addField(SCHEMA_RID_BLOCK, BIGINT);
    sch.addField(SCHEMA_RID_ID, INTEGER);
    return sch;
}
```

每次要搜尋index, insert, delete，會先呼叫beforefirst。這裡的beforefirst 的參數是Distancefn，用於取代SearchRange。他會先透過findNearestNeighbor()，搜尋centroid.tbl來取得與query vector最近的centroid，最後再開啟該cluster的index file

```
public void beforeFirst(DistanceFn target) {
    close();

    int cluster = findNearestCluster(target);
    String tblname = ii.indexName() + cluster;
```

```
        TableInfo ti = new TableInfo(tblname, index_schema(keyType));
        this.rf = ti.open(tx, false);

        // initialize the file header if needed
        if (rf.fileSize() == 0)
            RecordFile.formatFileHeader(ti.fileName(), tx);
        rf.beforeFirst();

        isBeforeFirsted = true;
    }
```

next()只要呼叫rf.next()即可，因為該cluster下的record都是我們的搜尋範圍

```
@Override
    public boolean next() {
        if (!isBeforeFirsted)
            throw new IllegalStateException("You must call beforeFirst()
before iterating index '"
                    + ii.indexName() + "'");

        if(rf.next())return true;
        else return false;
    }
```

2. Minheap Plan/Scan

在找到cluster之後，還必須找出前k筆資料，不同於使用SortPlan/Scan，我們使用minheap的資料結構，存取前k近的record，可以使複雜度從原本的O(nlogn)，降到O(nlogk)，其中n為該cluster的record數，k為minheap的容量

```
private void populateHeap() {
        src.beforeFirst();
        while (src.next()) {
            double distance =
embfld.distance((VectorConstant)src.getVal(embfld.fieldName()));
            HashMap<String, Constant> fldVals = new HashMap<>();
            for (String fldName : fieldnames) {
                fldVals.put(fldName, src.getVal(fldName));
            }
            if (heap.size() < k) {
                heap.offer(new Neighbor(distance, fldVals));
            } else if (distance < heap.peek().distance) {
                heap.poll();
                heap.offer(new Neighbor(distance, fldVals));
            }
        }
```

```
        src.close();
        if (heap.size() != k) {
          throw new IllegalStateException("Heap is not correctly populated
with k elements");
      }
    }
```

3. SIMD implement

EuclideanFn的distance()我們使用了SIMD指令提供了array的平行運算
其中將VectorSpecies<Float> SPECIES設定FloatVector.SPECIES_PREFERRED;
java會根據系統選擇最適合的平行度

```
@Override
    protected double calculateDistance(VectorConstant vector) {
        VectorSpecies<Float> SPECIES = FloatVector.SPECIES_PREFERRED;
        int i = 0;
        float sum = 0.0f;
        int dim = vector.dimension();

        for (; i < SPECIES.loopBound(dim); i += SPECIES.length()) {
            FloatVector va = FloatVector.fromArray(SPECIES, query.vec, i);
            FloatVector vb = FloatVector.fromArray(SPECIES, vector.vec, i);
            FloatVector vc = va.sub(vb);
            FloatVector vd = vc.mul(vc);
            sum += vd.reduceLanes(VectorOperators.ADD);
        }

        /*
        for (; i < dim; i++) {
            double diff = query.get(i) - vector.get(i);
            sum += diff * diff;
        }*/
        return Math.sqrt(sum);
    }
```

4. K-means implement (pseudocode)

1. Select K data points as initial centroids randomly
2. Repeat until the number of iterations reaches the specified count or the centroids no longer change:
   a. Assign each data point to the closest centroid to form clusters
      - For each data point, calculate its distance to all centroids
      - Assign the data point to the cluster of the nearest centroid

b. Recalculate the centroids for each cluster
- Set the new centroid to the average position of all data points belonging to that cluster

3. Return the final cluster assignments and centroid locations

---

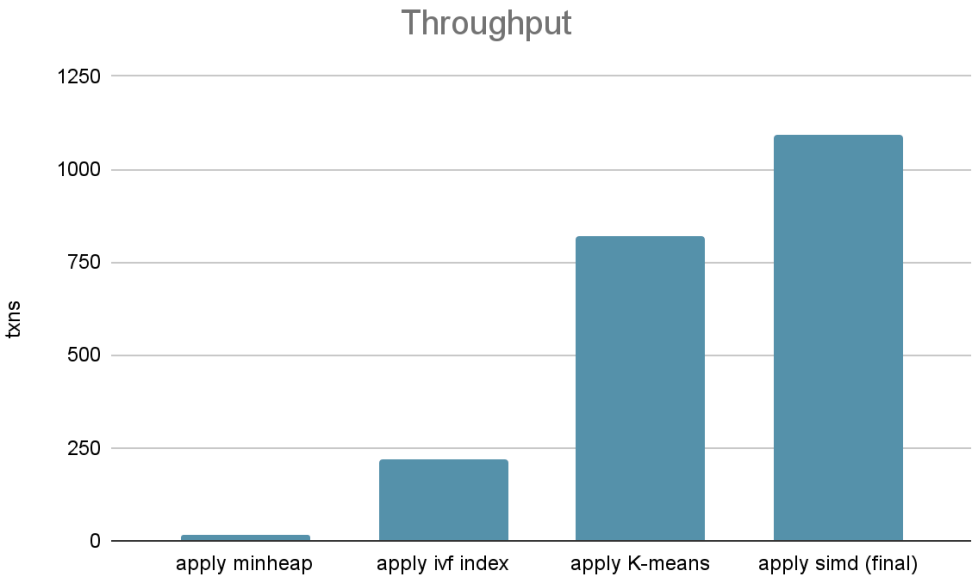
Experiment

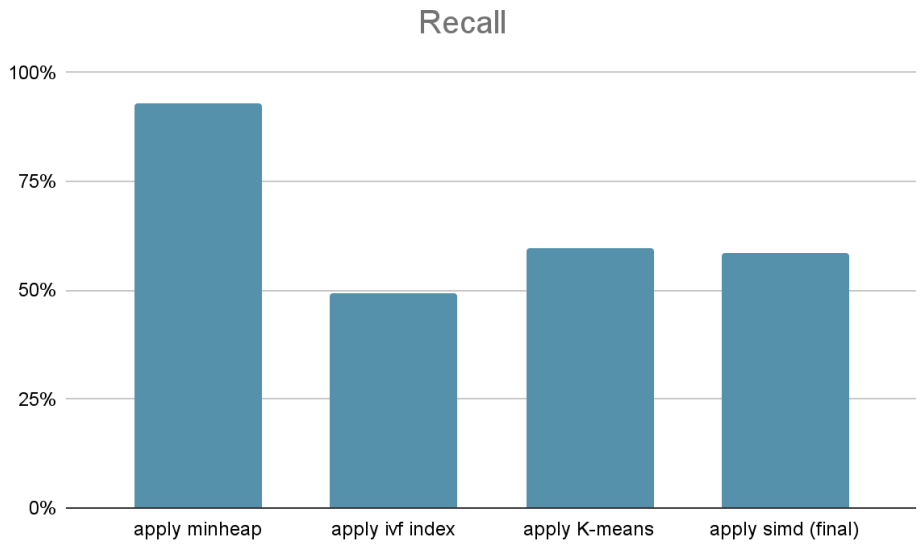
compare throughput / recall under different parameters

- **Experiment environment:**
  environment 1: Apple M1, 16GB RAM, 1TB SSD, macOS sonoma 14.2.1
  environment 2: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 128 GB SSDWindows 11

- **Overall improvement**
  （NUM_CLUSTER = 100 / K-means max_iteration = 20）
  （testing by environment 1）

	original	apply minheap	apply ivf index	apply K-means	apply simd(final)
throughput	**NaN**	**18**	**221**	**818**	**1091**
latency	**NaN**	**6530ms**	**528 ms**	**73 ms**	**106ms**
recall	**NaN**	**93%**	**49.25%**	**59.77%**	**58.53 %**

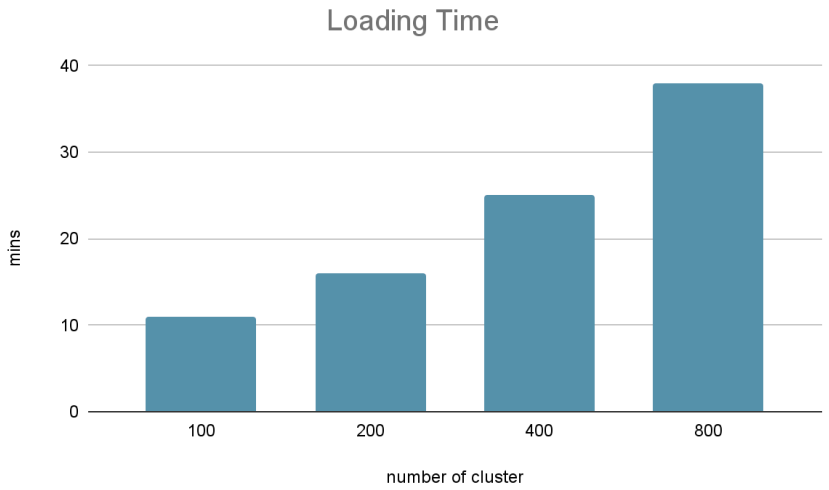




在實驗中，我們發現未經任何優化的原始程式會因為執行時間過久而沒有結果產生，因此我們下面會以對index初步優化的版本（apply minheap的版本）作為基準去進行比較。

可以發現在每個版本中 throughput 的表現上，呈現上升趨勢，說明我們確實有成功進行優化，而在 latency 的表現上，也大致呈現降低的趨勢，唯在 apply simd 的版本中我們實驗結果發現其 latency 有小幅度的增加，我們猜測原因與 JIT 本身的自動優化有關，因為 JIT 在對陣列進行操作時，便已經會將其重新編寫來達到 simd 的效果，因此在加上我們實作的 simd 後反而是會降低性能的。

- **Number of clusters**
  （K-means max_iteration = 20）
  （testing by environment 2）



	100	200	400	800
throughput	**450**	**988**	**930**	**920**
latency	**323 ms**	**116 ms**	**137 ms**	**153 ms**
recall	**51%**	**53.3 %**	**57%**	**60%**

經過測試我們發現，提升 cluster 數量 recall 值也會變得更高，然而增加 cluster 數量，會讓loading time變得更久，上面柱狀圖可以看到當cluster數量提升到800時，時間已經超過30分鐘，因此為了保險起見，我們最後將cluster數量值設在200。

- **Number of K means iteration**
  （NUM_CLUSTER = 100）
  （testing by environment 1）

	20	50	100
throughput	**1091**	**1043**	**884**
latency	**106 ms**	**98ms**	**132ms**
recall	**58.53 %**	**57.66%**	**58.74.%**

經過測試我們發現, K-means 中 max_iteration 的值並不太會影響整體 recall 的表現, 推測是 K-means 在這次project的data中, 只需要大約20次迭代便能接近收斂, 因此更多的iteration數量並不會有太大幫助, 甚至可能拉長整體時間。

- ## Result analysis

經過我們多次的實驗後, 我們發現cluster的數量會影響整體的效能表現, 而 K-means iteration的數量並不會有太大的影響, 因此我們選擇 **NUM_CLUSTER = 200** ， **K-means max_iteration = 20** 作為最後的參數設定。

另外, 我們查閱資料及相關研究後發現。K-means 結果的好壞其實不單取決於收斂與否, 很大程度是受到一開始的初始值決定, 然而我們本次的實作是隨機取出幾個作為初始的 centroids, 因此可能無法讓 K-means 演算法得到最好的發揮。