University at Buffalo The State University of New York

CSE 633: Parallel Algorithms
Spring 2014

Parallel Algorithms K – means Clustering

Final Results

By: Andreina Uzcategui

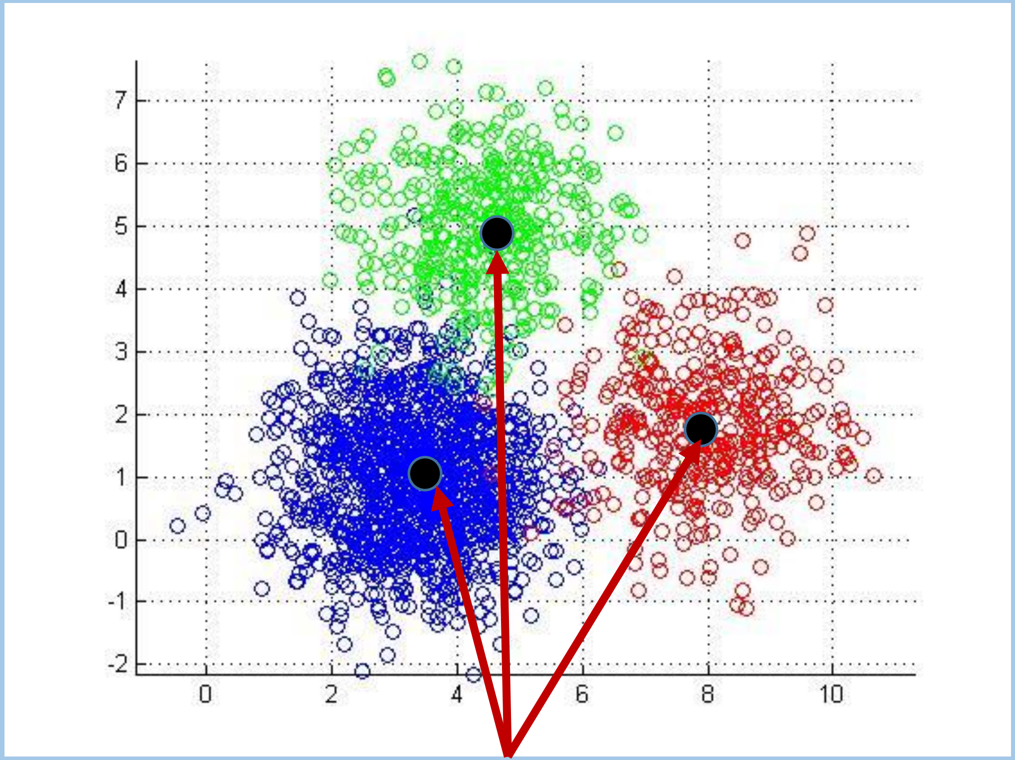
Outline

- The problem
- Algorithm Description
- Parallel Algorithm Implementation(MPI)
- Test Cases
- Results

The Problem

K-means Clustering

Dividing a large vector filled of points into smaller groups which are organized according to a centroid point, each group must have almost the same number of components.



Centroids (k)

Algorithm Description

K – means clustering

- It has by objective to partition n elements into k clusters.

- The partition is made grouping the observed elements according to it proximity with one of the k elements using as centroids.

- The distance between a centroid (k) and a point is calculated by:

- Euclidean Distance Metric:

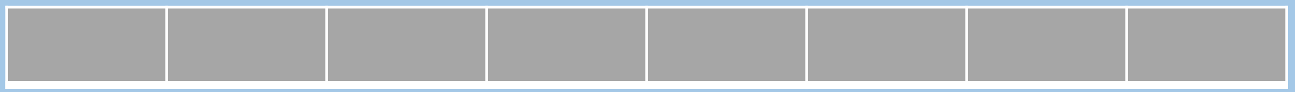
Point – K = |Distance| *(Absolute value result)*

Parallel Algorithm Implementation (MPI)

➡ *In order to make the k – means clustering problem parallel, the following steps will be implemented:*

Data organization

P1 … Pn



T1 **…** **Tn**

1- P processors, each will contain nxTn data values (points) randomly assigned.

2- Three k values (centroids) will be used in each iteration to determinate the clusters.

Parallel Algorithm Implementation (MPI)

Algorithm

- Iterative algorithm

1- For the first iteration 3 k values (centroids) will be determinate randomly.

2- Each PE in parallel will calculate the clusters associated to each k using the Euclidean Distance Metric.

Parallel Algorithm Implementation (MPI)

3- Each PE in parallel will calculate the median value of each of its cluster.

- Media:

1- Determinate a frequency table containing each point in the cluster frequency.

2- Calculate the media position according to the frequency table and hence the median value will be obtained.

Parallel Algorithm Implementation (MPI)

4- Each PE will broadcast its medians for each cluster to all other PEs.

5- In parallel each PE will determinate a new median for each cluster using the received data and its just calculated median.

6- Each PE will check for each cluster the different between the new calculated median and it previous calculated median.

Parallel Algorithm Implementation (MPI)

Final Conditions

- When the different between old and new median (error value) is minimal or zero the iteration process stops under normal considerations.

- For simplicity of the algorithm, in this case the number of iterations made was predetermined to avoid infinite iterations (10 itarations).

- For each iteration (except first one) the K values will be the closest medians to 0 determinate in previous iteration.

Test Cases & Conclusions

1- Same centroids, different data, same # processors, same # tasks.

2- Same centroids, same data, different # processors.

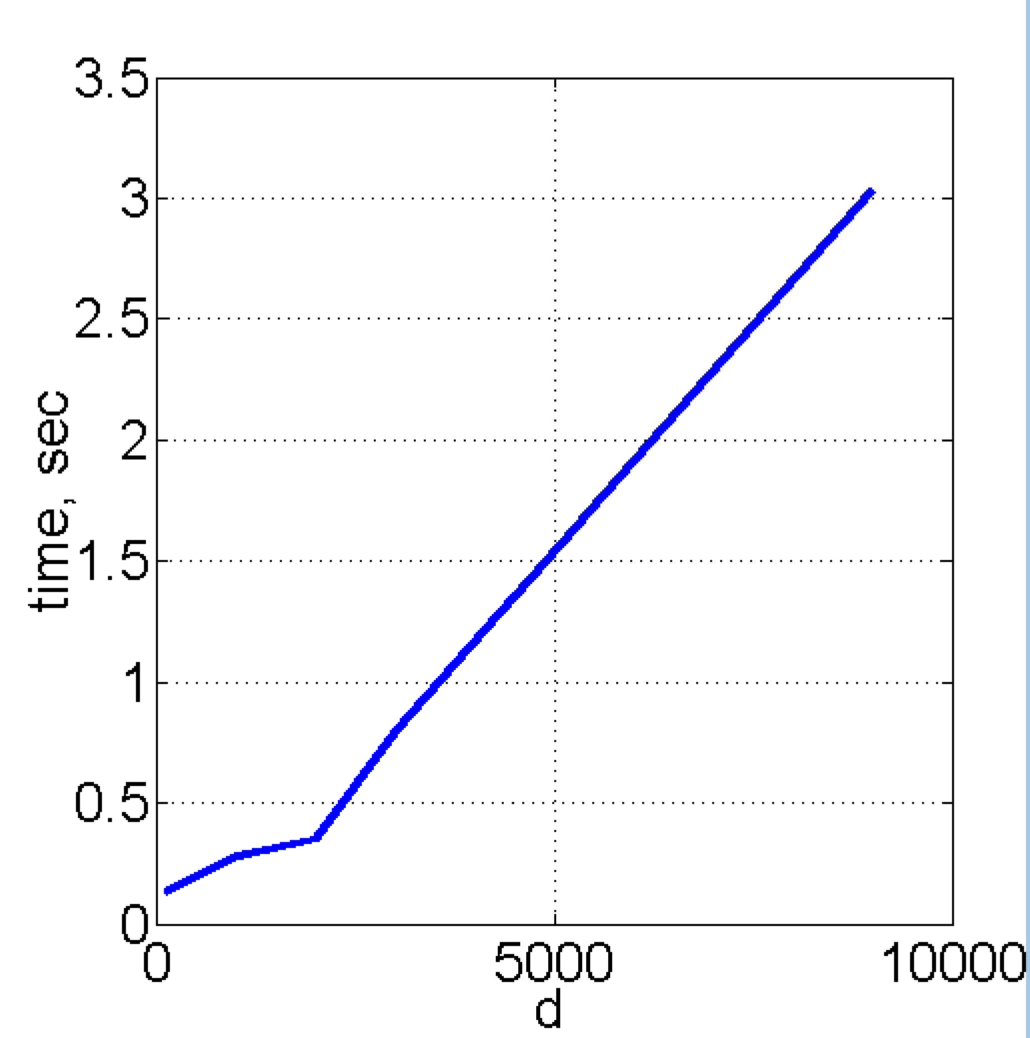
3- Same centroids, same data, different # tasks.

4- Different centroids, different data, different # processors.

5- Different centroids, different data, different # tasks.

6- Same data, different # processors.

<u>Test Case 1</u>: Same centroids, different data, same # processors, same # tasks.



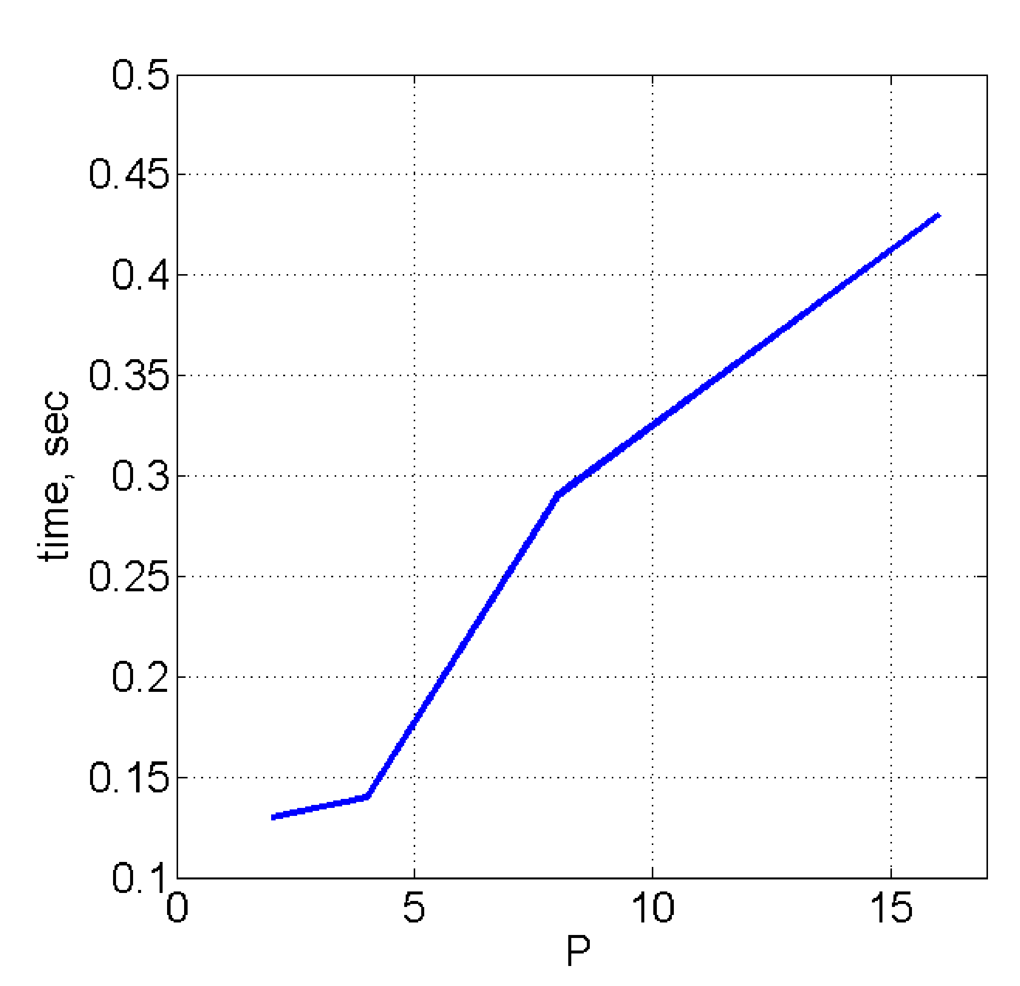
K	d	P	T	Time
3	100	2	8	0.13
3	1000	2	8	0.28
3	2000	2	8	0.35
3	5000	2	8	0.80
3	9000	2	8	3.03

K = # centroids
d = # data
P = # processor
T = # tasks

<u>Conclusion</u>

The processing time ***dramatically increase.***

__Test Case 2:__ Same centroids, same data, different # processors.



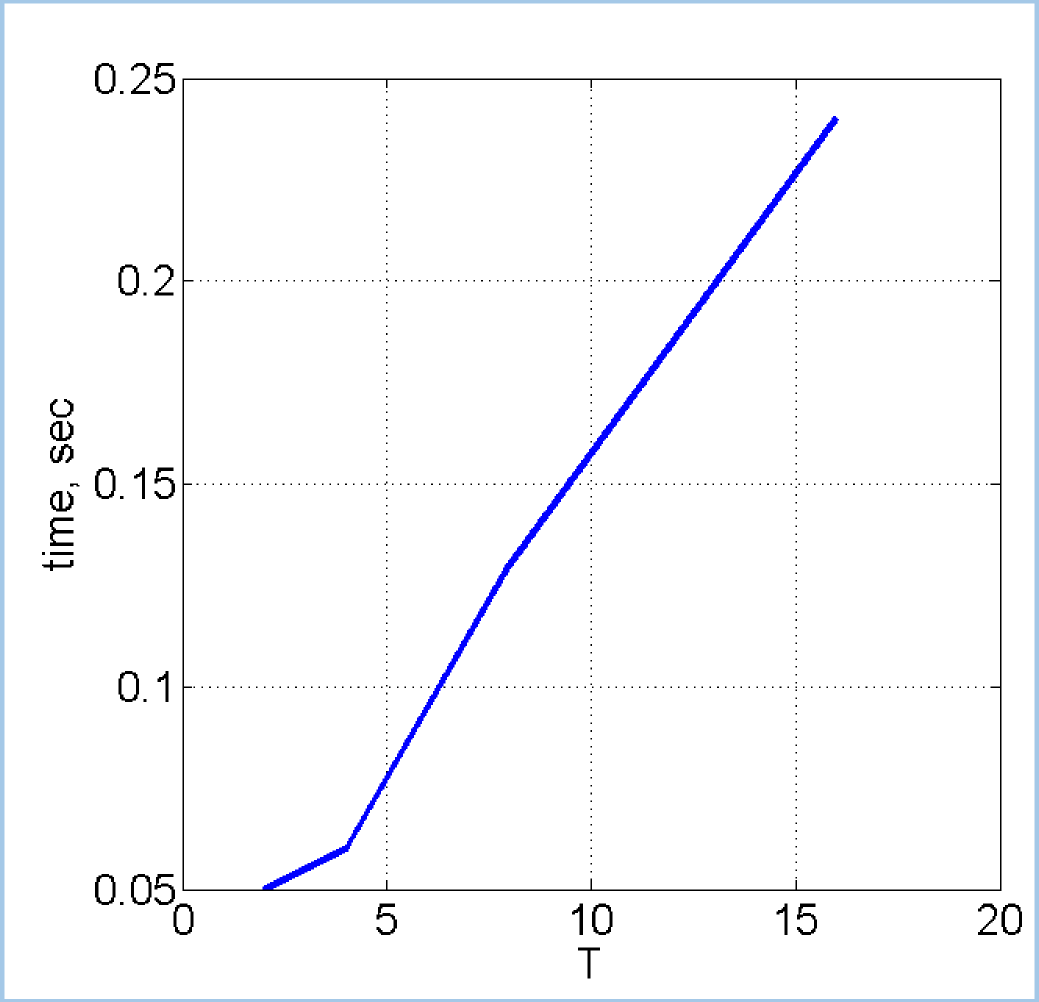
K	d	P	Time.sec
3	100	2	0.13
3	100	4	0.14
3	100	8	0.29
3	100	16	0.43

K = # centroids
d = # data
P = # processor

__Conclusion__

The processing time slowly increase.

Test Case 3: *Same centroids, same data, different # tasks.*



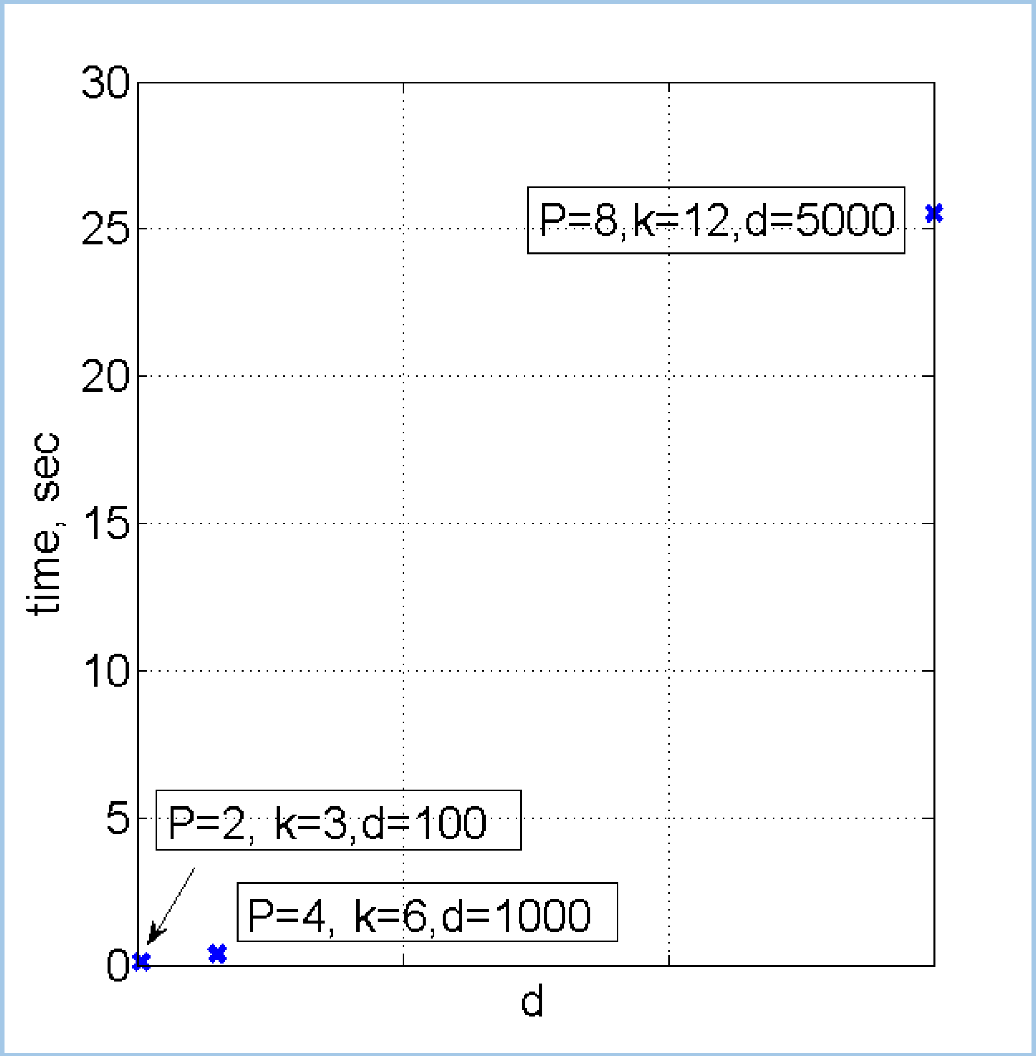
K	d	T	Time
3	100	2	0.05
3	100	4	0.06
3	100	8	0.13
3	100	16	0.24

K = # centroids
d = # data
T = # tasks

Conclusion

The processing time slowly increase.

Test Case 4: *Different centroids, different data, different # processors.*



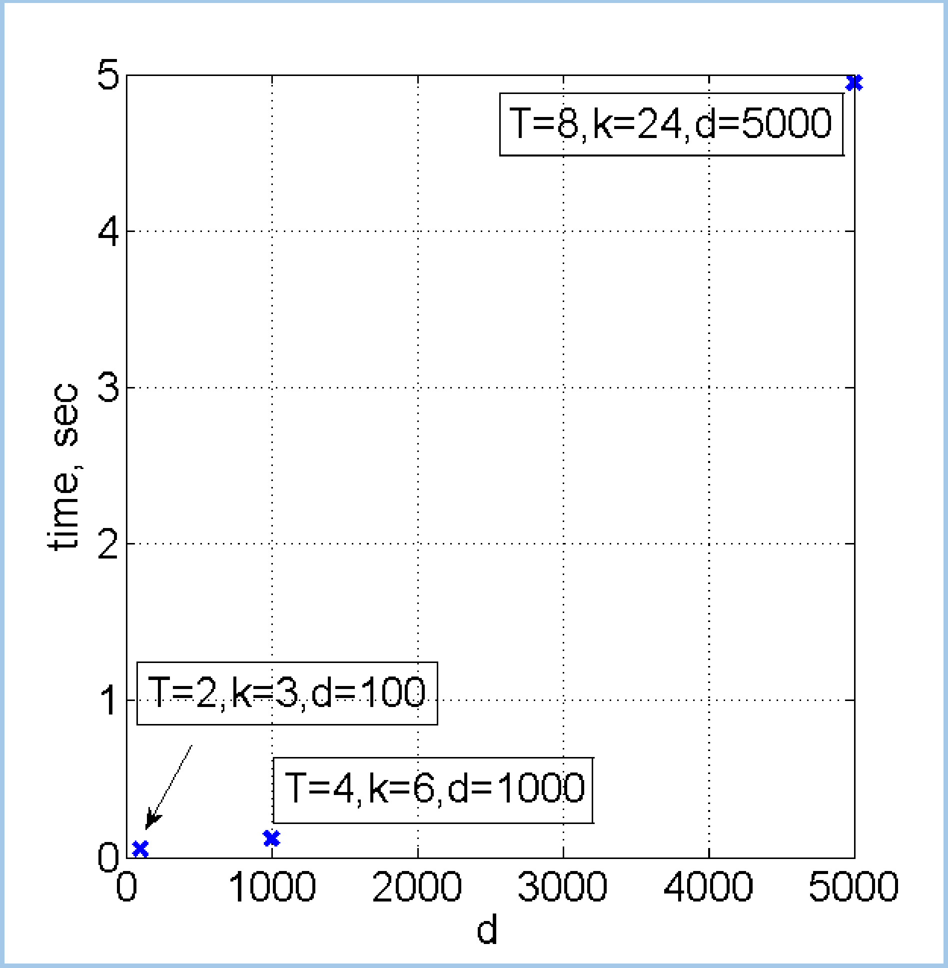
K	d	P	Time
3	100	2	0.1
6	1000	4	0.35
12	5000	8	25.54

K = # centroids
d = # data
P = # processor

Conclusion

The processing time ***dramatically increase.***

Test Case 5: *Different centroids, different data, different # tasks.*



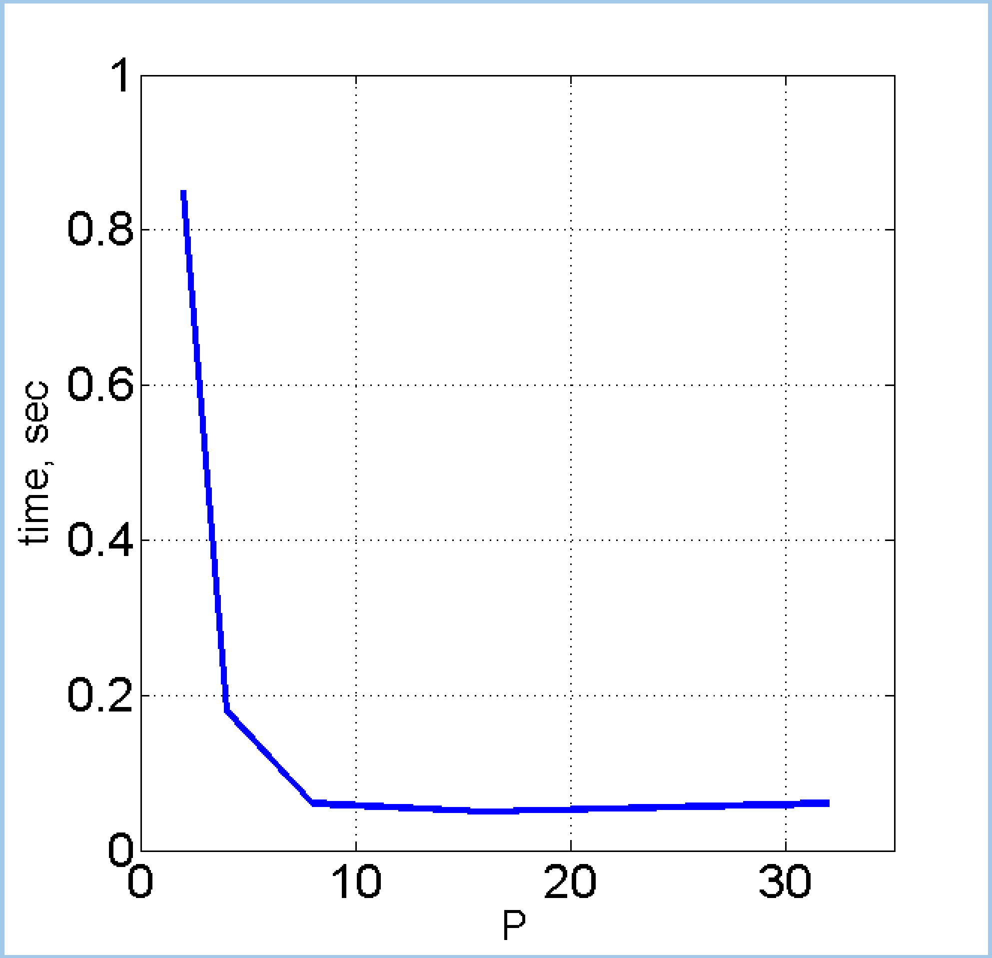
K	d	T	Time
3	100	2	0.05
6	1000	4	0.12
12	5000	8	4.95

K = # centroids
d = # data
T = # tasks

Conclusion

The processing time ***dramatically increase.***

__Test Case 6:__ Same data, different # processors.



P	time, sec
2	0.85
4	0.18
8	0.07
16	0.05
32	0.06

Total data, N = 12288, is divided by an increasing P in every stage

__Conclusion__

*The processing time **slowly decrease** until the # processors is to high and the data per P is too low.*

Questions?