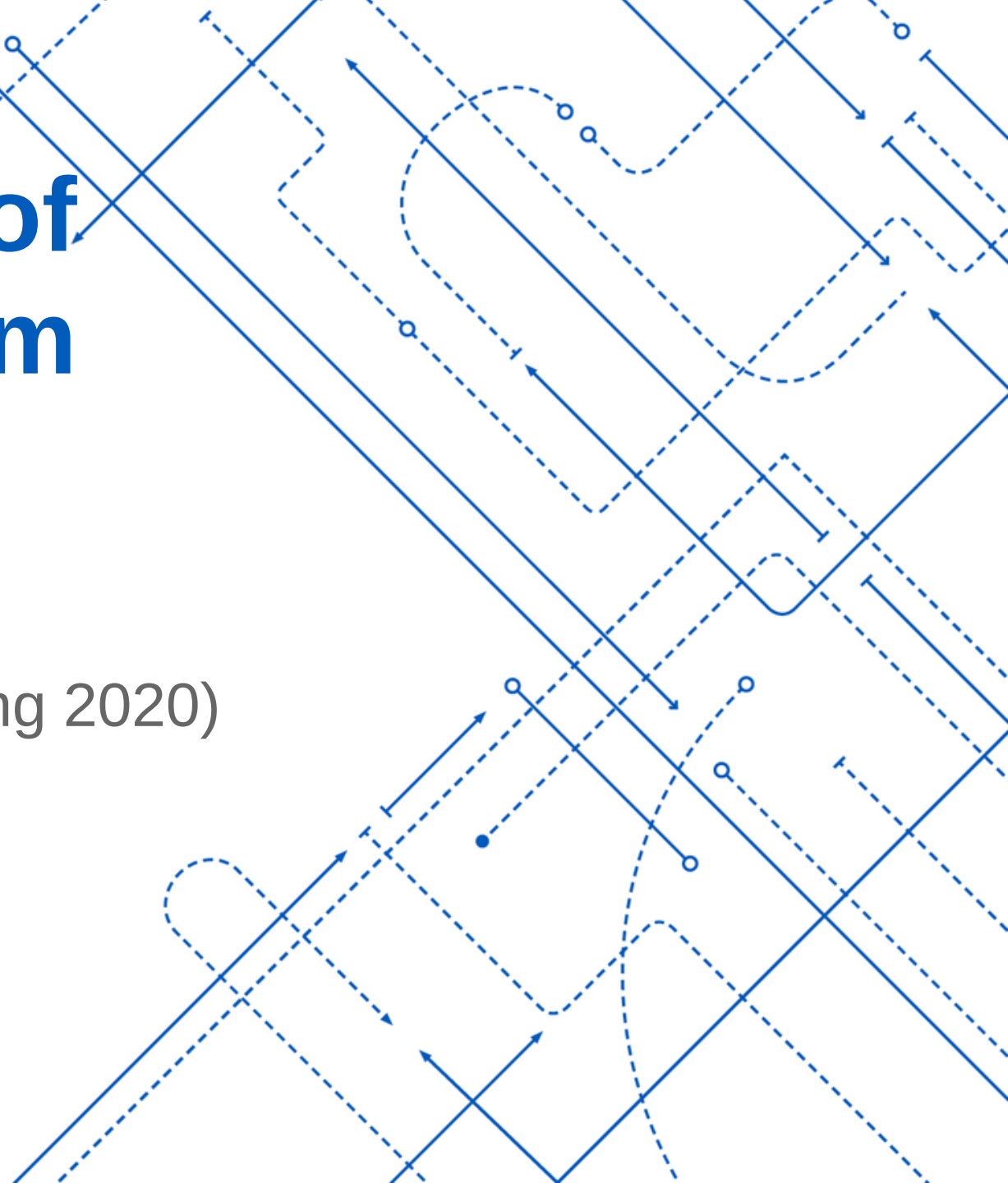
Parallel Implement of Convex Hull Problem

CSE 633 - Parallel Algorithms (Spring 2020)

Weiyang Chen

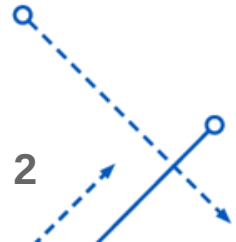
Instructor: Dr. Russ Miller

University at Buffalo The State University of New York



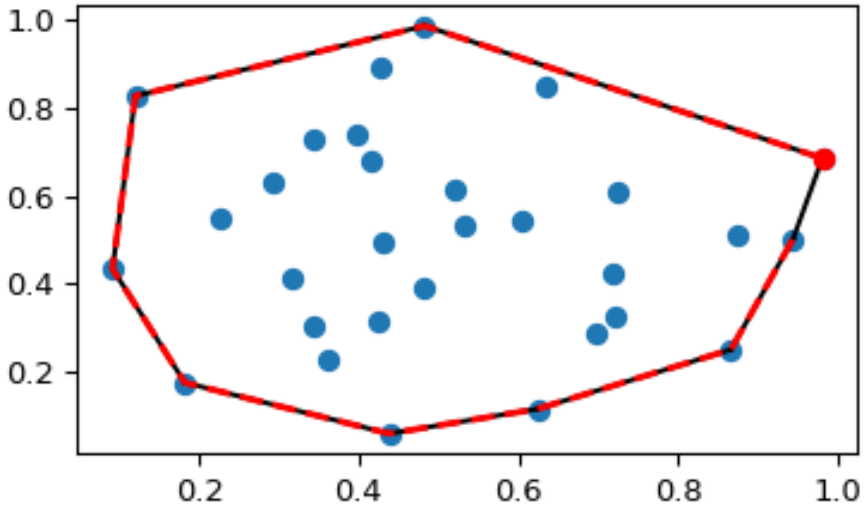
Outline

- Convex Hull Definition
- Graham’ Scan Aglorithm
- Combine Two Convex Hull
- Parallel Implementation
- Running Time Analysis
- Conclusion
- Future Work
- Reference



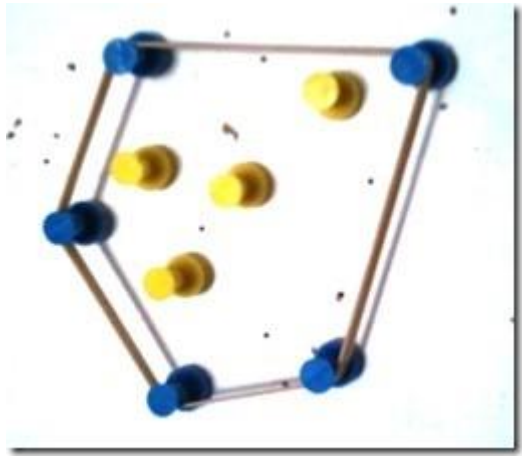
Convex Hull Definition

Assume we have a set of points in the plane. The convex hull of set of points is defined as the smallest convex polygon, that encloses all the points in the set



Convex Hull Example

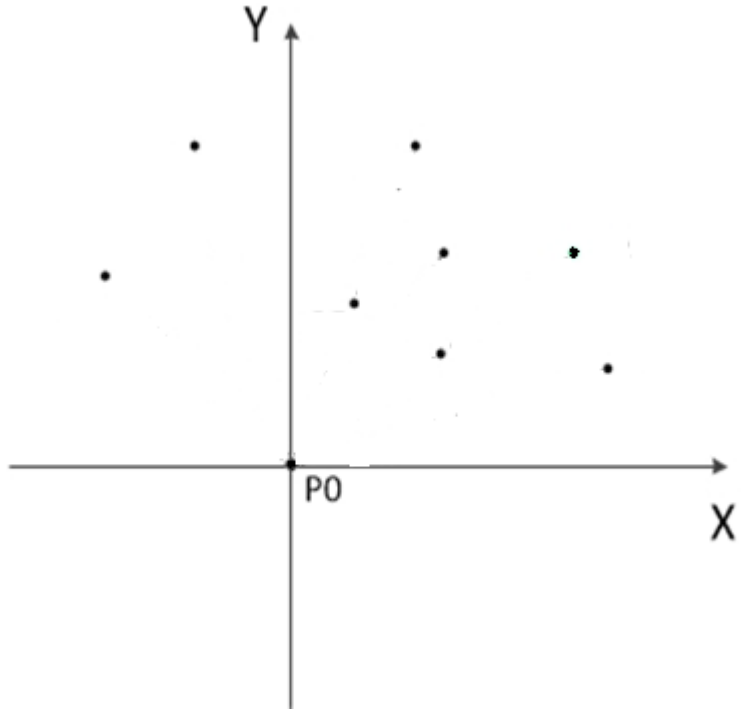
Imagine all the points as being pegs. The convex hull is the shape of rubber-band stretched around the pags.



Graham's Scan Algorithm

First :

Find a starting point that in the convex hull. (Lowermost point, if there is more than two lowermost points, then select leftmost one)



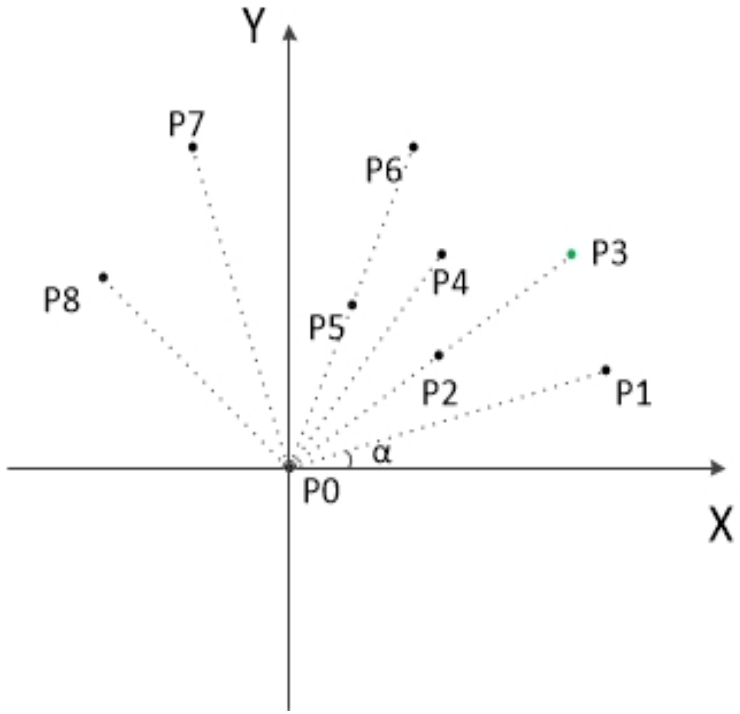
Second :

Compute remaining n - 1 points' angle with respect to the starting point.

Formula :

$$\phi = \cos^{-1} \frac{(x - x_0)}{\sqrt{(x - x_0)^2 + (y - y_0)^2}}.$$

After Computing, Sort the points by angle with respect to the starting point. (If two points have same angle, then sort by the distance respect to the starting point from small to big)

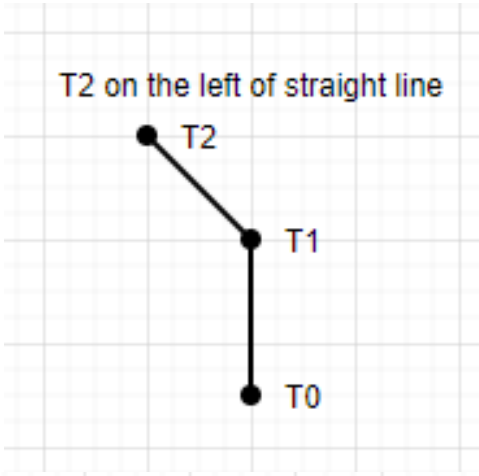


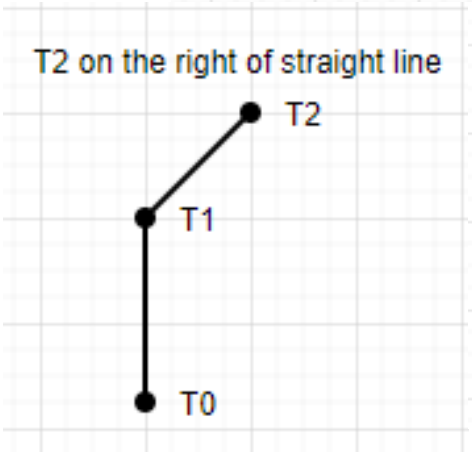
Graham's Scan Algorithm Continue

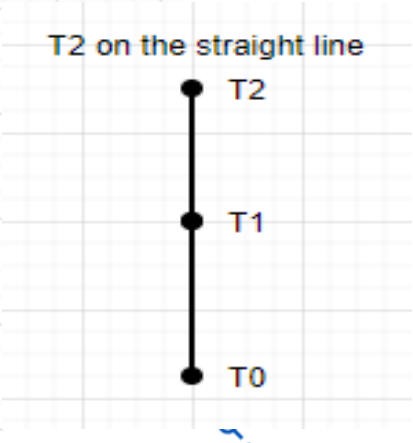
How to determine a point belongs to convex hull?

Scan:

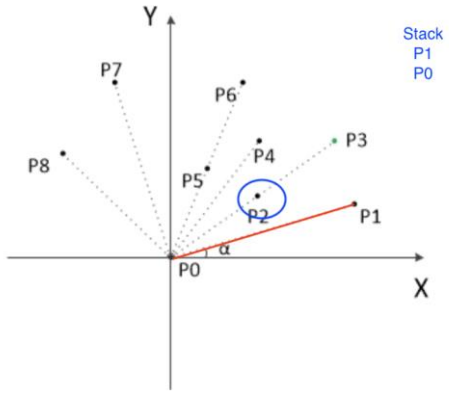
1. Using stack to store, add starting point and second point to the stack.
2. Start Scan from third point:
   a. Set T1 and T0 as top two points in the stack, and current scan point as T2
   b. Connect point T1 and T0 as straight line and to determine T2 is located on left or right of straight line. (Cross Product)
   c. If T2 on the left of straight line or on the straight line, then push the T2 into stack. and start next scan.
   d. If T0, T1, T2 are collinear then it means T1 is not extreme point, so remove T1, and start next scan
   e. If T2 on the right of straight line, then pop top point from the stack which is T1 and repeat from step a.
3. When we finish scanning, we will get a stack that contains all the points of convex hull in order.

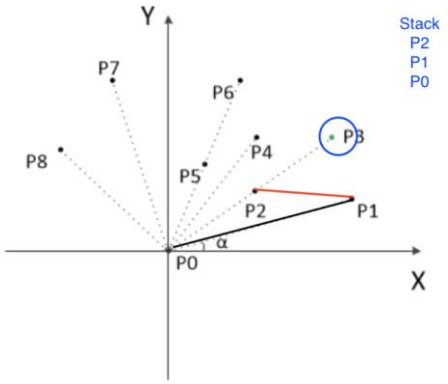


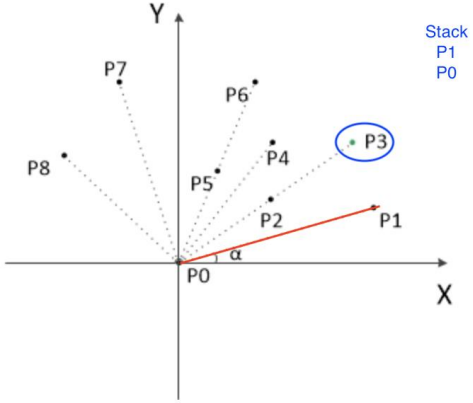


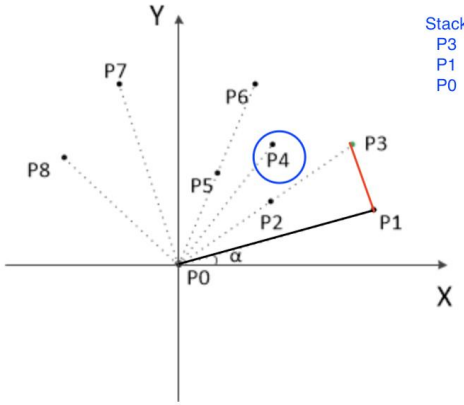


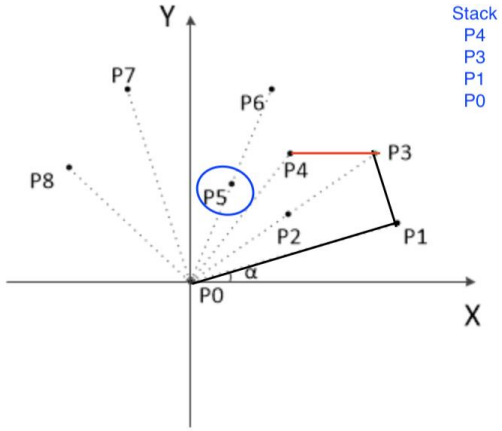
Graham's Scan Algorithm Continue

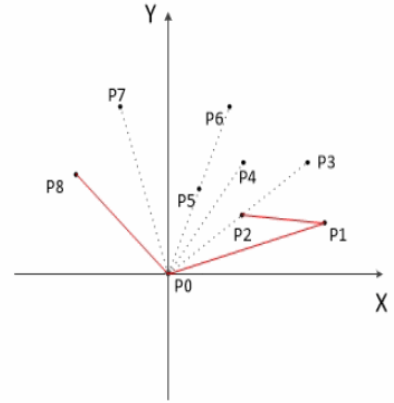






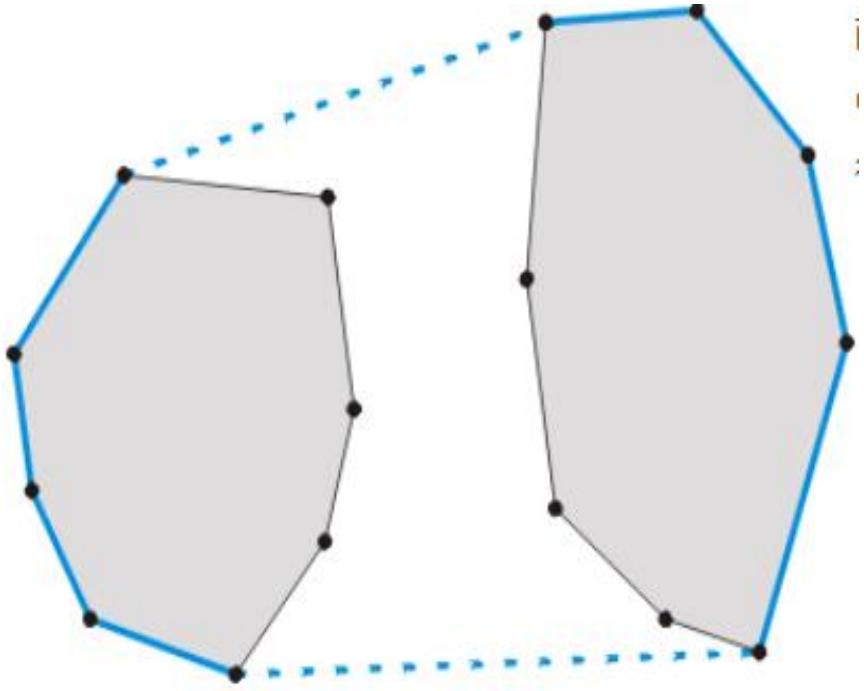






Merge Two Convex Hull Algorithm

The idea to merge two convex hull together is to find top and bottom tangent line, and connect them by the tangent line and remove all the points between two tangent lines.

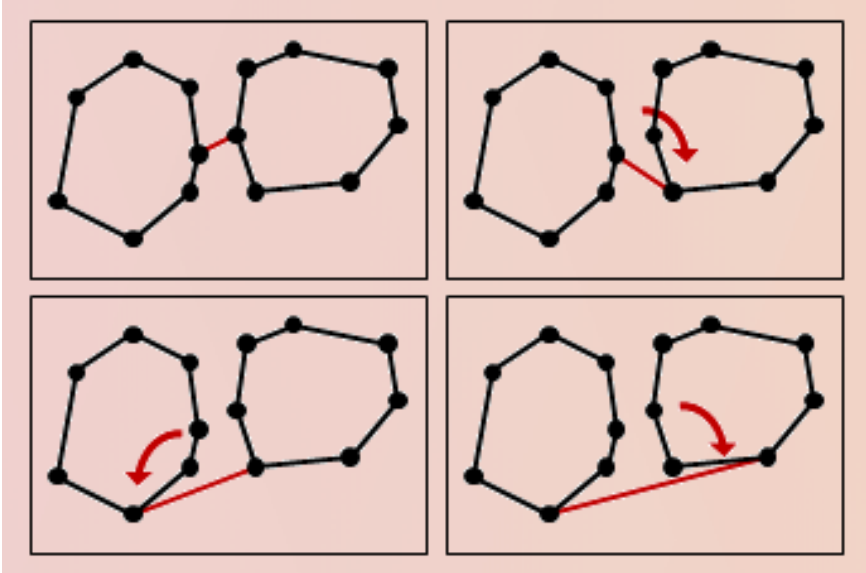


Merge Two Convex Hull Algorithm Continue

How to find the tangent line between two convex hull?

Bottom Tangent Line:

1. Find left convex hull's most right point P1, and right convex hull's most left point P2 and connect together.
2. Find the new P1 that is next point in the clockwise. And find new P2 that is next point in the counterclockwise.
3. Stop until it is fix.(Means if P1 or P2 go next, the connect line will cross to other line in the left or right convex hull)



Top Tangent Line:

Same to the Bottom tangent line, but P1 goes as counterclockwise and P2 goes as clockwise.

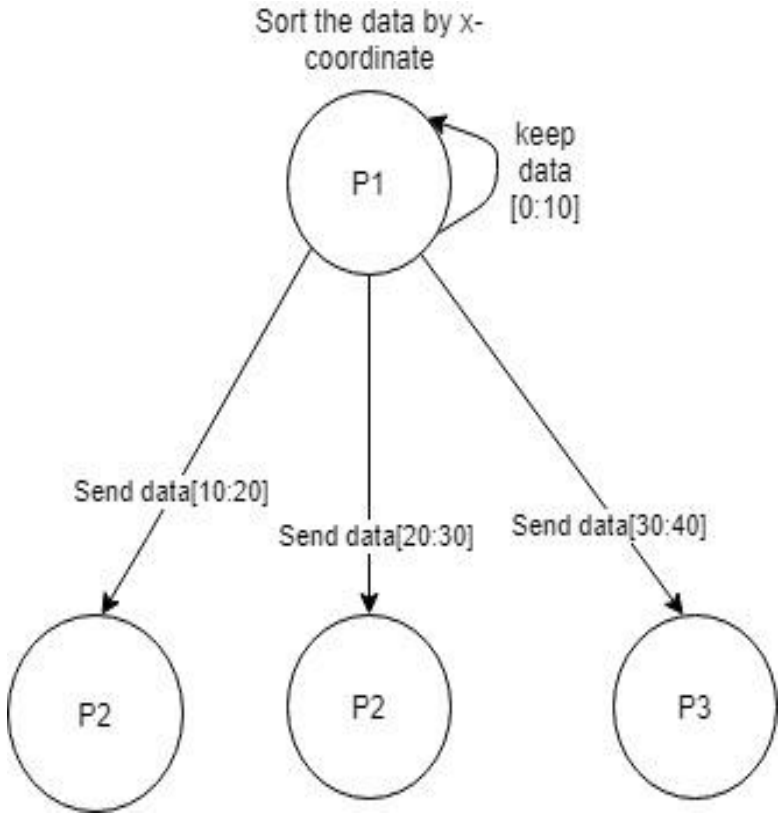
Parallel Implementation - Tree Structure

- Input a point data set with size n and p processors.
- Using master processor to sort the data by x-coordinate and separate the data to each processor equally.
- Each processor receive n/p size data from master processor, and parallelly find local convex hull by Graham's Scan.
- Recursively merge convex hull together between two processors by finding their top and bottom tangent line.

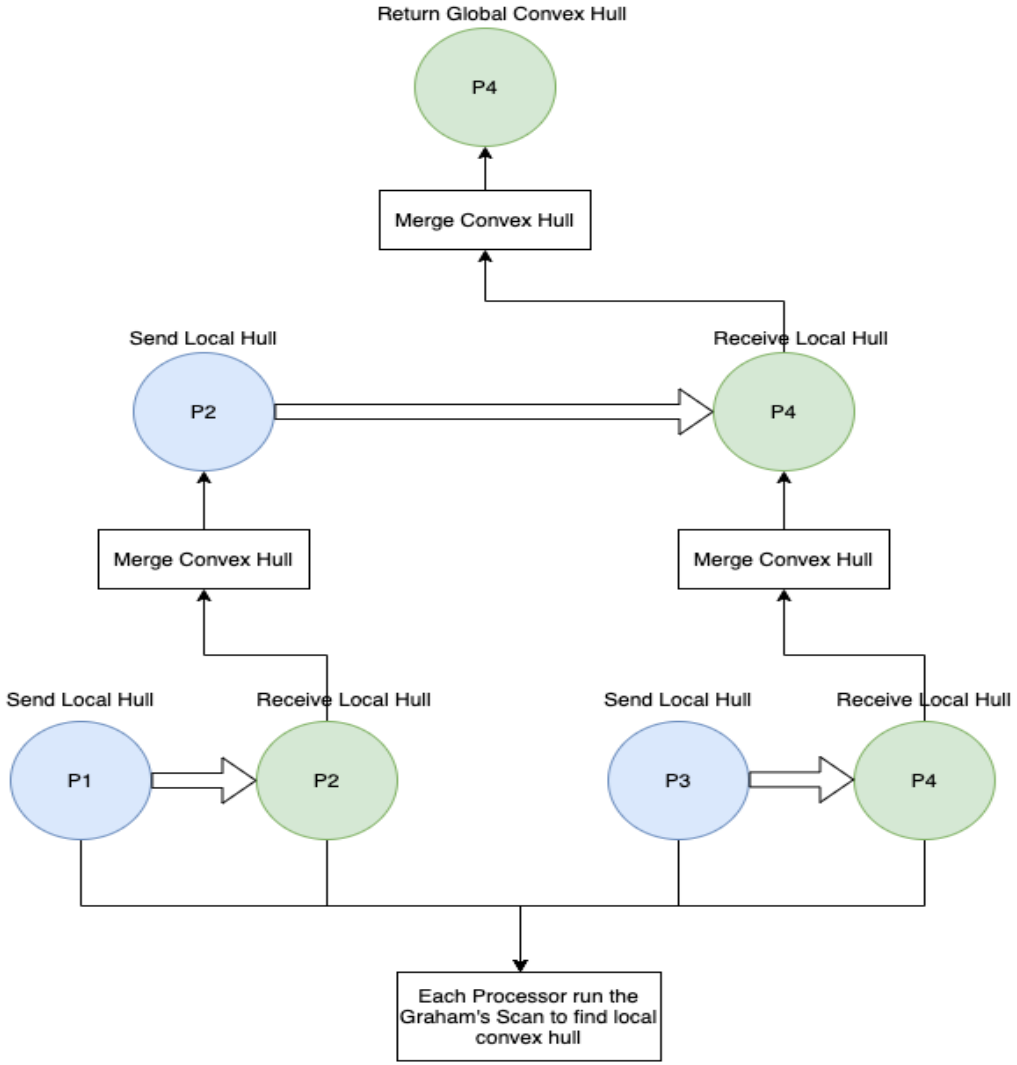
PARALLEL EXAMPLE

As Example :

Data Size = 40
# of Processor = 4



PARALLEL EXAMPLE



Parallel Implementation - Code

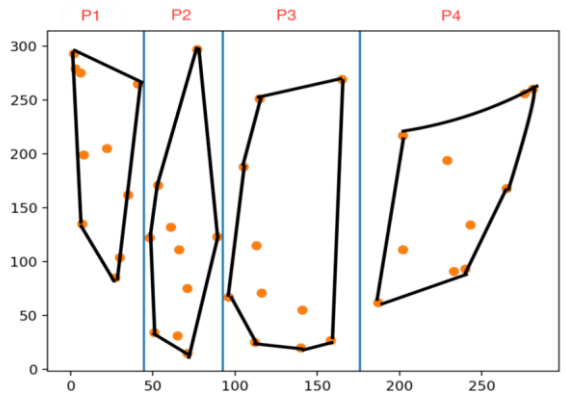
```
def treeBuildMPI(comm,data):
    size = comm.Get_size()
    rank = comm.Get_rank() + 1
    level = 1
    while(size>=(2**level)):
        if(rank%(2**level)==0):
            s = rank - (2**(level-1)) - 1
            get = comm.recv(source=s,tag=level)

            a,b = findBottomTangentLine(get,data)
            c,d = findTopTangentLine(get,data)
            data = combineTwoConvexHull(a,b,c,d,get,data)

            print("Rank %d Receive Data From Rank %d, And Combine Convex Hull In Level %d"%(rank,s+1,level))

        elif (rank+(2**(level-1)))%(2**level)==0:
            dest = rank + (2**(level-1)) -1
            comm.send(data,dest,tag=level)
            print("Rank %d Send Data To Rank %d In Level : %d"%(rank,dest+1,level))
        else:
            print("Rank %d Out In Level %d......."%(rank,level))
        level += 1
```

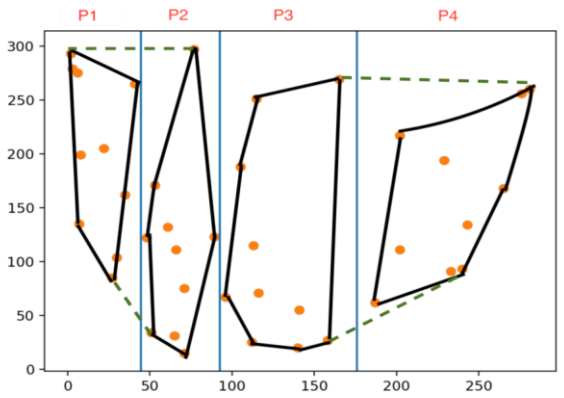
Receive Rank:
R = multiple of 2^level

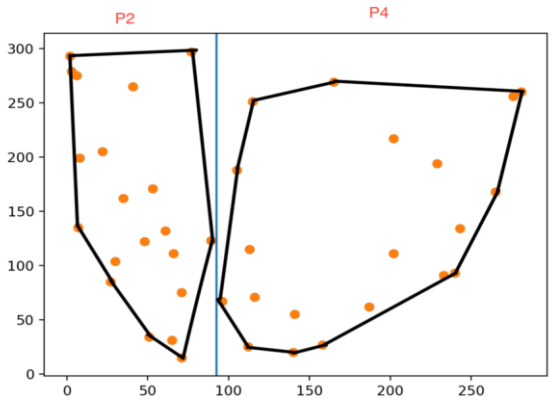
distance between Send and Receive :
D = 2 ^(level-1)

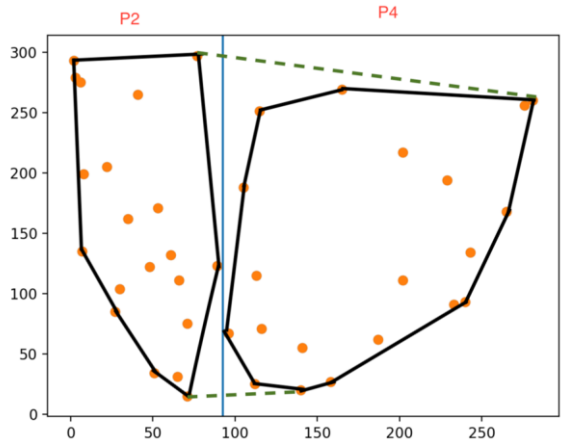
Send Rank:
S = R- D

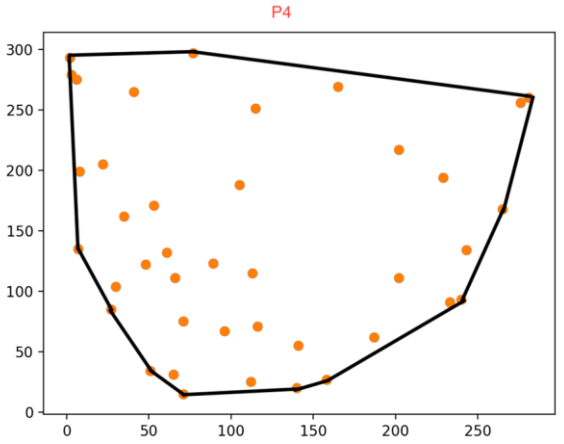
Image Example







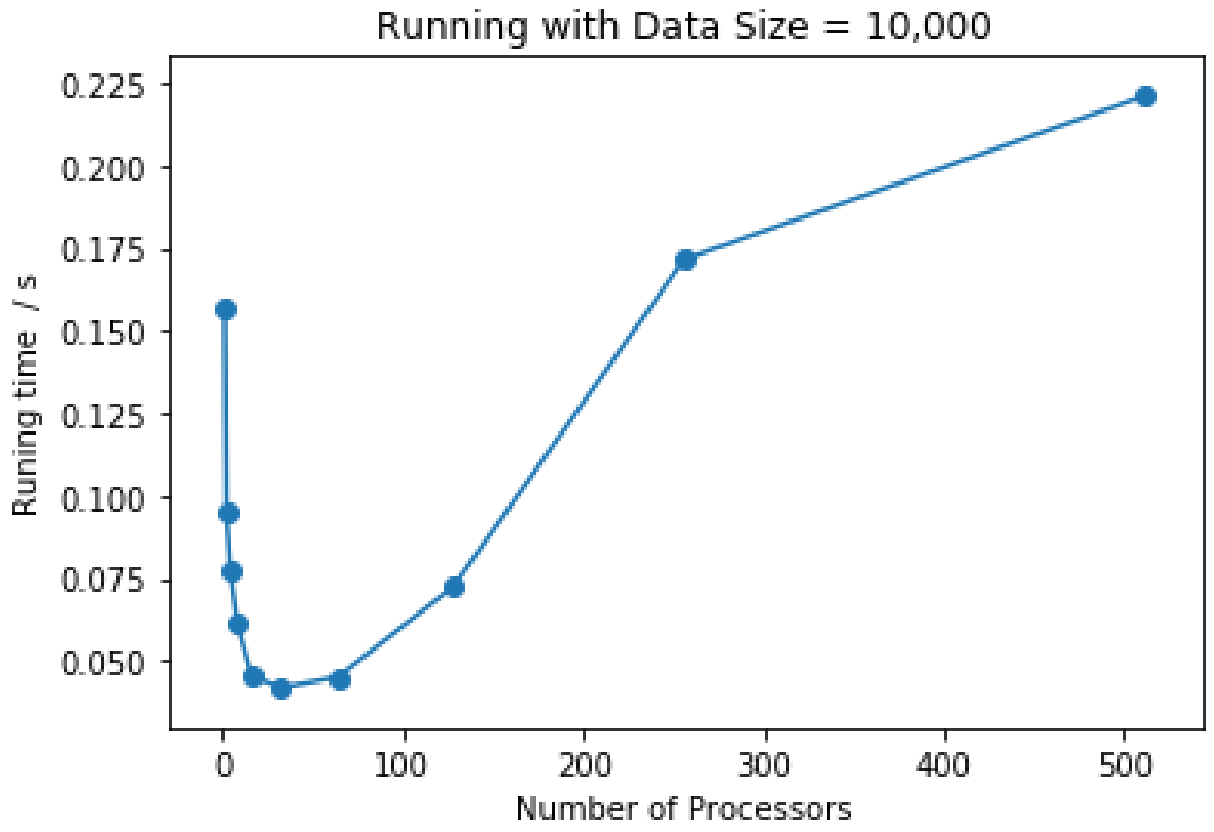




Running Time Analysis

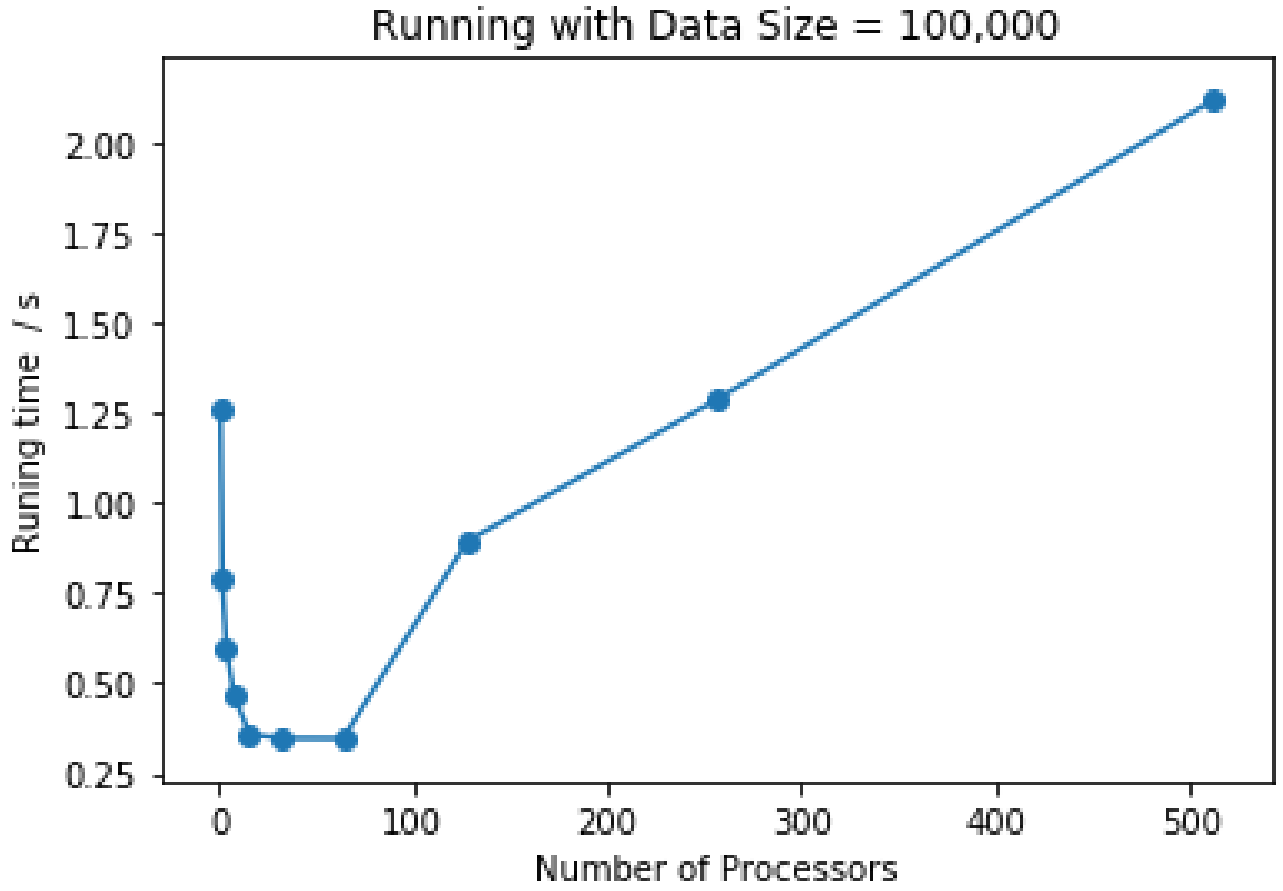
- Running Time vs. # processor with 10,000 points
- Running Time vs. # processor with 100,000 points
- Running Time vs. # processor with 1,000,000 points
- Running Time vs. # processor with 10,000,000 points
- Speedup Rate vs. # processor with four different size data

Graph of Running Time with Data Size = 10,000



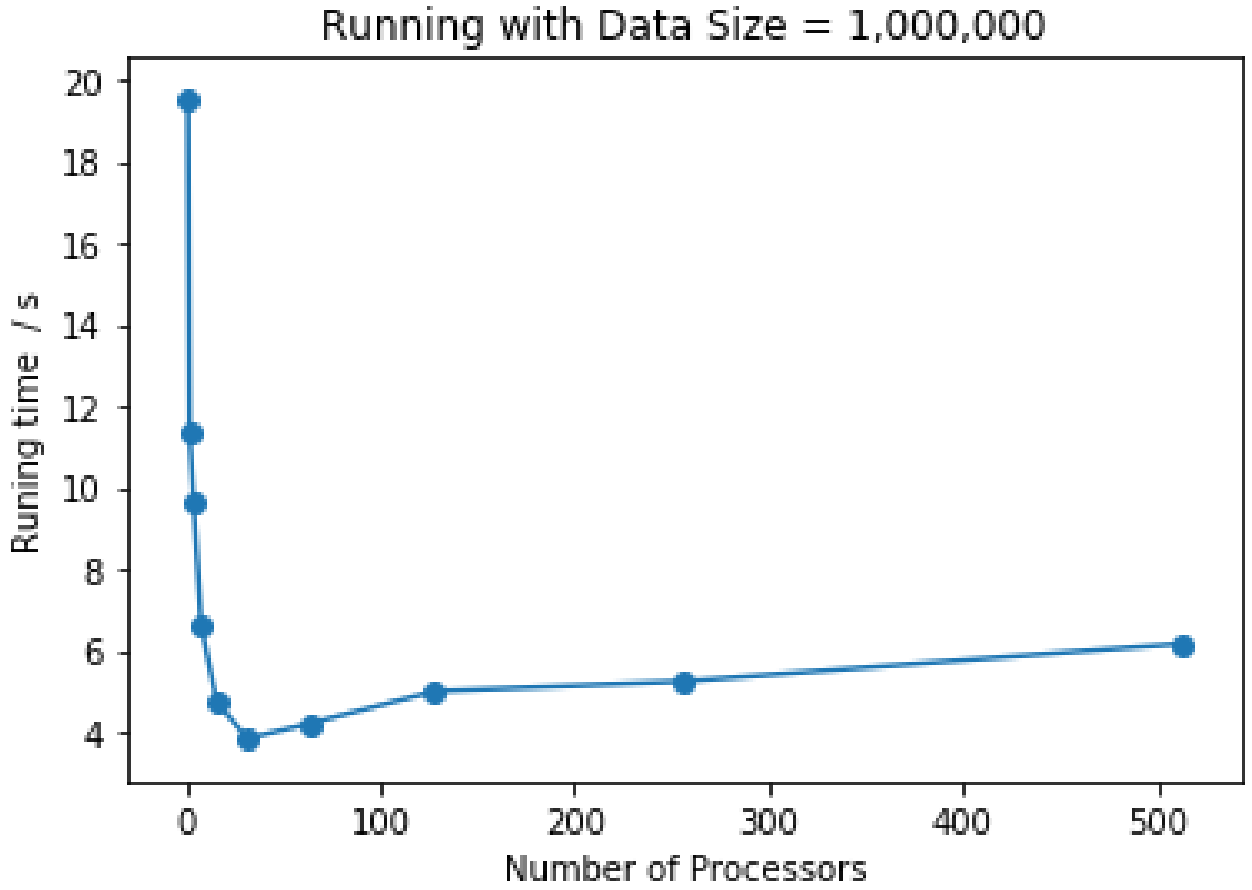
Processor #	Time(sec)
1	0.1565
2	0.0947
4	0.0775
8	0.0616
16	0.0456
32	0.0419
64	0.0452
128	0.0725
256	0.1715
512	0.2211

Graph of Running Time with Data Size = 100,000



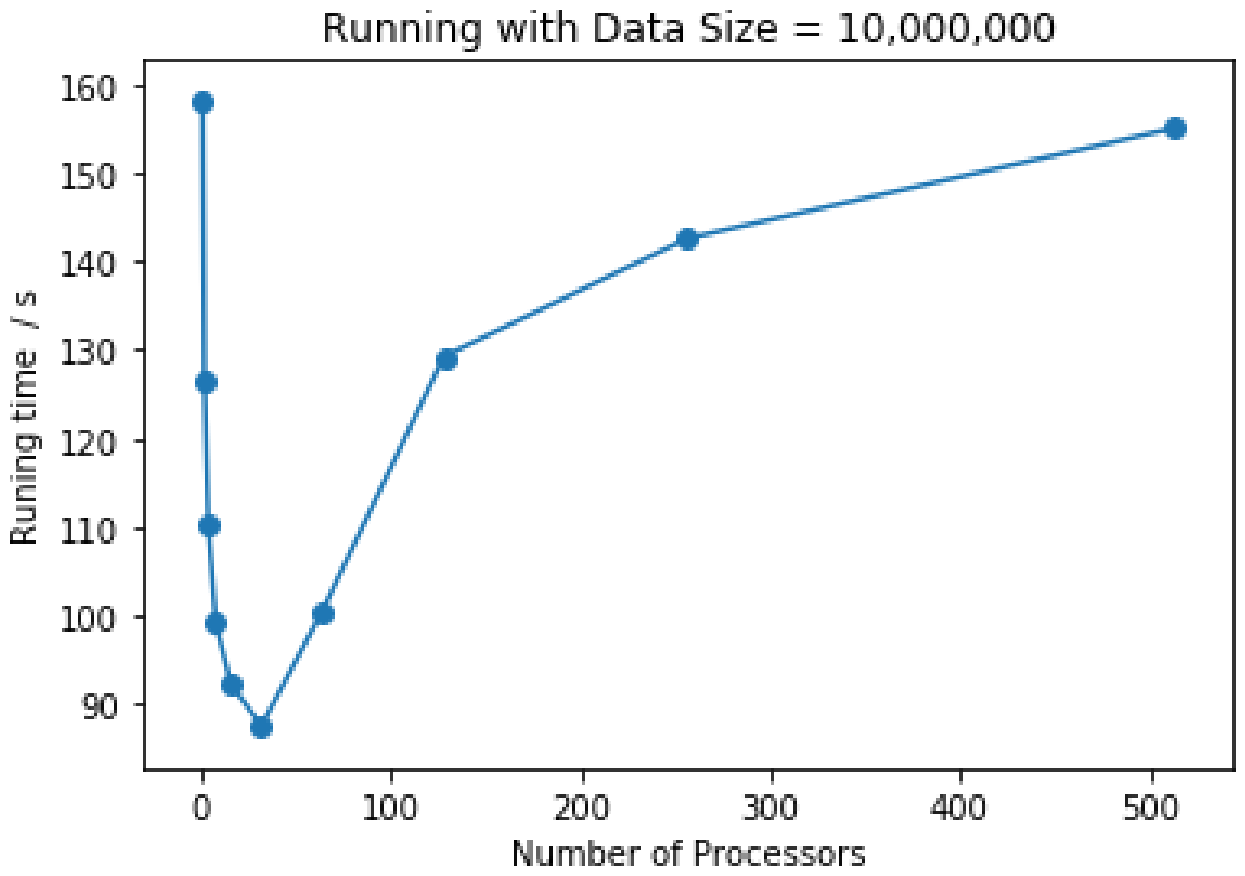
Processor #	Time(sec)
1	1.260
2	0.790
4	0.600
8	0.466
16	0.359
32	0.348
64	0.347
128	0.895
256	1.287
512	2.121

Graph of Running Time with Data Size = 1,000,000



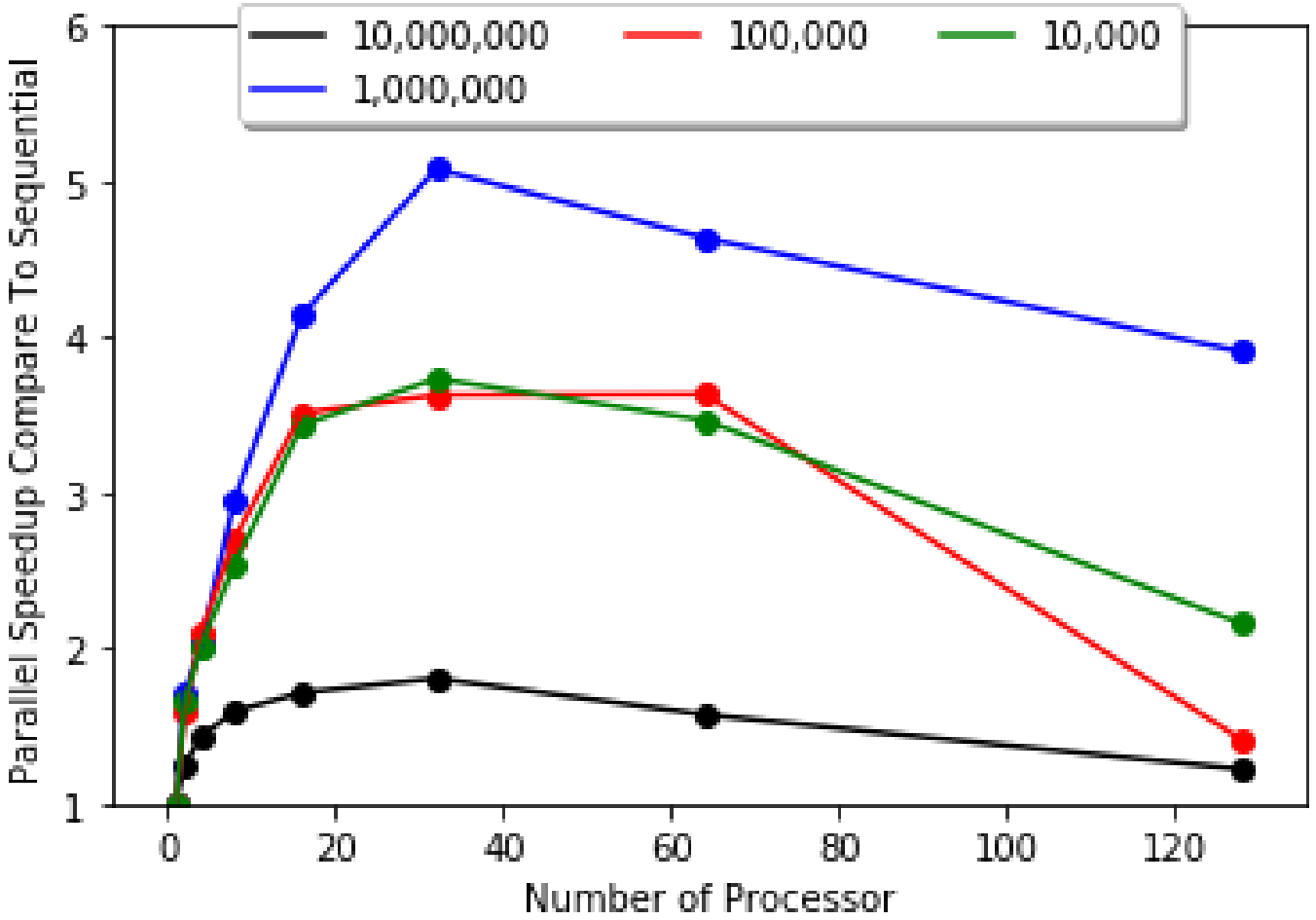
Processor #	Time(sec)
1	19.498
2	11.371
4	9.619
8	6.590
16	4.710
32	3.839
64	4.205
128	4.992
256	5.234
512	6.155

Graph of Running Time with Data Size = 10,000,000



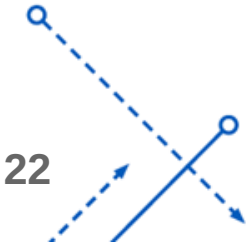
Processor #	Time(sec)
1	158.06
2	126.20
4	110.33
8	99.12
16	92.24
32	87.52
64	100.53
128	129.25
256	142.58
512	155.02

Speedup


10,000,000
100,000
10,000
1,000,000
Parallel Speedup Compare To Sequential
Number of Processor
1
2
3
4
5
6
0
20
40
60
80
100
120

Conclusion

- As observation, more processors does not mean better running time performance.
  - Running Time decreasing when we adding processor before 32 processor.
  - Running Time increasing when continue adding processor after 32 processor.
- Parallel algorithm can save time a lot. When we have 10,000,000 size data. Although the best speedup is about 180% compare to sequential running time, it save about 70 sec in real time.



Future Work

For now. I implemented the Tree Structure Parallel Algorithm which only accept the number of processor is equal to power of 2. And in the future I will implement the Mesh Structure Parallel Algorithm to accept more different number of processor.

Reference

- Russ Miller; Laurence Boxer : Algorithms Sequential and Parallel: A Unified Approach (Third Edition)
- Russ Miller and Quentin F. Stout : Efficient Parallel Convex Hull Algorithm ; IEEE Transaction on Computers vol. 37 no. 12; December 1988.
- Pascal Sommer; A gentle introduction to the convex hull problem. Dec 10, 2016 https://medium.com/@pascal.sommer.ch/a-gentle-introduction-to-the-convex-hull-problem-62dfcabee90c
- MPI For Python : https://mpi4py.readthedocs.io/en/stable/tutorial.html
- CCR Tutorial : https://ubccr.freshdesk.com/support/solutions

Thank You!