Indian Institute of Information Technology, Sri City Computer Science and Engineering

High Performance Computing
(HPC-S2023)

Course project

“PARALLELIZING KNN ALGORITHM”

MS DHEERAJ/S20200010128/ 4
D.STEVEN CHRIS/S20200010054/ 4
M.KRISHNA VAMSHI/S20200010138/ 4
M.BANNU DEEPAK/S20200010122/ 4

Abstract:

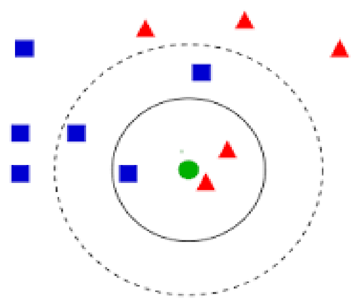
Machine learning has evolved deeper into our lives. k nearest neighbors is one such algorithm which is the simplest machine learning algorthim for classification problems. K nearest neighbors algorithim has so much uses because of its simplicity and high accuracies. one of the main drawbacks of KNN algorthim is huge computation time becuase of tremendous calculatuions of distances.In this project we are using cuda programming to parallelize the distance calclutions and speed up the knn algorithm by following master and slave approach. The dataset uses is a dummy image data created within the code. and parallel KNN outperms the sequential KNN by huge time difference.

Introduction:

K nearest neighbors is a machine learning model free algorithm which is used for classification problems. The main problem with KNN is hugh amount of calculations of distances which drastically slow down the knn and KNN is not used for big datasets.

But time has changed. Now we have powerful gpus which can do Thousands calculations in parallel and which can drastically improve performance.GPU contains many cores that are purposefully made for parallelism.

In the K Nearest Neighbors algorithm, There is set of training set which consists of points in high dimensional space where the true labels are known. Our motto is to classify the new datapoint whose label is unknown which is called a test point. In order to classify the datapoint we calculate the distance of the test point to all the training points in the high dimensional space and take the K nearest neighbors provided the k value as hyper parameter.



And we classify the test point as the label X which is the most common label in those K nearest neighbors.

KNN is model free which means there is no training phase. And knn has better accuracies than most neural networks provided huge training set.So parallelize KNN and speeding up has many uses in real life from biotechnology to image processing.

The main objective of the project is to implement the KNN in parallel in cuda C to speed up the distance calculation process which is the bottleneck of this algorithm.Three places where we can parallelise the Knn algorithm one is in calculation of distance between two points. Second is dividing the training and testing data to chunks of small pieces. Third is parallelizing the sort algorithm.

Literature Review:

Garcia et al. parallelised K-nearest neighbor algorithm using cuda and C using blocks and threads. Each thread calculates distances between test point and subset of train points.Garcia et al. classify only one query per iteration, The approach which classifies several queries per iteration is better and faster.

In 2012, Sismanis, Pitsianis, and Sun conducted a study on K-Nearest Neighbors (KNN) algorithm for higher dimensions using GPU.They parallelized the KNN algorithm using truncated bitonic sort (TBiS) for sorting, which is a sorting technique optimized for parallel computing, and found that utilizing GPUs with TBiS provides better performance compared to traditional sort and select techniques. The study showed that parallelizing the KNN algorithm using GPUs with TBiS improved the speed of computations, making it a promising approach for high-dimensional data analysis.

In 2019,parallel KNN algorithms was used for 3d construction matching.For the research's higher quality photos, a GPU device running the Nvidia CUDA SDK was employed.The distance calculation results are kept in shared memory and used in the real time feuature tracking which significantly saved alot of time. According th their research parallel KNN is 10 times faster than serial distance

In a 2019 research. A parallel knn classifier was implemented in MPI C++ using two methods task parallelism and data parallelism.To compare Both the methods classifier was made to classify Images. Data parallelism is more effective according to their research.In our project we implemented Data parallelism by dividing the dataset into chunks of small sizes.

Research was done in 2020 to categorize encrypted data using the parallel K-Nearest Neighbours (KNN) algorithm on cloud services. The study used data parallelism to the KNN method and discovered that the sequential KNN took between 12.02 and 55.5 minutes to process the identical data, but the parallel KNN took just 4.16 minutes. This shows that KNN-based classification tasks may be performed much more effectively when using parallel KNN operating on cloud services, especially when dealing with huge datasets.

METHODOLOGY:

The main pipeline of our project is to parallelize the knn algorithm.For that we are using data parallelism where we are dividing the training set into chunks of 32. and utilizing 32 threads in our program each work on its own chunk.We are using master and slave approach where there is a master node which is host or main of the C program and set of slave nodes which actually do all the work

1. Master node which is the host
2. slave nodes of 32 which are parallel threads
3. Divide the training set into chunks of 32
4. assign each chunk to its corresponding thread
5. To classify the test set,each thread calculates the test point distance to all training point to the corresponding chunk
6. master node takes all distances from all threads
7. Sort the distances and find K nearest neighbors
8. identify the class label

In our project we are using knn to classify image to a binary class problem 0 or 1. we randomly created dummy images of size 64X64 and label as 0 or 1 for images.The serial KNN calculates distance from test image to all images in training images using euclidean distance and sorting them.
The parallel KNN divides the test set consisting of points to be classified into chunks and there are a total 32 threads that work upon the test set in an exclusive manner.

32+32
0
0X32+32

Thread allocation is by this logic:

```
thread0 => start index as 0
           end index as 0+32
thread1=> start index as 32
          end index as 32+32
   .
   .
thread31=> start index as 31*31
           end index as total test set lenght

thread t will get start as t* chunk size
              end as start+chunk
```

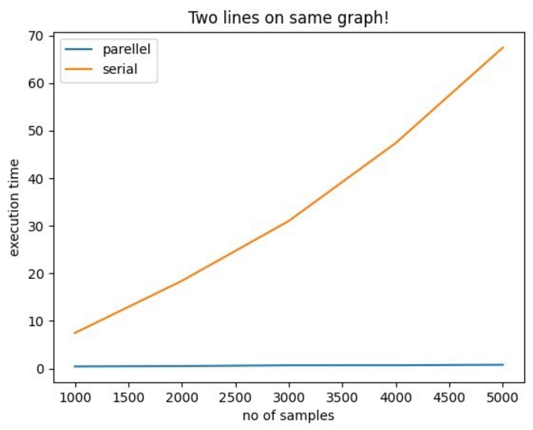
Each thread will work within its start and end of the test set there for concurently working on distance calculations and speeding up the entire algorithm.In our project we are using 1 block and 32 threads to use on the kernel.

so technically if it's ideal then speed up should be increased by 32 times as we are processing 32 chunks at the same time.

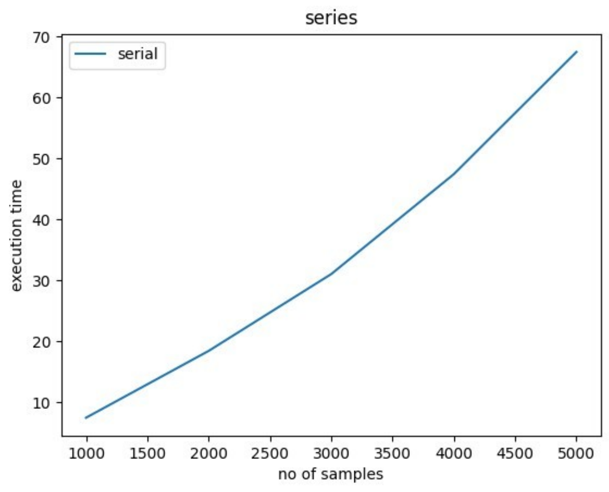
RESULTS:

The parallel KNN outperforms the serial version by a huge margin. we have tested the time taken to run the program using n= 1000,2000,3000,4000,5000 and plotted the time taken to run the algorithm vs no of testing points and its been observed that as no of sample increses there is a exponential growth in serial code and barely any time difference in parallel version compared to serial.

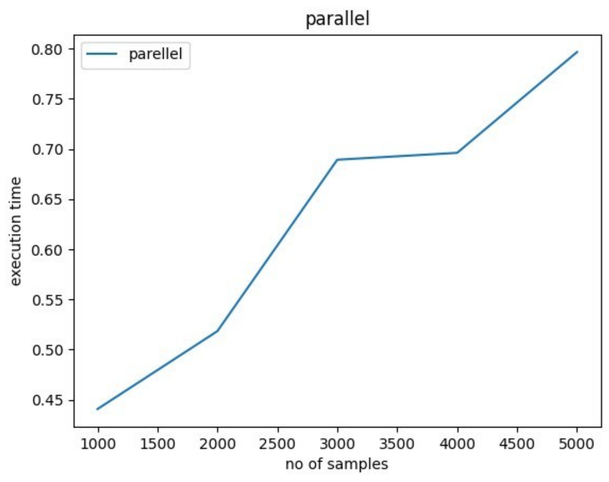
The plot showing no of samples vs execution time in seconds on serial and parallel



the execution time in seconds vs no of samples in serial:



The execution time in seconds vs no of samples in parallel:



CONCLUSION:

There is a significant improve in KNN algorithm when parallelised by GPU.but our project can be further parallelised.sorting is the first thing that can be parallelised and also distance between 2 vectors in high dimensions can also be parallelised

The contributions of the project are:
MS DHEERAJ --> parallel code
M.krishna vamshi--> parallel code
D. Steven chris--> serial code and plotting of graphs
M.bannu deepak--> serial code

So By conclusion our project is a simplest way of implementing parallelism on KNN algorithm to speed up using CUDA C.There are many ways and better implementations than this.And speeding KNN has many benifits because it makes KNN highly scalable.hence can be used for small scale productions and also fields like biotechnology.

REFERENCE:

[1] Applying Parallel Processing to Improve the Computation Speed of K-Nearest Neighbor Algorithm by Maha A. Alanezi and Abdulla AlQaddoumi
[2] N. Sismanis, N. Pitsianis and X. Sun, "Parallel search of k-nearest neighbors with synchronous operations," in 2012.
[3]"Parallel K Nearest Neighbor Matching for 3D Reconstruction," by M.-W. Cao, L. Li,
[4]A CUDA-based Parallel Implementation of K-Nearest Neighbor Algorithm by Shenshen Liang,Ying Liu.
[5]A high efficient and fast kNN algorithm based on CUDA by pei Tong and Zhang