

# Approximating the Temperature Distribution with K means algorithm using MPI

BY

MANGESH VILAS KASLIKAR


VIVEK RAGHUVANSHI

KARTHICK KRISHNA VENKATAKRISHNAN

MOHAN UPADHAYA

# We have the Raw Data



- Data is available for Temperature in detail for USA .
  - Our focus right now is on all of the 50 states of the US.
  - It includes daily summaries of climate parameters such as maximum and minimum temperature for one year from Jan 1 2011 to Dec 31 2011.
- 

# Idea

- We plan to use **K - Means algorithm** on Temperature parameter for the data available on 50 states of USA for the year 2011 -2012.

# K-Means Clustering

- Initially, the number of clusters must be known, or chosen, to be  $K$  say.
- The initial step is to choose a set of  $K$  instances as centres of the clusters. Often chosen such that the points are mutually “farthest apart”, in some way.
- Next, the algorithm considers each instance and assigns it to the cluster which is closest.
- The cluster centroids are recalculated either after each instance assignment, or after the whole cycle of re-assignments.
- This process is iterated till either

Convergence is reached i.e. there is not much difference between centroids obtained in previous iteration and current iteration

Or

Maximum number of iterations is reached

# K – Means

- Algorithm used for clustering similar data together.
- Choosing correct value of K is critical
- No fixed value, but depends upon distribution.
- Complexity in general ,  $O(n^{dk+1} \log n)$  , where n - Number of entities, d : Number of Dimensions, K : Number of clusters.
- Our project : D = 1 , and K = 4 ( default ) , but may change as per data and distribution.

# MPI – K- Means – R Connection

1. `main [switches] -i filename -n num_clusters`

Where `-i filename` : file containing data to be clustered

`-n num_clusters` : number of clusters (K must > 1)

2. R Package with MPI : `Rmpi` provides an interface (wrapper) to MPI APIs. It also provides interactive R slave environment.

# Algorithm

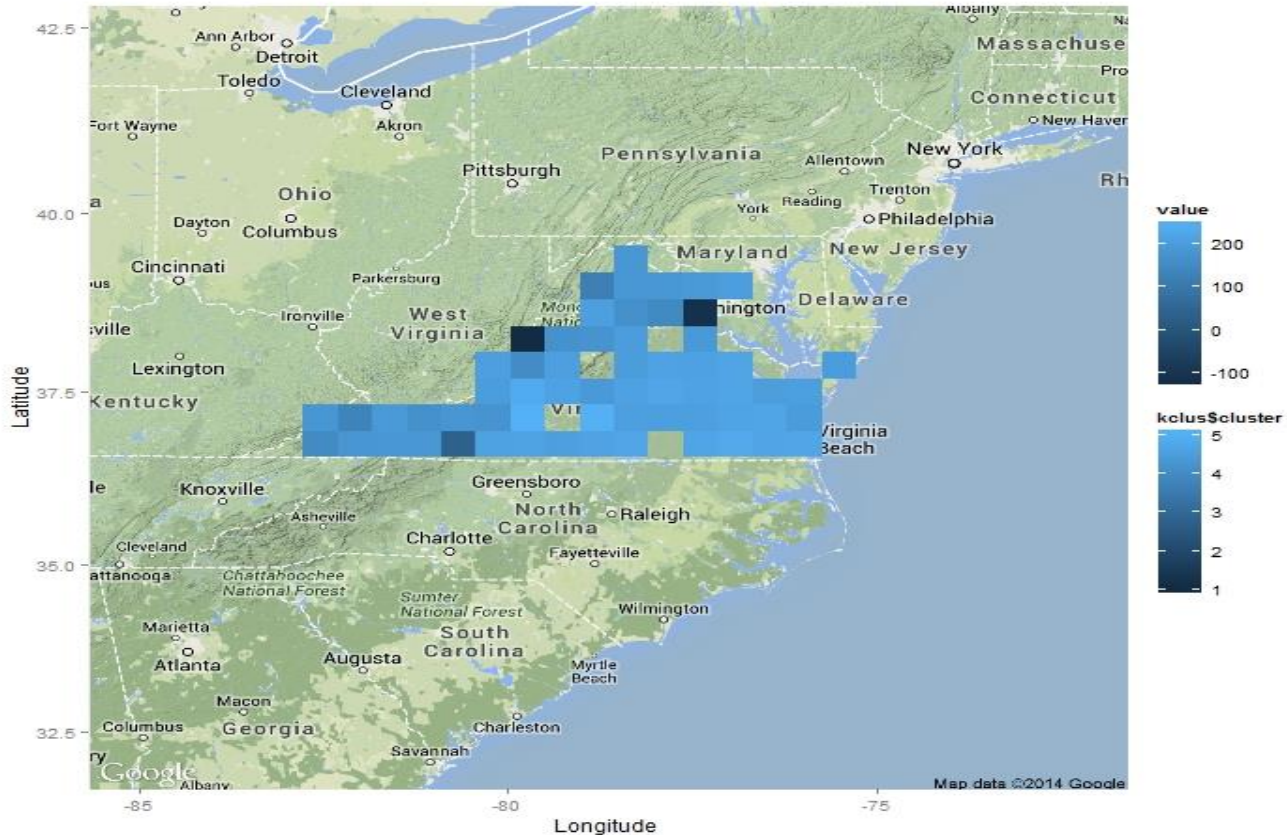
## MASTER :

- 1) Broadcast path of input files to workers
- 2) Gather computational output from workers
- 3) Produce Kmeans

## WORKERS :

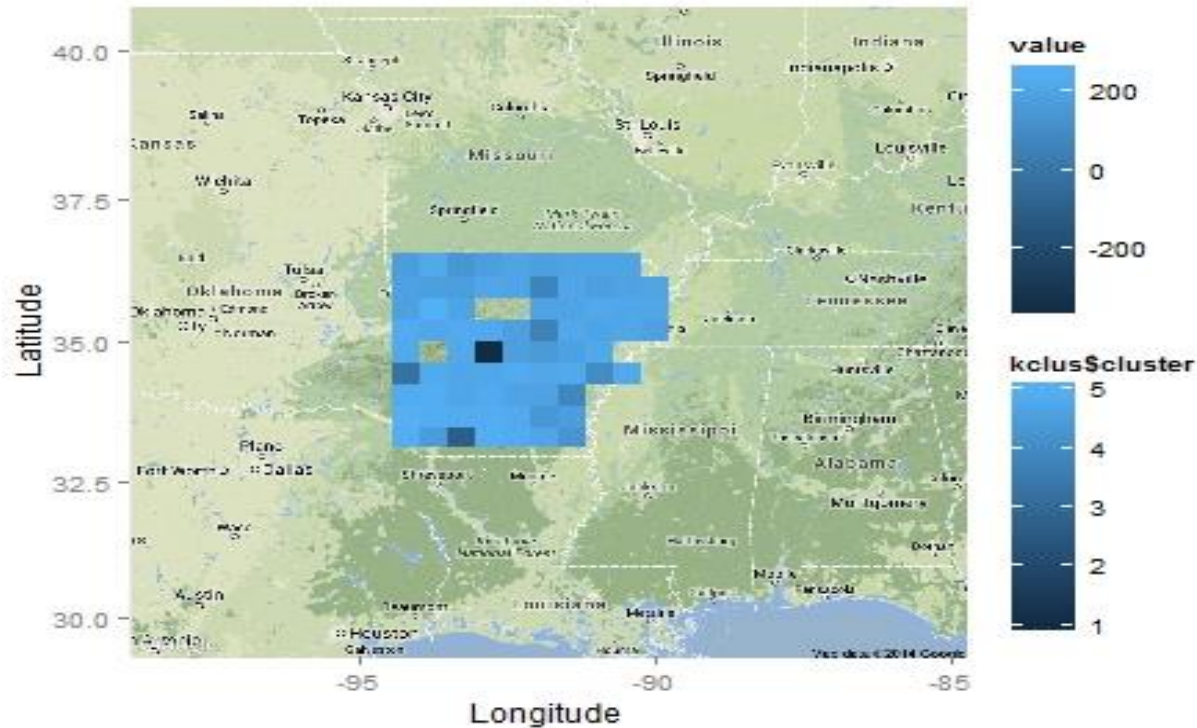
- 1) Fetch input data from master.
- 2) Perform computation on input data.
- 3) Send output to master.

# Output Received – Virginia State

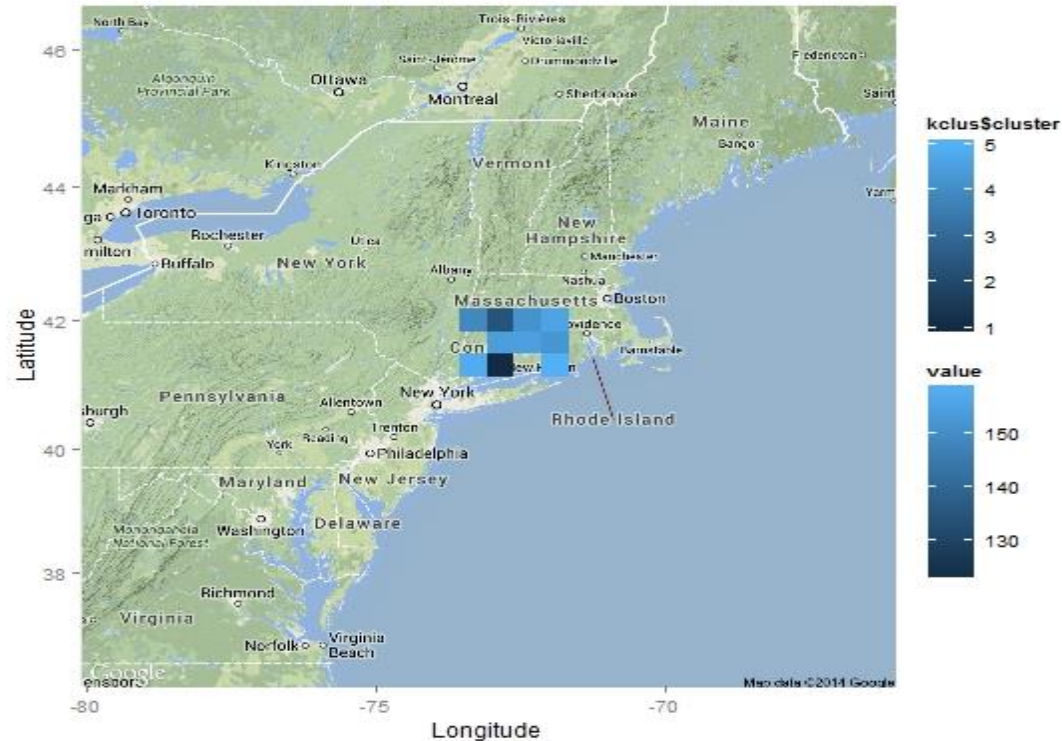




# Output Received – Arkansas State



# Output Received – Connecticut State



# Test Results

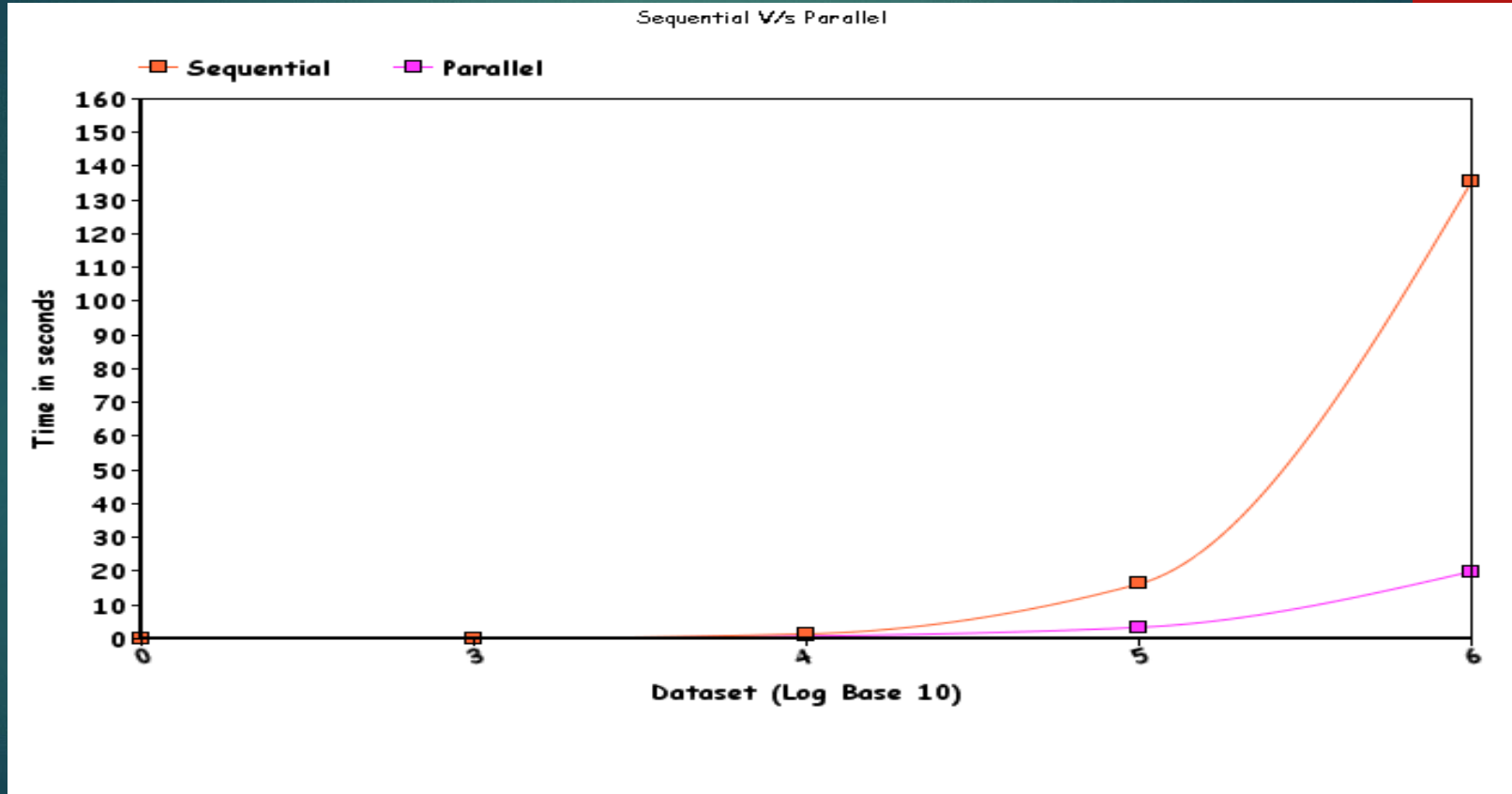
1) Number of data points tested were 1000, 10000, 100000, 1000000

2) The following aspects were analyzed

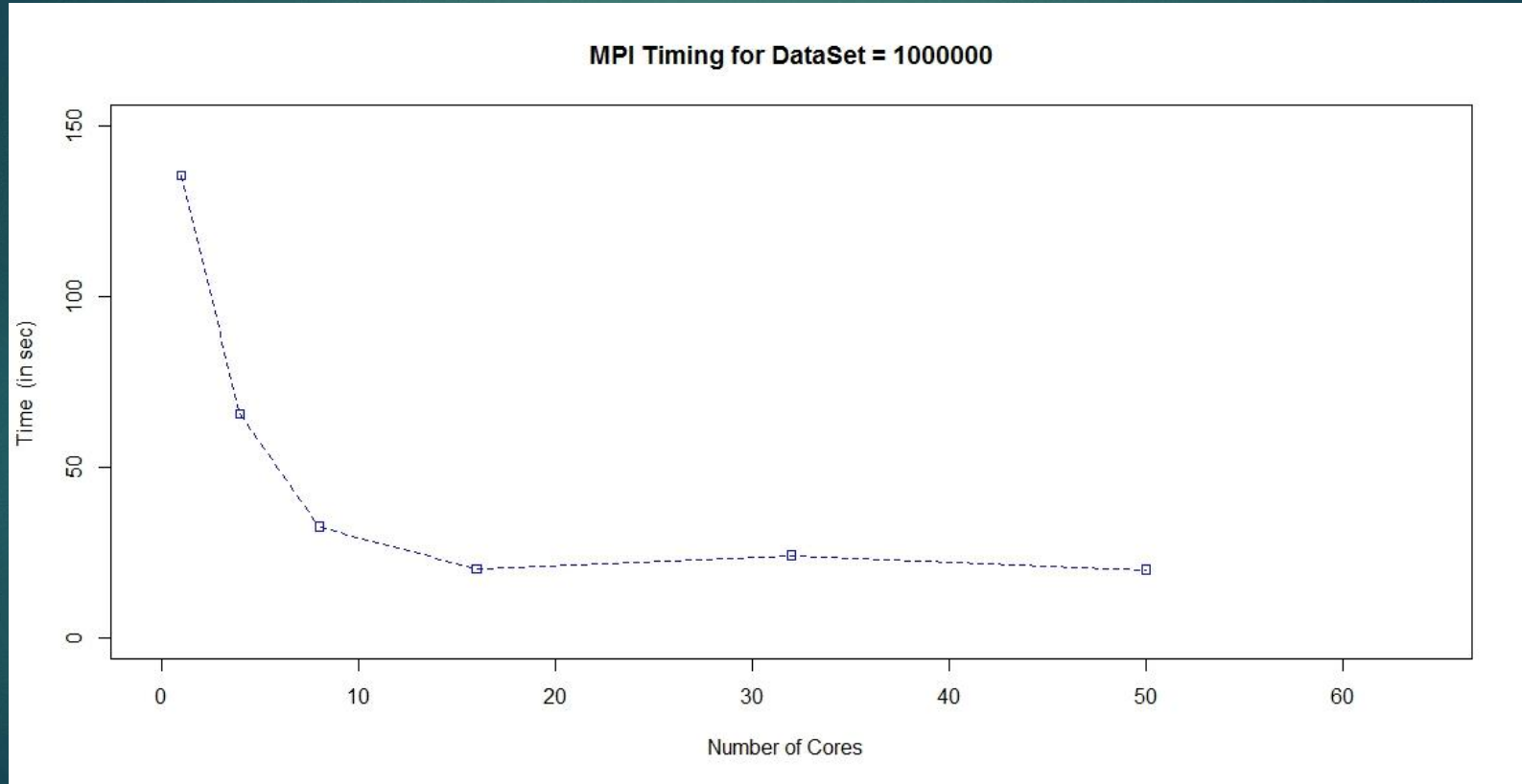
( i ) Speedup

( ii ) Timing

# Graphs 1 - Sequential vs Parallel

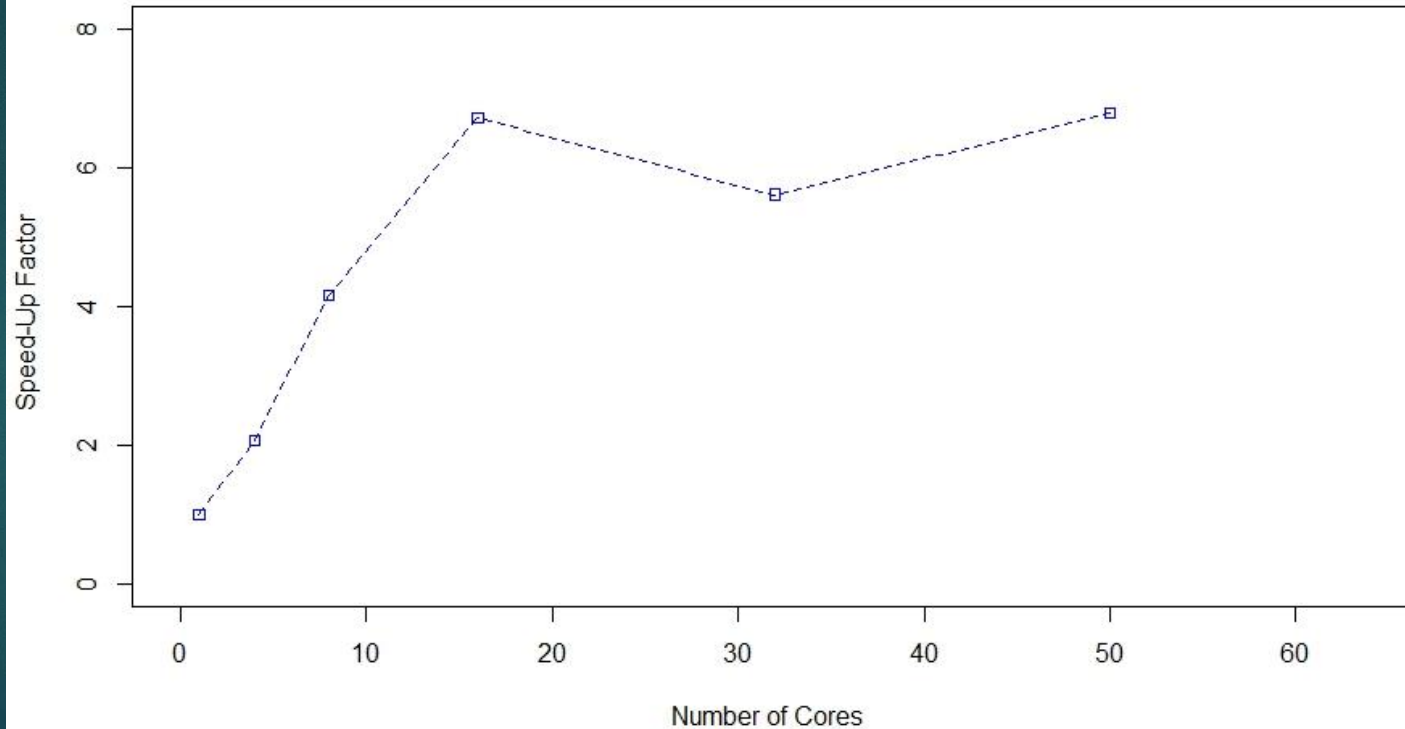


# Graphs 2 – MPI Timing for DataSet 1000000



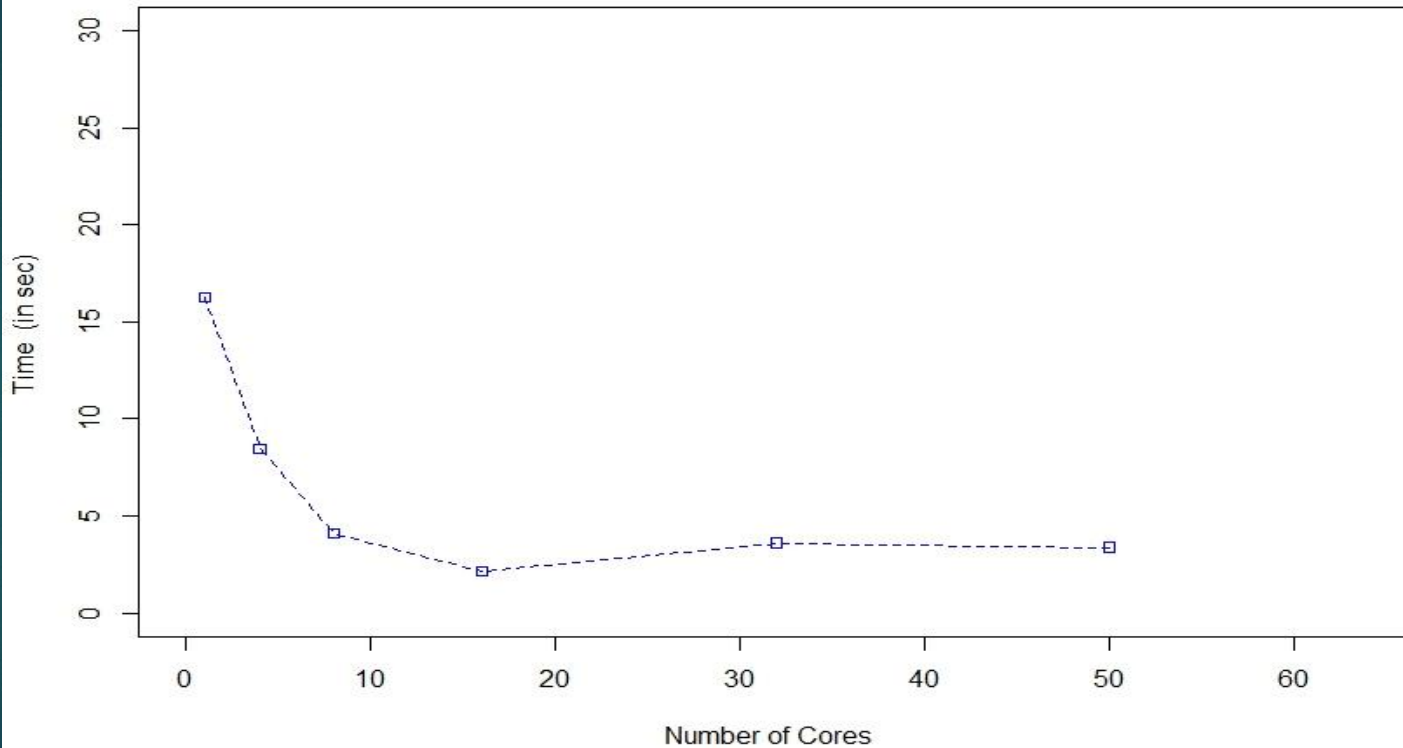
# Graphs 2 – Speed Up DataSet 1000000

MPI SpeedUp for DataSet = 1000000



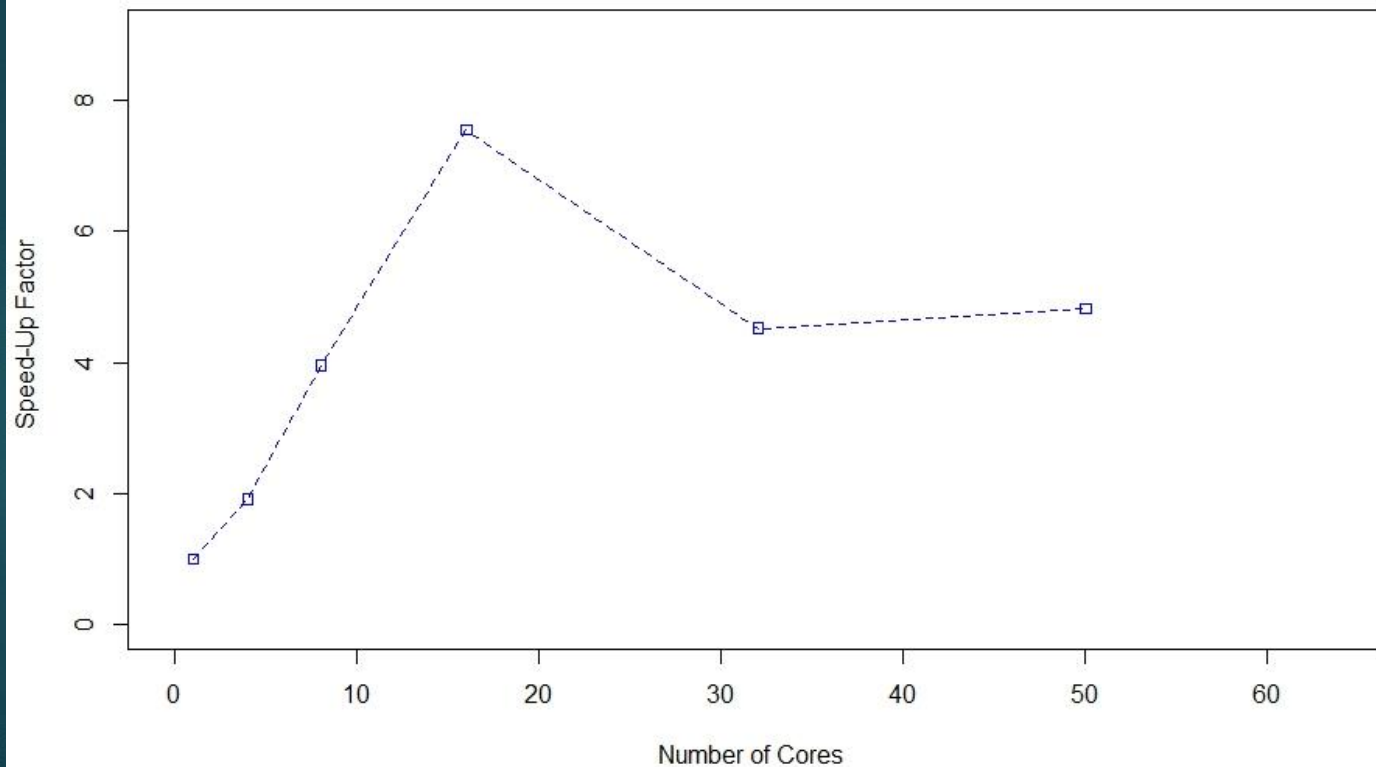
# Graphs 3 – MPI Timing for Data 100000

MPI Timing for DataSet = 100000



# Graphs 3 – Speed Up for DataSet 100000

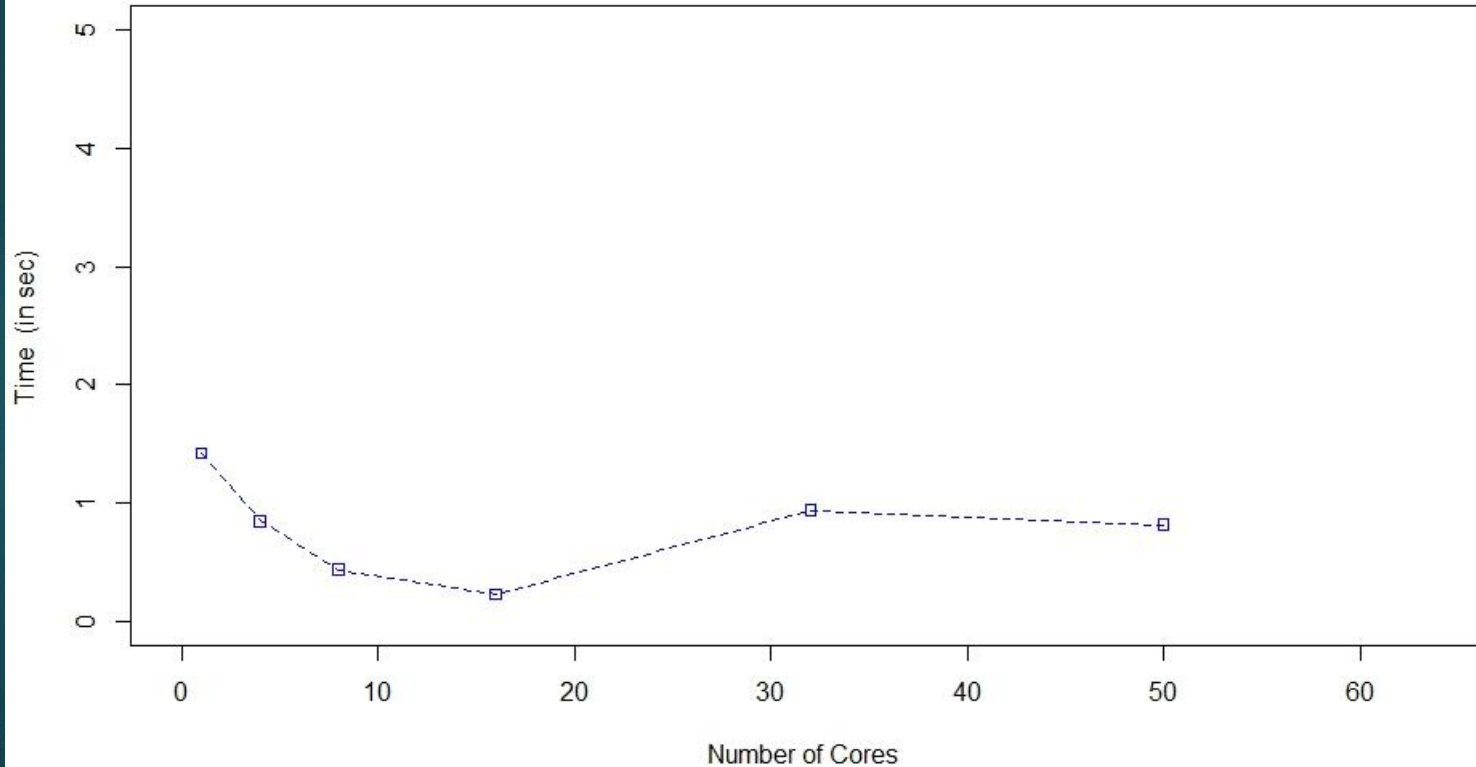
MPI SpeedUp for DataSet = 100000





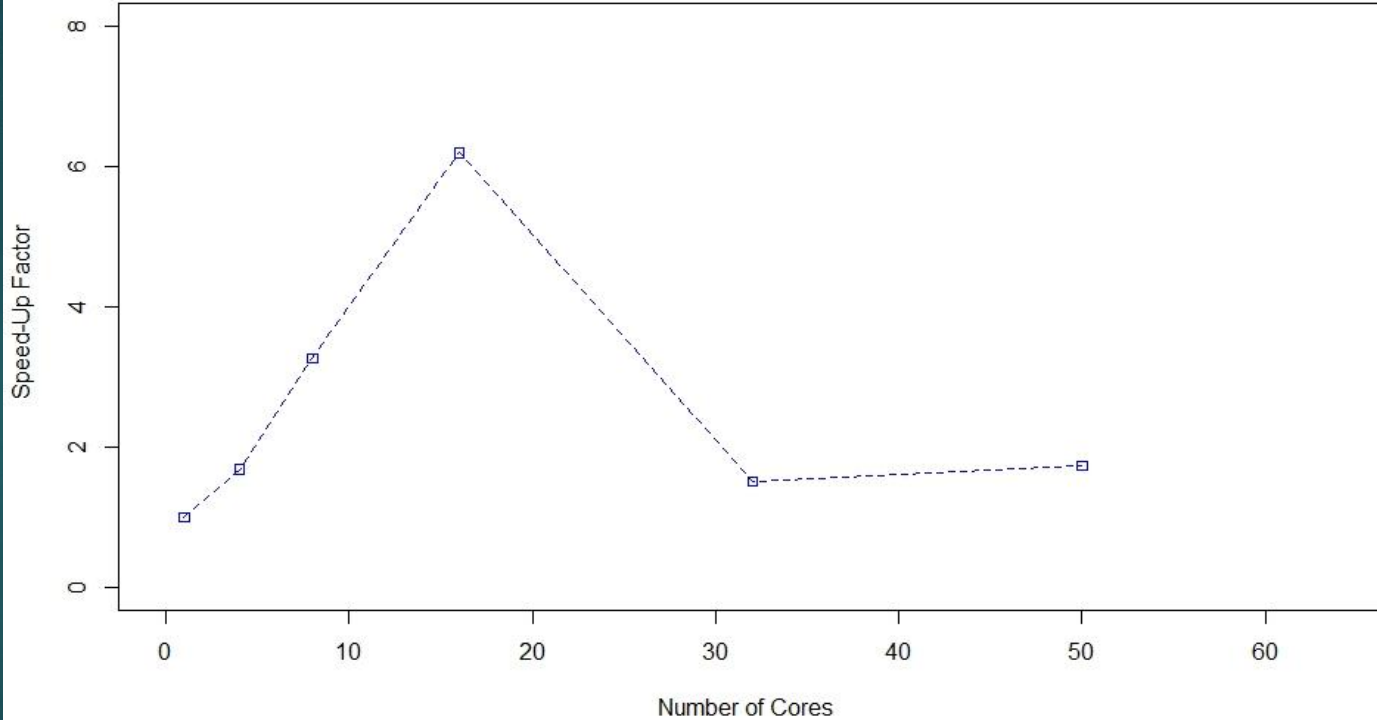
# Graphs 4 – MPI Timing for Data 10000

MPI Timing for DataSet = 10000



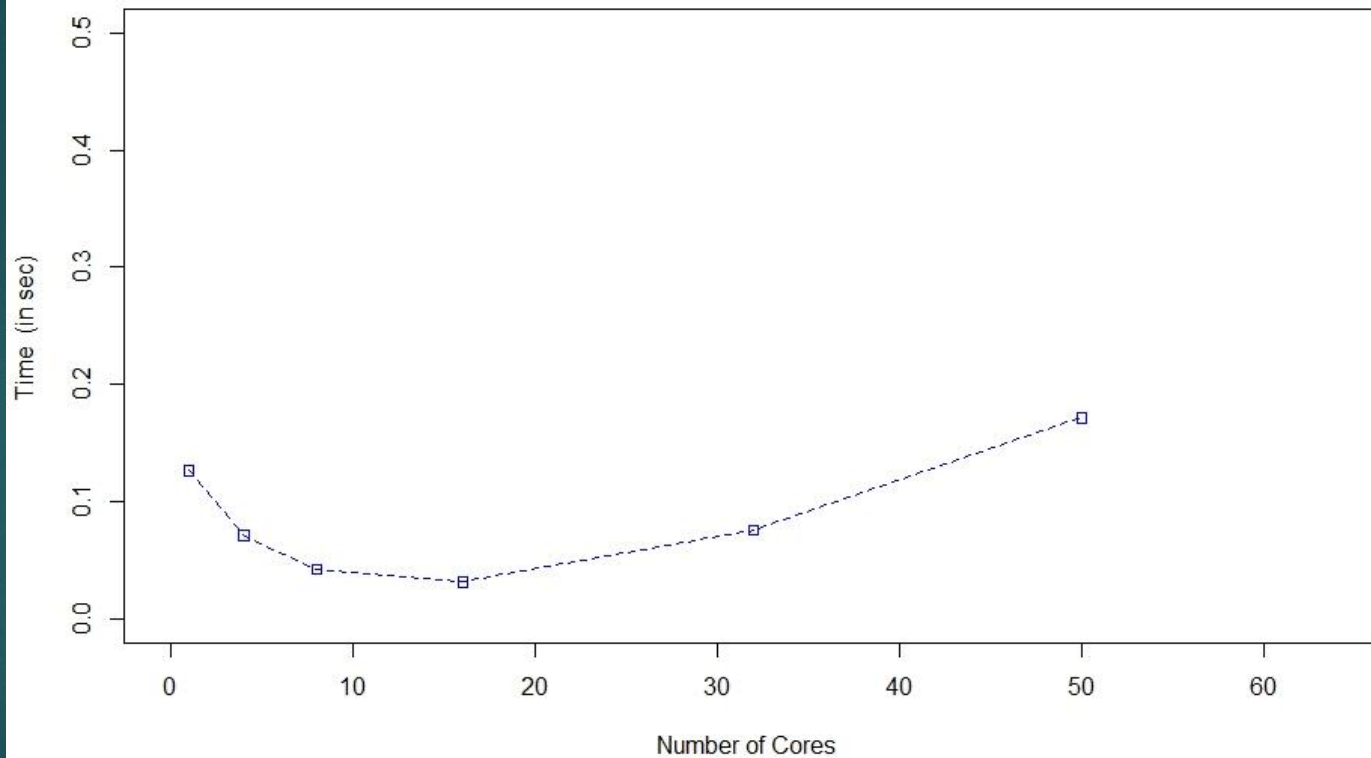
# Graphs 4 – Speed Up for DataSet 10000

MPI SpeedUp for DataSet = 10000

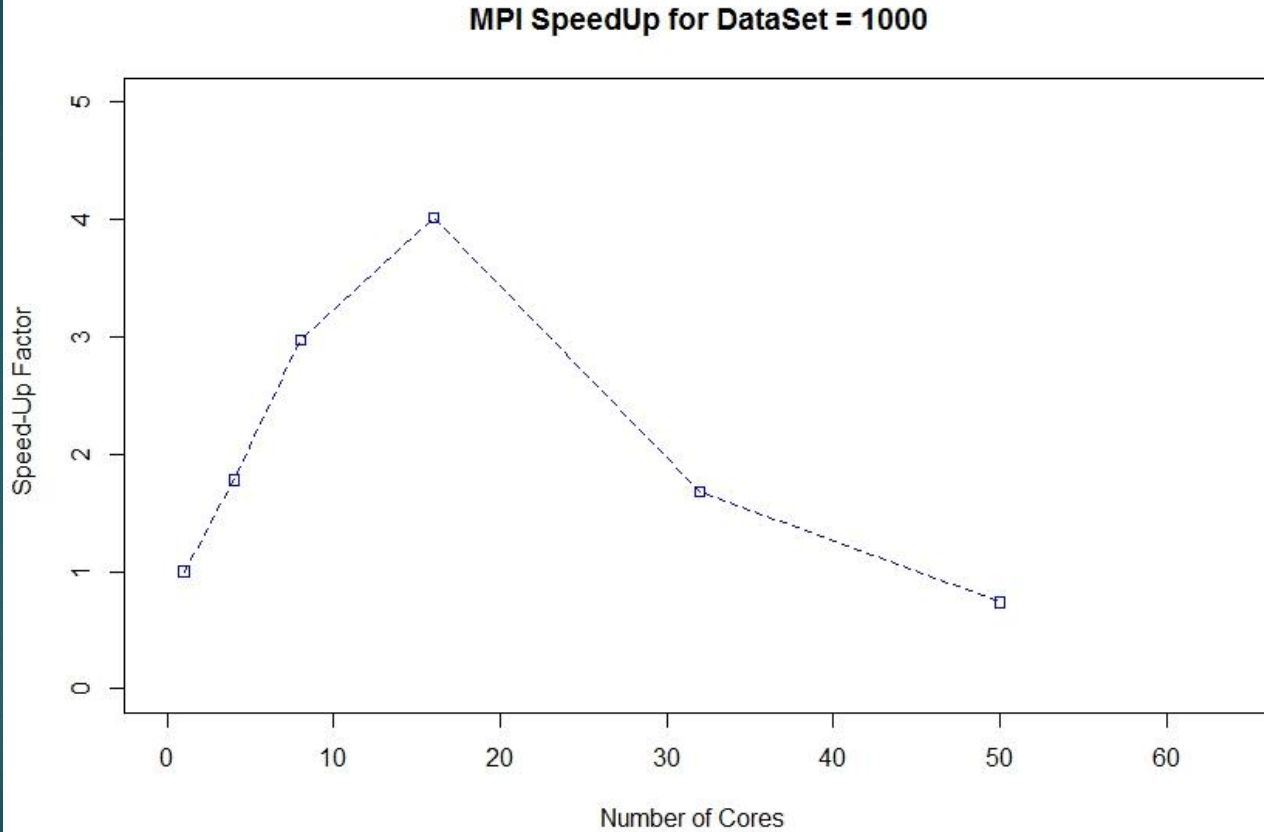


# Graphs 5 – MPI Timing for DataSet 1000

MPI Timing for DataSet = 1000



# Graphs 5 – Speed Up for DataSet 1000



# Data Table

1. MPI Timing for Different DataSets vs Number of Cores . The Time is measured in seconds.

Cores →	1	4	8	16	32	50
<b>Data</b>						
1000	0.12676	0.0712	0.0426	0.03156	0.07544	0.17156
10000	1.42116	0.84692	0.43596	0.22948	0.9384	0.81768
100000	16.25996	8.4684	4.11684	2.1542	3.59968	3.37952
1000000	135.43626	65.66148	32.51916	20.14512	24.13796	19.94604

2. MPI Speed-Up for Different DataSets vs Number of Cores.

Speed-Up Factor						
Cores →	1	4	8	16	32	50
<b>Data</b>						
1000	1	1.78	2.9755	4.0164	1.68	0.7388
10000	1	1.678	3.259	6.1929	1.5144	1.738
100000	1	1.92	3.949	7.548	4.517	4.811
1000000	1	2.0626	4.164	6.723	5.6109	6.7901

# Conclusion

- K-Means computation can be parallelized using multiple processors.
- As a result we get a significant reduction in time using parallel computation of K-Means over sequential computation.
- However as the number of processors increase, the overhead of communication between nodes increases as well.
- Thus the performance and speedup reduces due to communication overhead between nodes.

# References

- <http://users.eecs.northwestern.edu/~wkliao/Kmeans/>
- <http://cran.r-project.org/web/packages/Rmpi/index.html>
- [http://en.wikipedia.org/wiki/K-means\\_algorithm](http://en.wikipedia.org/wiki/K-means_algorithm)
- <http://www.ncdc.noaa.gov/cdo-web/datasets>