Convex Hull Using K-Means Clustering in Hybrid(MPI/OpenMP) Environment

Presented by:

Kodingari Rajasekhar (202IT009)

Manaswita Datta (202IT011)

Presented To:

Prof Ananthanarayan V.S

Content

- Introduction
- Convex hull
- MPI
- OpenMP and Hybrid
- K-means clustering
- Quick hull
- Quick hull using K-means clustering
- Experiment setup
- Result
- Conclusion
- Reference

Introduction

- Parallel computing is the use of multiple compute resources to solve a computational problem.
- It is the ability of program to run on parallel computers which is still quite difficult and complex task to achieve.
- In this project, K-Means Clustering algorithm is used for solving the problem of Convex Hull in parallel environment.
- This algorithm is implemented in MPI, OpenMP, and Hybrid mode.

Convex hull

- **Convex hull:** Given a set S of n points in the plane, the convex hull of S is the smallest convex region containing all points in S.
- They have been proved to be useful in applications such as pattern recognition, including algorithms for detecting human faces, reading a license plate or analyzing soil particles etc.
- If the number of points increases, the time required to create Convex hull also increases.
- To reduce this time, the problem must be broken into multiple threads or processes.

MPI

- The generic form of message passing in parallel processing is the Message Passing Interface (**MPI**), which is used as the medium of communication.

- Limitations of MPI:
  - In MPI communication can often create a large overhead, which needs to be minimized.
  - Global operations can be very expensive.
  - Significant change to the code are often required, making transfer between the serial and parallel code difficult.

OpenMP and Hybrid mode

- **OpenMP** is an Application Program Interface (API) that is used to explicitly direct multi-threaded and shared memory parallelism.
- Limitations of OpenMP:
  - OpenMP works only for shared memory.
  - Threads are executed in a non-deterministic order.
  - OpenMP requires Explicit synchronization.

- **Hybrid model** combines both approaches in the pursuit of reducing the weaknesses in individual. The idea of using OpenMP threads to exploit the multiple cores per node while using MPI to communicate among the nodes appears.

K means clustering

- K-means clustering is an algorithm to classify or to group the objects based on attributes into K of groups called clusters.
- The way k-means algorithm works is as follows:
  - Specify number of clusters *K*.
  - Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
  - Keep iterating until there is no change to the centroids i.e., assignment of data points to clusters isn’t changing.
  - Compute the sum of the squared distance between data points and all centroids.
  - Assign each data point to the closest cluster (centroid).
  - Compute the centroids for the clusters by taking the average of the all the data points that belong to each cluster.

Quick hull

- Quick hull algorithm is them applied on the individual clusters for form k convex hulls. The Quick hull algorithm is a Divide and Conquer algorithm.
- Let a[0…n-1] be the input array of points. Following are the steps for finding the convex hull of these points:

1. Find the point with minimum x-coordinate let's say, min_x and similarly the point with maximum x-coordinate, max_x.
2. Make a line joining these two points, say L. This line will divide the whole set into two parts. Take both the parts one by one and proceed further.
3. For a part, find the point P with maximum distance from the line L. P forms a triangle with the points min_x, max_x. It is clear that the points residing inside this triangle can never be the part of convex hull.
4. The above step divides the problem into two sub-problems (solved recursively). Now the line joining the points P and min_x and the line joining the points P and max_x are new lines and the points residing outside the triangle is the set of points.
5. Repeat point no. 3 till there no point left with the line. Add the end points of this point to the convex hull.

Quick hull using K means clustering (Hybrid approach)

Algorithm:

1) Take input array of 2D points.

2) Determine the centroid coordinate.

3) Determine the distance of each object to the centroids.

4) Use MPI task for each cluster.

5) Group the object based on minimum distance.

6) Call the Convex hull algorithm on each cluster points to find out the Clusters Convex hull Boundary Points.

7) In each cluster 1st find the X minimum and X maximum (Extreme points).

8) In each cluster divide the points in two parts, Upper half and Lower half.

9) Again use MPI task for Upper hull and Lower hull for each of the Cluster.

Quick hull using K means clustering (Hybrid approach)

Algorithm(2):

10)By using the Area of Triangle find out the Clusters Convex hull Boundary points for each half of the cluster.

11)Also use the OpenMP task for each Sub hulls in each cluster.

12)Then get Clusters Convex hull Boundary Points for each cluster.

13)Again call the Convex hull algorithm for computing the Final Clusters Convex hull Boundary Points.

14)Then consider the clusters convex hull boundary points each cluster as input and repeat the step (7), (8), (9) and (10).

15)Then finally get the Final Clusters Convex hull Boundary Points.

16)Terminate all MPI communication and computation.

17) End.

Experiment setup

- Programming Language: Cpp
- No of clusters: 2, 3, 4
- No. of points in each cluster: 1000, 5000, 10000, 15000
- Experimented on master node of the server.

Results

- These results are taken from the implemented algorithms of Cluster Convex Hull in MPI, OpenMp and Hybrid parallel programming approach.
- The implemented algorithms are tested for different set of points and different number of clusters. As we can clearly see in below tables that the number of points increases the time required to solve the Convex Hull also increases.

Points	MPI(msec)	OpenMp(msec)	Hybrid(msec)
1000	0.023153	0.024051	0.022215
5000	0.026332	0.062936	0.02447
10000	0.02765	0.093253	0.028226
15000	0.03023	0.120743	0.031211

TABLE I PERFORMANCE OF CLUSTERS CONVEX HULL FOR 2 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(msec)	OpenMp(msec)	Hybrid(msec)
1000	0.024664	0.028931	0.024538
5000	0.027163	0.056868	0.027703
10000	0.029013	0.109241	0.031000
15000	0.033421	0.166041	0.034841

TABLE 2 PERFORMANCE OF CLUSTERS CONVEX HULL FOR 3 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(msec)	OpenMp(msec)	Hybrid(msec)
1000	0.02530	0.042040	0.024953
5000	0.028199	0.092855	0.029132
10000	0.031667	0.141979	0.032629
15000	0.032588	0.192424	0.035942

TABLE 3 PERFORMANCE OF CLUSTERS CONVEX HULL FOR 4 CLUSTER AND DIFFERENT SET OF POINTS

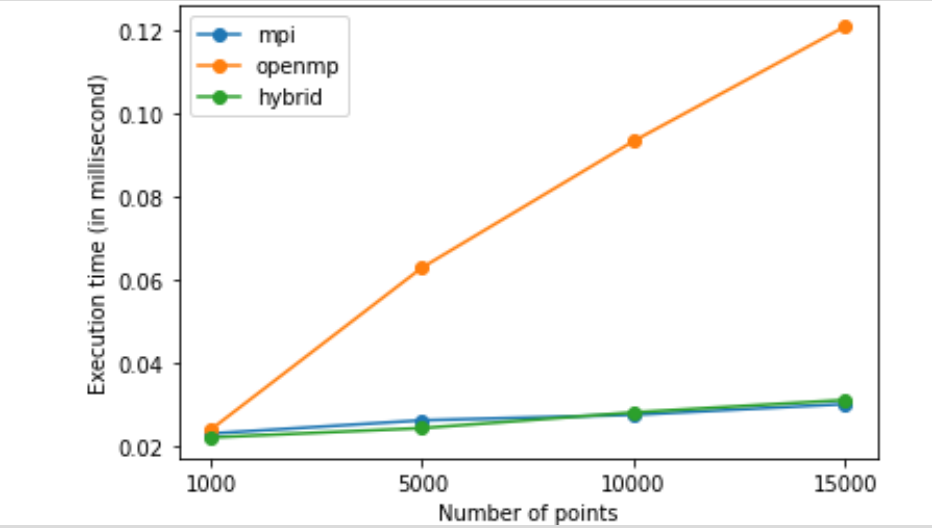

Figure 1. Graphical representation for performance analysis of Clusters Convex Hull for 2 clusters

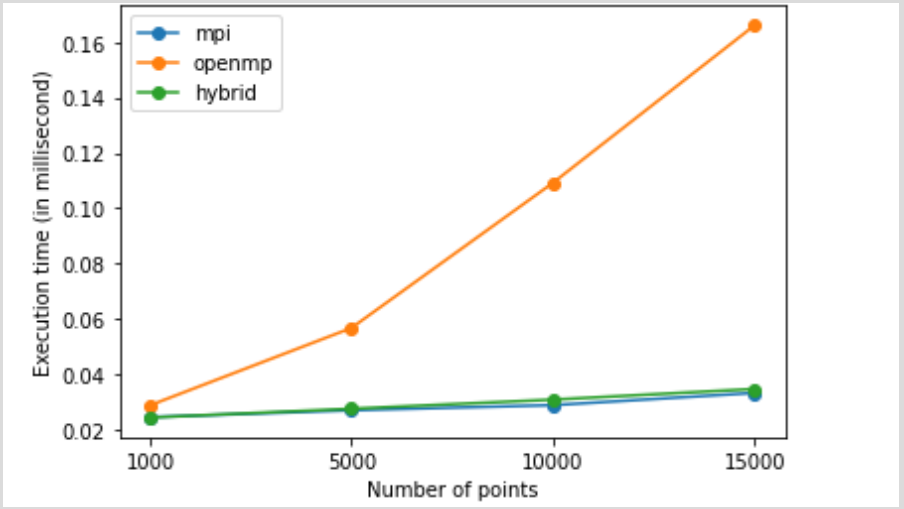

Figure 2. Graphical representation for performance analysis of Clusters Convex Hull for 3 cluster

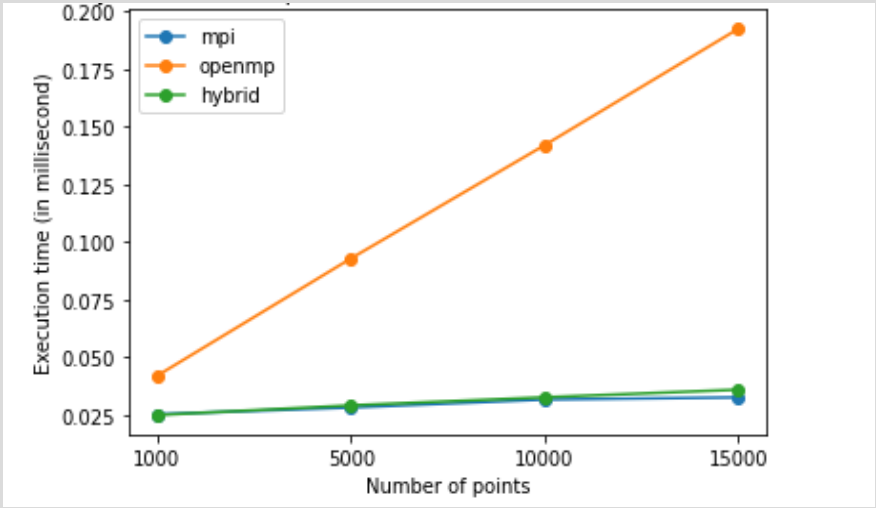

Figure 3. Graphical representation for performance analysis of Clusters Convex Hull for 4 clusters

Conclusion

- We observed that OpenMP implementation of cluster convex hull takes more execution time than others irrespective of number of clusters and points considered.
- We noticed that MPI implementation of cluster convex hull performs better than OpenMP. But when compare with Hybrid implementation there is no such significant change in the performance between them (we can observe this in graphs also).
- Overall Hybrid mixed mode programming model implementation of Cluster Convex hull gives better performance than others and it also scalable.

Future work /Suggestion:

- Here we considered 2,3 and 4 clusters and number of points up to 15000 only. By taking a greater number of clusters and points, we would be able to distinguish between Hybrid and MPI mode of programming performance.

References

- Waghmare, Vivek N., and Dinesh B. Kulkarni. "Convex Hull Using K-Means Clustering in Hybrid (MPI/OpenMP) Environment." In *2010 International Conference on Computational Intelligence and Communication Networks*, pp. 150-153. IEEE, 2010.

- https://www.open-mpi.org/doc/current/

- https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf

THANK YOU