Project Presentation

20220848 선민수

Minsu Sun

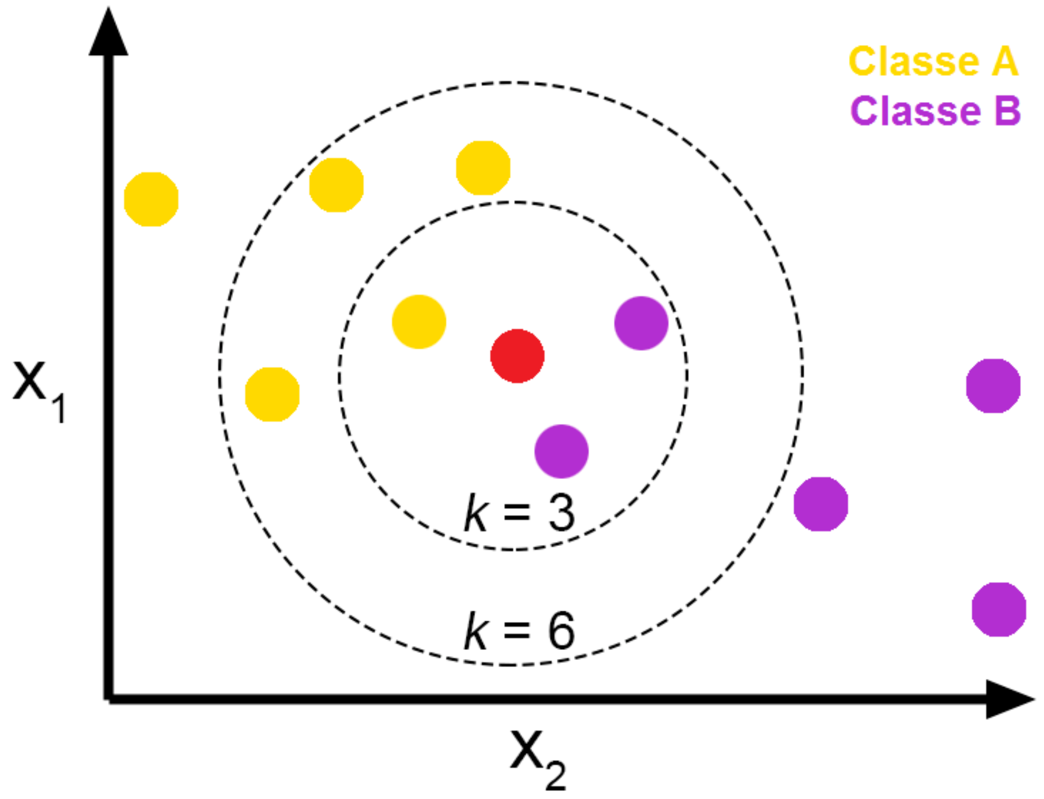
(minus.sun@postech.ac.kr)

PBBS(Problem Based Benchmark Suite) v2

- Outlined in ACM SIGPLAN Symposium on Principles&Parctice of Parallel Programming (PPoPP), 2022
- Collection of over 20 benchmarks defined in terms of their IO characteristics
  - Basic Building Block(SORT, HIST, ISORT, DDUP)
  - Graph Algorithms(BFS, MIS, MM, MSF, SF)
  - Text Processing(BWD, IIDX, LRS, SA, WC)
  - Computational Geometry/Graphics(CH, DR, DT, KNN, RAY, RQ)
  - Others(CLAS, NBODY)
- https://github.com/cmuparlay/pbbsbench

Problem Definition: KNN

- KNN(K-Nearest Neighbors)
- Find K nearest neighbors of all individual n points in d-dimensional space
- Often used in classifier or regression tasks in machine learning



Overview

Step 1: Calculating Distance between points

Step 2: Sorting points by distance(+ get k nearest points)

Strategy: Distance

- Naïve: calculate distance of all pairs of points in serial
- Strategy: calculate distance in the manner of matrix multiplication with tiling in parallel

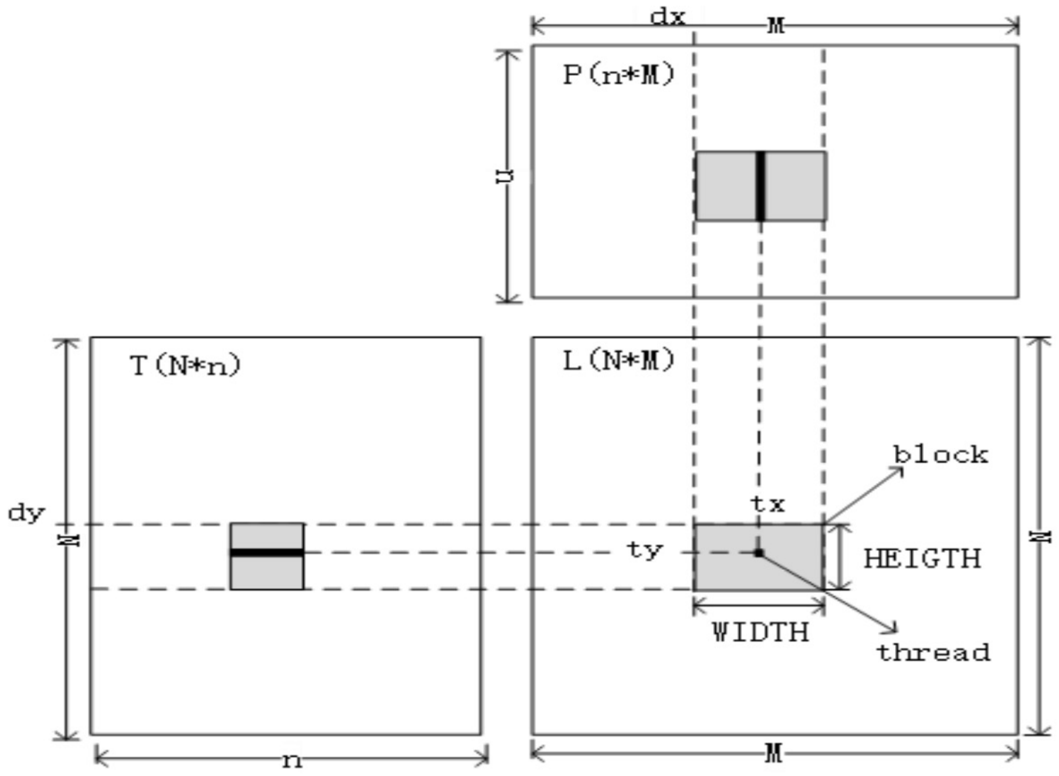
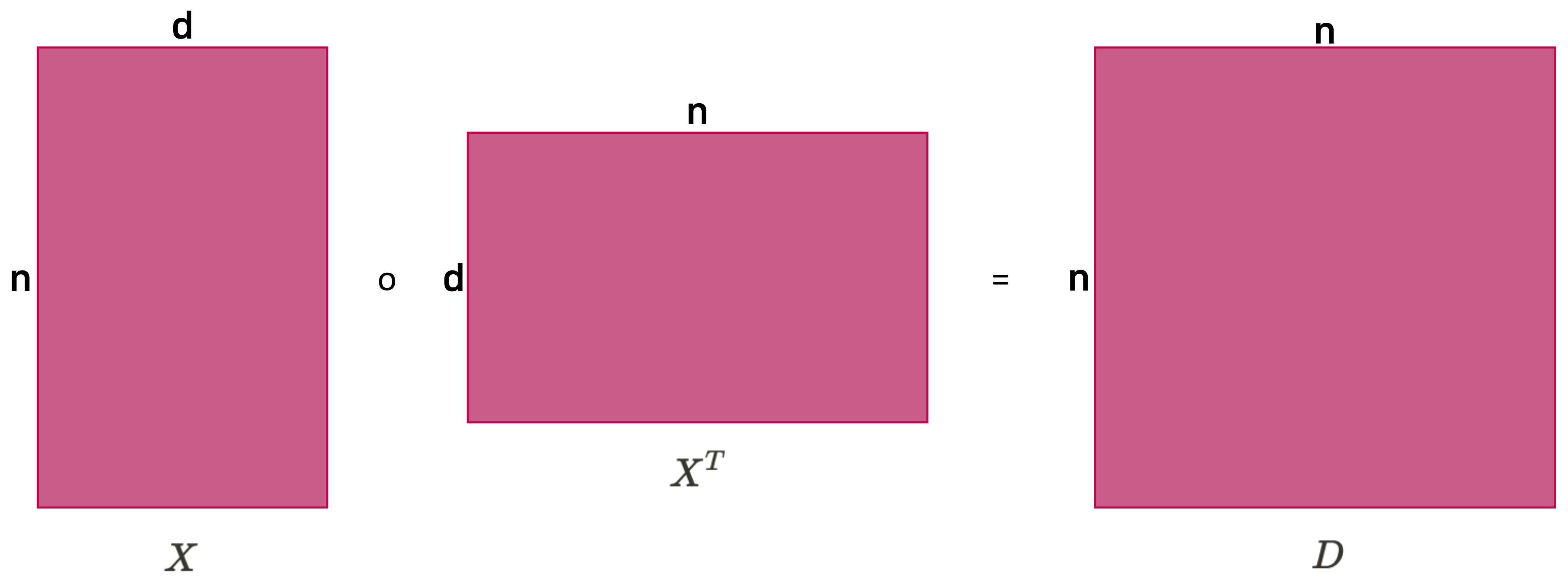


Figure 2 Matrix calculation model

Strategy: Distance



given n points in d-dimensional space as X

$$X \in \mathbb{R}^{n \text{ x } d}$$

$$dist(X) = dist(X, X^T) = D \in \mathbb{R}^{n \text{ x } n}$$

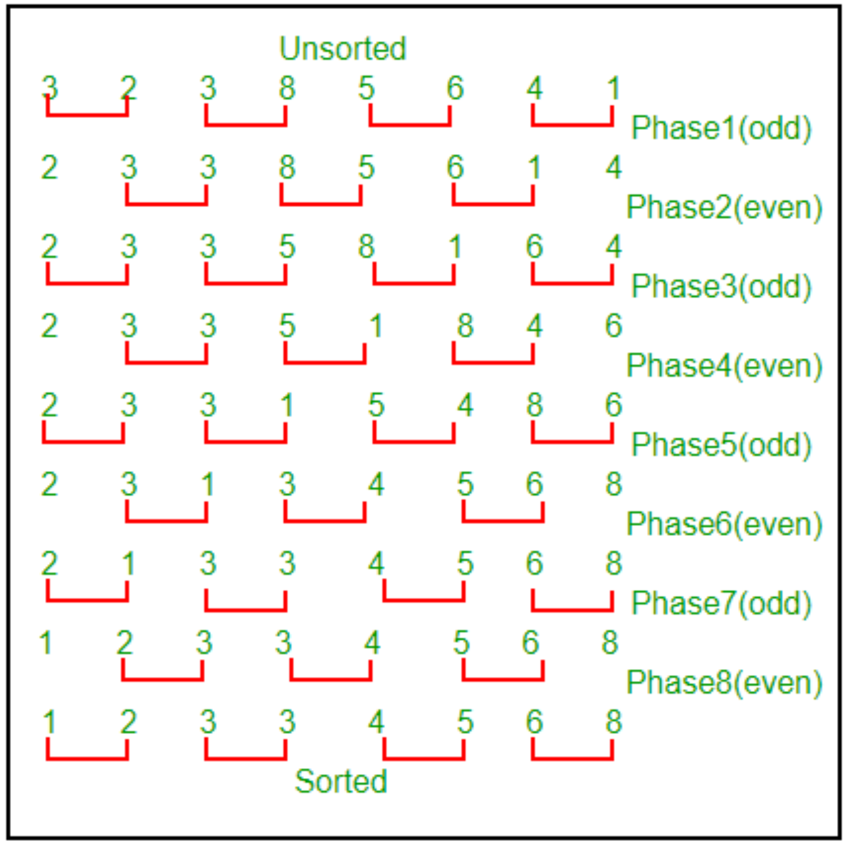
$$D_{i,j} = (\text{distance between point i and point j}) = \sqrt{\sum_{k=0}^{d-1} (X_{i,k} - X^T_{k,j})^2}$$

Strategy: Sort

- Naïve: serial O(nlogn) sort (e.g. quick sort, merge sort …)
- Strategy: use odd-even transposition sort in parallel

Strategy: Sort

- Variation of Bubble Sort
- n phases for data size n
- Serial: O(n^2)
- Parallel: O(n)



< Odd-Even Transposition Sort >

Experiment

Baseline: naïve CPU based implemented KNN

- Distance: naïve calculation in serial
- Sorting: quick sort in serial

Experiment #1: improved CUDA based implemented KNN

- Distance: matrix tiling based calculation in parallel
- Sorting: Odd-Even transposition sorting in parallel

Experiment

Test

- 2D points in Cube, n=1M, d=2, k=1, rounds=3
- 2D points in Kuzmin Distribution, n=1M, d=2, k=1, rounds=3
- 3D points in Cube, n=1M, d=3, k=1, rounds =3
- 3D points on Sphere, n=1M, d=3, k=1, rounds = 3
- 3D points in Cube, n=1M, d=3, k=10, rounds = 3
- 3D points in Plummer Distribution, n=1M, d=3, k=10, rounds=3

Progress

- Implement data loader and run script from PBBS benchmark data
- Implement baseline(naïve version)
- Working on implementing CUDA version

QnA