2010 International Conference on Computational Intelligence and Communication Networks

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

Vivek N. Waghmare
Walchand College of Engineering,Sangli
Maharastra, 416415
INDIA
comsvivek@gmail.com

Dinesh. B. Kulkarni
Walchand College of Engineering,Sangli
Maharastra, 416415
INDIA
d_b_kulkarni@yahoo.com

***Abstract*—Parallel computing is the use of multiple compute resources to solve a computational problem. Parallel computers can be roughly classified as Multi-Core and Multi-Processor. In both these classifications, the hardware supports parallelism with computer node having multiple processing elements in a single machine. Parallel programming is the ability of program to run on this infrastructure which is still quite difficult and complex task to achieve. Two of the different approaches used in parallel environment are MPI and OpenMP, each one of them have their own merits and demerits. Hybrid model combines both approaches in the pursuit of reducing the weaknesses in individual.**

In proposed approach k- Means Clustering algorithm used for solving the problem of Convex Hull in parallel environment. In this design, 2D points are grouped into different Cluster and then Convex hull for each of these Clusters are computed. Points defining these Convex hulls are used to construct the final Convex hull. This algorithm is implemented in MPI, OpenMP, and Hybrid mode. The algorithm is tested for number of Clusters with different set of points. The results indicates that the Hybrid approach out performs the MPI and OpenMP approach.

***Index Terms*—Prallel programming, MPI, OpenMP, Hybrid (MPI+OpenMP), Convex hull, K-Means Clustering Algorithm.**

I. Introduction

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 multile threads or processes. 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. The convex hull (S) is considered as an ordered list. One of the reasons for the choice of the convex hull is that, in addition to be considered a main topic in Computational Geometry, convex hulls have attracted the interest of researchers from a wide range of areas [1]. Because of the complexity among others, 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; computer graphics, computerized tomography, collision detection, prediction of chemical equilibrium or ecology.

Many different design approaches lead to optimal (or expected optimal) solutions. These include divide and conquer, sweep line, incremental randomized constructions, quick hull, Graham scan and many others. This fact, together with the possibility of optimizing, its execution time and expected memory requirement, makes the convex hull an excellent benchmark for testing different architectures and programming models. The divide and conquer algorithm is an elegant method for computing the Convex hull of a set of points based on this widely-known algorithm design technique. The main goal of opting for a parallel program is to utilize the mutticore resource variable in common for improving the performance of convex hull algorithm. There are several issues that one needs to consider when designing the parallel code to obtain the best performance possible within the constraints of available resources.

The generic form of message passing in parallel processing is the Message Passing Interface (MPI), which is used as the medium of communication. A standard Message Passing Interface (MPI) is originally designed for writing applications and libraries for distributed memory environments. However, MPI does provide message-passing routines for exchanging all the information needed to allow a single MPI implementation to operate in a heterogeneous environment [2].

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.
- In MPI Dynamic load balancing is often difficult.

OpenMP is an Application Program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism. It is a specification for a set of compiler directives, library routines and environment variables [3]. The available programming environment on most of the Multi-Core processors will address the thread affinity to core and overheads in OpenMP Programming environment.

Limitations of OpenMPI

- OpenMP works only for shared memory.
- Limited scalabilty, not much speed up.
- Threads are executed in a non deterministic order.
- OpenMP requries Explicit synchronization.

Combining shared-memory and distributed-memory programming model is an old idea. One wants to synergize the

978-0-7695-4254-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CICN.2010.40

strengths of both models: the efficiency, memory savings, and ease of programming of the shared-memory model and the scalability of the distributed-memory model [4]. Until recently, the relevant models, languages, and libraries for shared-memory and distributed-memory architectures have evolved separately, with MPI becoming the dominant approach for the distributed-memory, or message-passing, model, and OpenMP emerging as the dominant high-level approach for shared memory with threads.

The idea of using OpenMP [5] threads to exploit the multiple cores per node while using MPI to communicate among the nodes appears. It appears to be heavily dependent on the hardware, the MPI and OpenMP implementations, and above all on the application and the skill of the application writer.

II. Design Issues

K-means clustering is an algorithm to classify or to group the objects based on attributes into K of groups called clusters [6]. The K-Means clustering algorithm will carry out the following three steps to group the objects in cluster.

- Determine the centroids for each clusters.
- Determine the distance of each object to the centroids.
- Regroup the object based on minimum distance.

A convex hull is an important structure in geometry that can be used in the construction of many other geometric structures. The convex hull of a set S of point in the plane is defined to be smallest convex polygon containing all points of S. The subset of S forms a convex hull. There are two steps of the convex hull problem:

- Obtain the vertices of the convex hull.
- Obtain the vertices of the convex hull in some order (clockwise or anticlockwise).

K-means is a popular clustering method because it is simple to program and is easy to compute on large samples. The k-means clustering algorithm classifies N vectors with d dimension into k clusters by assigning each vector to the cluster whose average value is nearest to it by some distance measure on that set [6]. The algorithm computes iteratively, until reassigning points and recomputing averages reaches optimal value.

Initially given k, total number of clusters, the training samples are assigned to k clusters randomly. Alternately the first k training sample can be considered as single-element cluster and theneach of the remaining (N-k) training sample is assigned to the nearest cluster [7]. After each assignment, the cluster centers are recomputed as reffered as centroid of the cluster. Again each sample is taken in a sequence and computed its distance from the centroid. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample. Repeat step 3 until convergence is achieved, that is until sample causes no new assignments. We calculate the distance to all centroid and get the minimum distance. At

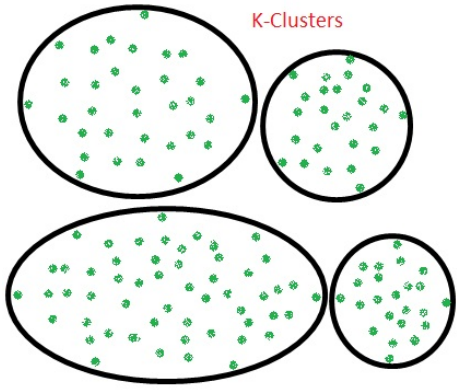

Fig. 1. K-Clusters

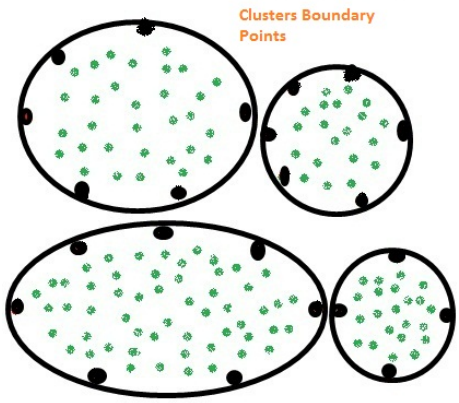

Fig. 2. Clusters Boundary Points

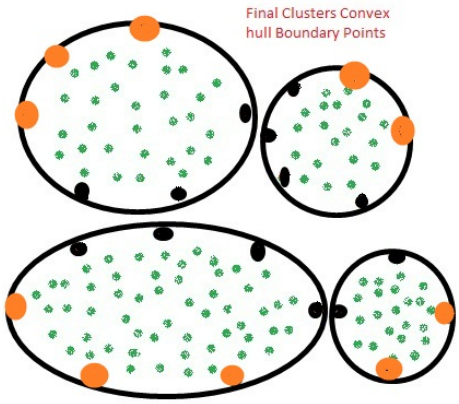

Fig. 3. Final Clusters Convex hull Boundary Points

the termination of the K means algorithms all the ponts will belong to a cluster with shortest distance from its centroid. Figure (1) shows the clusters formed with number of points and 4 clusters.

Convex hull algorithm is them applied on the individual clusters for form k convex hulls. These k convex hulls are merged together to form final convex hull. The Quick Hull algorithm is used for computing the convex hull. This algorithm is closely related to the quick sort algorithm. The Quick Hull begins with the generation of the extreme points along the compass directions. This divides each cluster into two parts i.e Upper hull and Lower hull. For each hull search for the point which will have maximum area for the triangle formed with the extreme points. Find out the points which are inside or outside of the triangle. This can be done by 3*3 ISLEFT Determinant function. If the value of the determinant is negative the point is inside the triangle otherwise the point is outside.

This will give Convex hull Boundary Points for each cluster as shown in Figure (2). Points defining these convex hulls for each cluster are used to construct the final Convex hull by application of the Convex hull Algorithm on these points. These results are shown in Figure (3) and the final Convex hull as shown in Figure (4).The parallel version of Clusters Convex hull program is design in MPI, OpenMP and Hybrid.

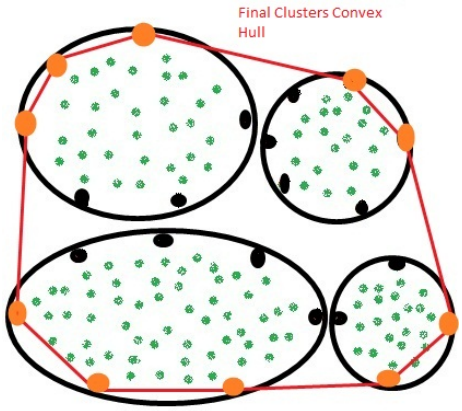

Fig. 4. Final Clusters Convex hull

Hybrid approach:-

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 clusters 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 by using ISLEFT determinant.
9) Again use MPI task for Upper hull and Lower hull for each of the Cluster.
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.

III. Results

The results taken from the implemented algorithms of Clusters Convex hull, in MPI, OpenMP, and Hybrid are analyzed using different parallel programming tools. The implemented algorithms are tested for different set of points and it is observed that as the number of points increases the time required to solve the Clusters Convex hull also increases as shown in be Table (1) and (2).

It can be noticed that MPI implementation of Clusters Convex hull performs better than OpenMP, and Hybrid for Convex hull for 8 and 16 clusters with small set of points. Hybrid model scales well for large set of points. For 24 and 32 clusters for different set of points, the Hybrid implementation of Cluster Convex hull gives better performance.

The graphical representation of performance analysis for Cluster Convex hull on PGI tool kit has shown in Table (1), (2), (3) and (4). All this results are taken for 8, 16, 24, 32 clusters, which are graphically shown by Figure.(5), (6), (7), and (8).

TABLE I
PERFORMANCE OF CLUSTERS CONVEX HULL FOR 8 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(Msec)	OpenMP(Msec)	Hybrid(Msec)
100	0.2861	0.3991	0.2899
500	0.7610	1.168	0.8558
1000	1.297	2.509	1.266
5000	3.854	5.974	3.733
10000	6.476	11.140	6.189

TABLE II
PERFORMANCE OF CLUSTERS CONVEX HULL FOR 16 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(Msec)	OpenMP(Msec)	Hybrid(Msec)
100	0.3681	0.5617	0.389
500	0.9723	1.197	0.1011
1000	1.605	2.7533	1.504
5000	6.716	14.91	6.621
10000	12.84	22.83	11.59
50000	58.40	79.61	53.04
100000	102.8	135.9	94.24

TABLE III
PERFORMANCE OF CLUSTERS CONVEX HULL FOR 24 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(Msec)	OpenMP(Msec)	Hybrid(Msec)
5000	8.93	20.35	8.693
10000	17.68	25.36	15.05
50000	82.9	91.02	72.711
100000	124.4	189.5	115.7

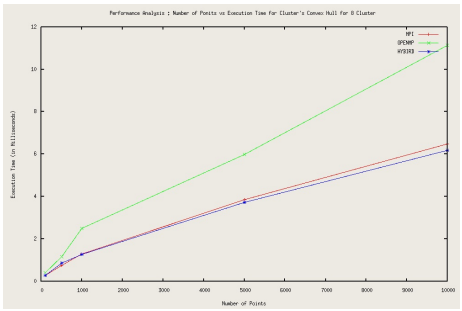

Fig. 5. Graphical representation for performance analysis of Clusters Conex Hull for 8 cluster

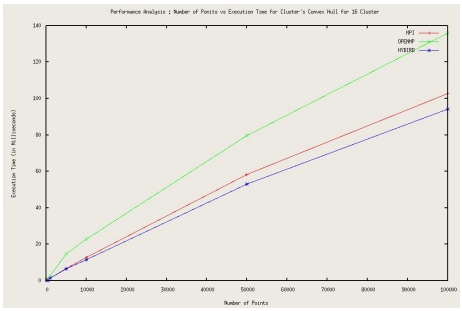

Fig. 6. Graphical representation for performance analysis of Clusters Conex Hull for 16 cluster

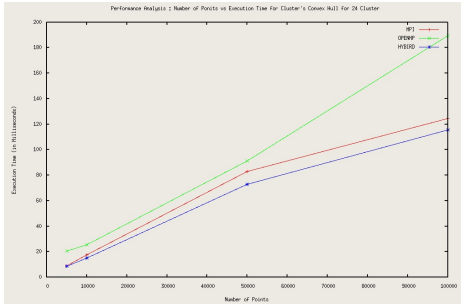

Fig. 7. Graphical representation for performance analysis of Clusters Conex Hull for 24 cluster

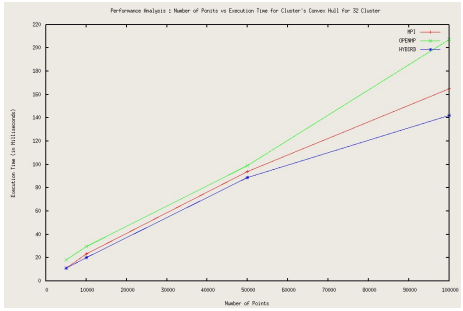

Fig. 8. Graphical representation for performance analysis of Clusters Conex Hull for 32 cluster

TABLE IV
PERFORMANCE OF CLUSTERS CONVEX HULL FOR 32 CLUSTER AND DIFFERENT SET OF POINTS

Points	MPI(Msec)	OpenMP(Msec)	Hybrid(Msec)
5000	11.20	18.17	11.08
10000	23.35	29.73	20.06
50000	93.96	99.04	88.61
100000	165.1	207.1	142.2

IV. CONCLUSION

This paper, compares the performance for Clusters Convex hull program by using MPI, OpenMP, Hybrid (MPI+OpenMP). It is observed that the Hybrid mixed mode programming model gives better performance than that of MPI and OpenMP programming model and it also scalable.

REFERENCES

[1] http://www.softsurfer.com/Archive/algorithm_0109.html.
[2] http://www.mpi-forum.org/docs/docs.html.
[3] http://www.openmp.org/blog/specifications
[4] F. Cappello, D. Etiemble, Message Passing Interface versus Hybrid (MPI+OpenMP) on IBM SP ",for the NAS Benchmarks, presented at Super computing,Dallas http://www.sc2000.org/proceedings/techpapr/papers/pap214.pdf.
[5] Development and performance of a mixed OpenMP/MPI Quantum Monte Carlo code, L.A. Smith and P. Kent.
[6] http://www.cs.uic.edu/ wilkinson/Applets/cluster.html .
[7] http://people.revoledu.com/kardi/tutorial/kMean/index.html.