Abstract Presentation Back to Agenda Next Abstract

A Parallel Data Mining Toolbox Using MatlabMPI

Parna Khot Ashok K. Krishnamurthy Stanley C. Ahalt John W. Nehrbass
Juan C. Chaves
Department of Electrical Engineering
The Ohio State University
2015 Neil Ave
Columbus, OH 43210

Extended Abstract

The ready availability of vast quantities of data has driven the need for data mining algorithms that can discover patterns, correlations and changes in the data. The amount and high dimensionality of the data make data mining an important application for high performance computing [Joshi, 2002]. The mathematical and interactive nature of many of the data mining algorithm, makes it natural to use a language like MATLAB both to design algorithms and for post-processing of the results. Recently, Kepner [2002] has developed a system, called MatlabMPI, which implements the six basic functions of the Message Passing Interface (MPI) standard in MATLAB, and thus allows any Matlab program to exploit multiple processors. This has motivated us to develop a parallel data mining toolbox that is based on MatlabMPI. Implementations of a parallel clustering algorithm and a parallel classification algorithm have been completed, and other functions are currently under development.

We present two parallel implementations of K-Means clustering using MatlabMPI in this poster.

1. **Master-Slave Method.** In this approach there is a main node (Master) that performs data distribution, convergence check and centroid update. The slave processors are used only to calculate the centroids of their own local data. The algorithm is as follows:
    a. The processor with rank 0 distributes the data & initial random centroids to the non-rank 0 processors.
    b. All other processors receive the data and compute the centroids for their local data (using Serial K-Means clustering).
    c. The non-rank 0 processors send their local clustered data to the rank 0 processor.
    d. The rank 0 processor receives the data sent by each processor and recomputes the centroids.
    e. The rank 0 processor checks for convergence condition. If convergence condition is not reached, then it sends the updated centroids to the other processors and steps 2 & 3 are repeated. This process is repeated until the convergence condition is reached. If convergence condition is reached, then the rank 0 node sends the status bit informing the non-rank 0 processors to exit Matlab.

2. **Peer-to-Peer Method.** In this approach the Rank 0 node, after initial data distribution, is used like any other node for clustering data. All the processors (including the main node) inter-communicate to update centroids and check for convergence condition locally. The algorithm is as follows:
    a. The processor with rank '0' distributes the data & initial random centroids to rest of the processors.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE
20 AUG 2004

2. REPORT TYPE
N/A

3. DATES COVERED
-

4. TITLE AND SUBTITLE
A Parallel Data Mining Toolbox Using MatlabMPI

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical Engineering The Ohio State University 2015 Neil Ave Columbus, OH 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**43**	

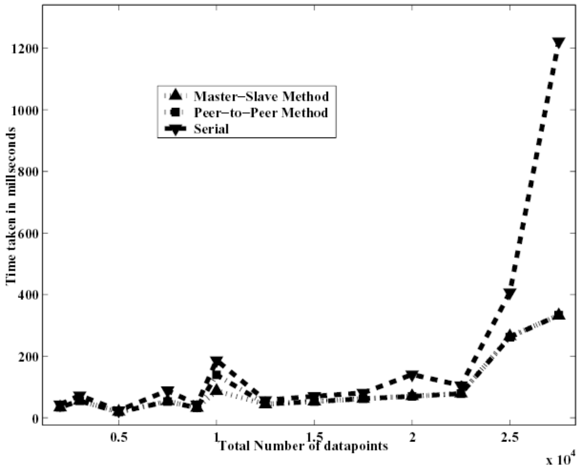
b. All the processors calculate the centroids for their local data, using Serial KMeans clustering.

c. All the processors send their local cluster data to rest of the processors.

d. All the processors receive the data sent by other processors and recompute the centroids locally.

e. Each processor checks for convergence condition. If convergence condition is not reached, then steps 2 & 3 are repeated. This process is repeated till convergence condition is reached.

Figure 2 compares the two MatlabMPI implementation of K-Means clustering with the Serial implementation. From Fig. 2 it is observed that the difference in the time taken by serial process and that taken by the two MatlabMPI implementations increases as the number of centroids to be clustered or the number of data points to be clustered increases. Moreover, both the parallel implementations take nearly the same amount of time.



This publication was made possible through support provided by DoD HPCMP PET activities through Mississippi State University under the terms of Agreement No. #GS04T01BFC0060**.** The opinions expressed herein are those of the author(s) and do not necessarily reflect the views of the DoD or Mississippi State University.

References

Vipin Kumar Mahesh V. Joshi, George Karypis [2002]. *Shared memory parallelization of data mining algorithms: Techniques, programming interface, and performanc.* In Second SIAM conference on Data Mining, 2002.

Jeremy Kepner [2002]. MatlabMPI Improves Matlab Performance By 300x. In MAUI HIGH PERFORMANCE COMPUTING CENTER Appication Briefs, 2002.

Abstract *Presentation* *Back to Agenda* *Next Abstract*

A Parallel Data Mining Package Using MatlabMPI

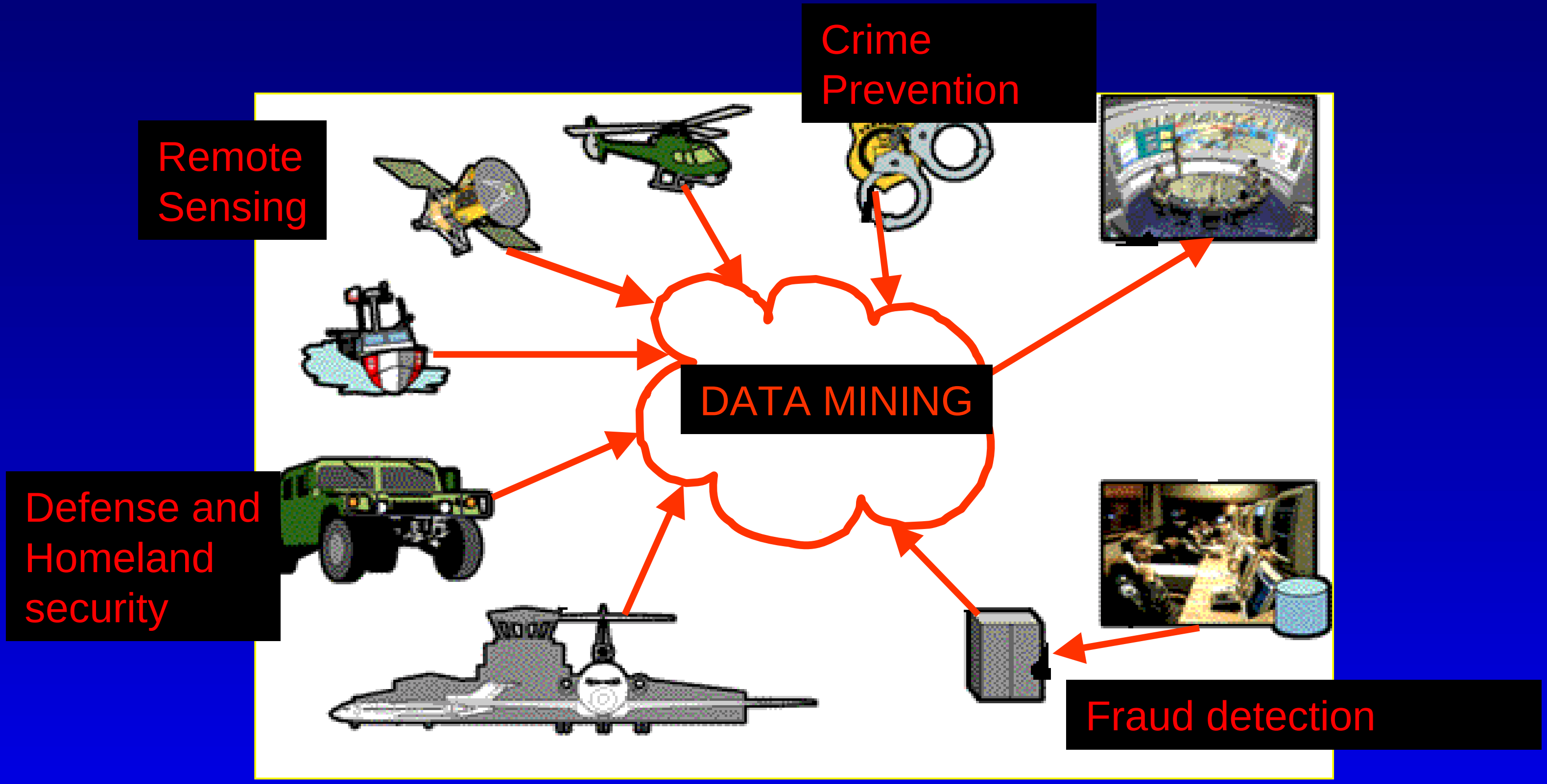
Parna Khot
Ashok Krishnamurthy
Stan Ahalt
John Nehrbass
Juan Carlos Chaves

The Ohio State University

Outline

- Motivation
  - Why parallel data mining toolbox?
- MatlabMPI
  - What is MatlabMPI?
- Parallel data mining toolbox
  - K-Means Clustering
  - CART
- Results of MatlabMPI implementation
- Conclusions
- Future Work

Motivation



Remote Sensing

Crime Prevention

DATA MINING

Defense and Homeland security

Fraud detection

- Today, the amount of data that is collected from sensors and computerized transactions is huge.
- Data Mining algorithms arise in many different fields and typically are used to search through this data to look for patterns.
- Parallel data mining algorithms can help handle the huge datasets in a timely manner.

Typical Data Mining Tasks

- Clustering.
- Classification.
- Association Rules.
- Regression.
- Pattern Recognition

We will consider only Clustering and Classification in this presentation.

MatlabMPI Overview

The latest MatlabMPI information, downloads, documentation, and information may be obtained from*:*

http://www.ll.mit.edu/MatlabMPI

Parallelization using MPI

- The Message Passing Interface (MPI) is a general method of parallelization by including explicit calls within the code to a library for exchanging messages between the processing elements.
  - MPICH
    - Implementation of Message Passing Interface standard for C, C++, Fortran77, Fortran90.
  - MatlabMPI
    - A Matlab implementation of MPI.

MPI & MATLAB

- **Message Passing Interface (MPI):**
  - A message-passing library specification.
  - Specific libraries available for almost every kind of HPC platform: shared memory SMPs, clusters, NOWs, Linux, Windows.
  - Fortran, C, C++ bindings.
  - Widely accepted standard for parallel computing.

- **MATLAB:**
  - Integrated computation, visualization, programming, and programming environment.
  - Easy matrix based notation, many toolboxes, etc
  - Used extensively for technical and scientific computing.
  - Currently: mostly SERIAL code.

What is MatlabMPI?

- It is a MATLAB implementation of the MPI standards that allows any MATLAB program to exploit multiple processors.
- It implements, the basic MPI functions that are the core of the MPI point-to-point communications with extensions to other MPI functions. (Growing)
- MATLAB *look and feel* on top of standard MATLAB file I/O.
- Pure M-file implementation: about 100 lines of MATLAB code.
- It runs anywhere MATLAB runs.
- Principal developer: Dr. Jeremy Kepner (MIT Lincoln Laboratory)

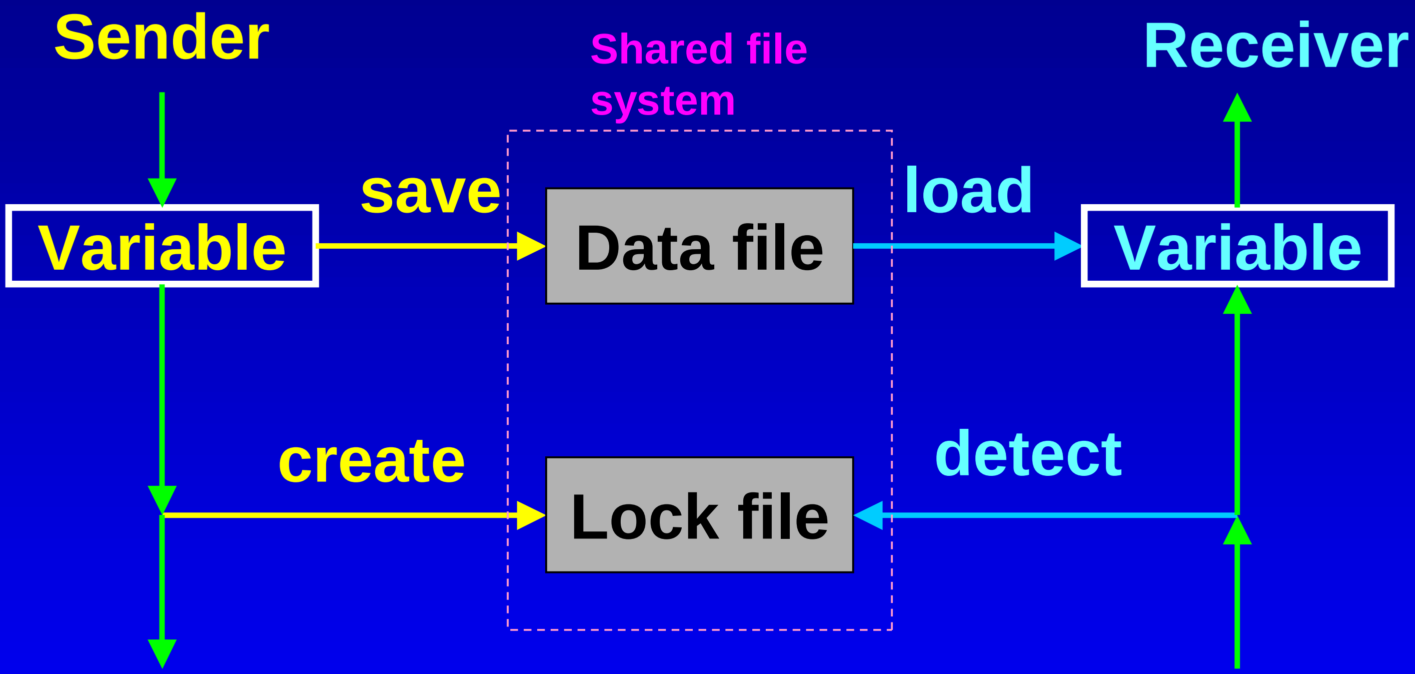
General Requirements

- As MatlabMPI uses file I/O for communication, a common file system must be visible to every machine/processor.
- On shared memory platforms: single MATLAB license is enough since any user is allowed to launch many MATLAB sessions.
- On distributed memory platforms: one MATLAB license per machine / node.
- Currently Unix based platforms only, but Windows support coming soon.

Basic Concepts

- **Basic Communication:**
  - **Messages: MATLAB variables transferred from one processor to another**
  - **One processor sends the data, another receives the data**
  - **Synchronous transfer: call does not return until the message is sent or received**
  - **SPMD model: usually MatlabMPI programs are parallel SPMD programs. The same program is running on different processors/data.**

Communication architecture



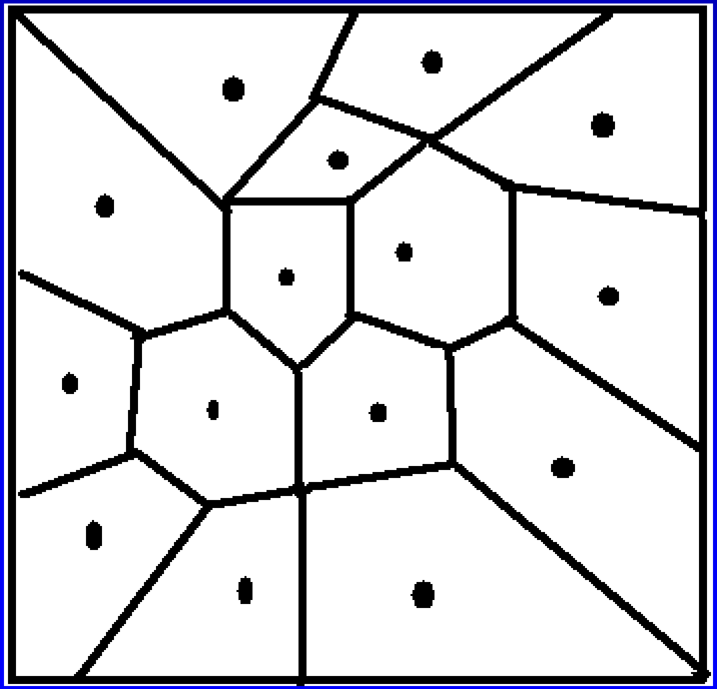
- Receiver waits until it detects the existence of the lock file.
- Receiver deletes the data and lock file, after it loads the variable from the data file.

Possible modifications/customizations

- ssh vs rsh.
- Path variables.
- System dependent information required to run MATLAB.

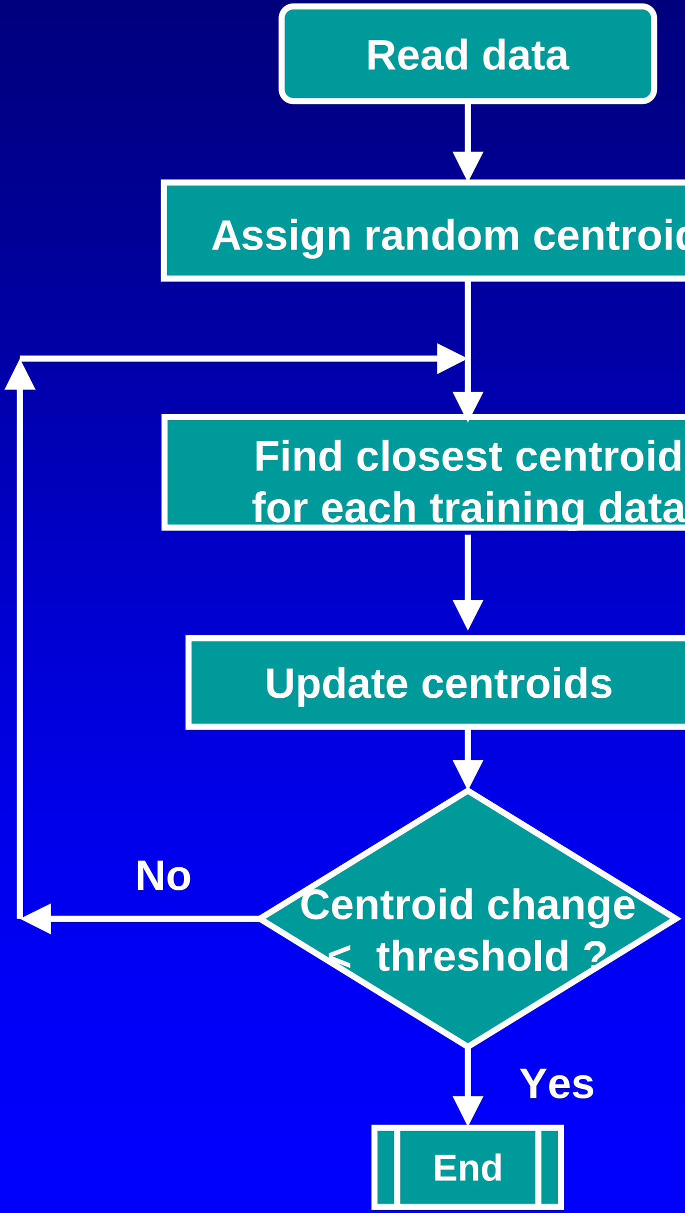
Data Mining Toolbox: Clustering

- Clustering divides the data into disjoint subsets based on a similarity measure.
- Each subset (cluster) is characterized by its centroid.
  - Training data is used to estimate the centroids.
- K-Means is a commonly used clustering algorithm.
  - The number of clusters is assumed to be known apriori.



Voronoi Diagram

K-Means Clustering



Read data

Assign random centroids

Find closest centroid for each training data

Update centroids

Centroid change < threshold ?

No

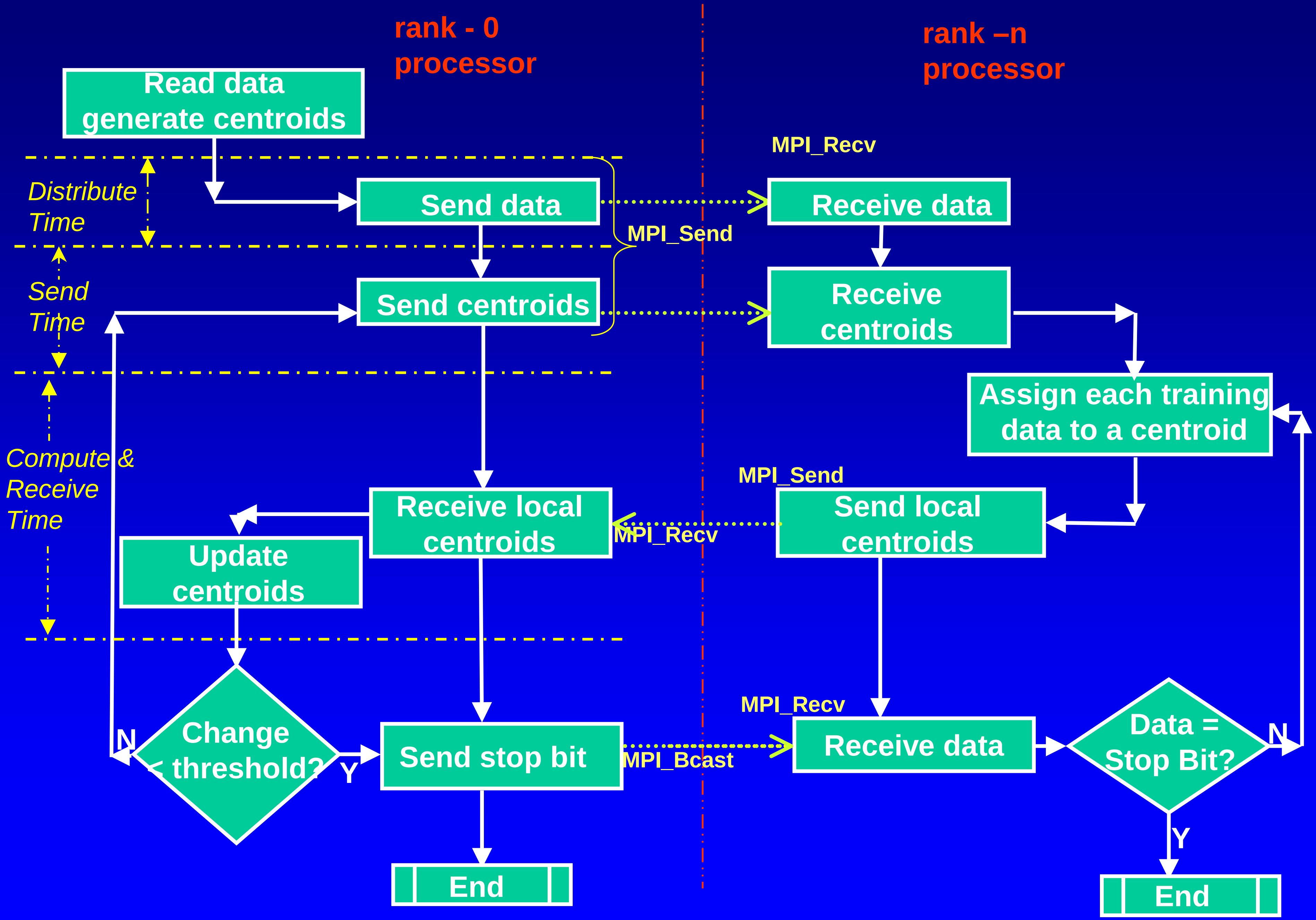
Yes

End

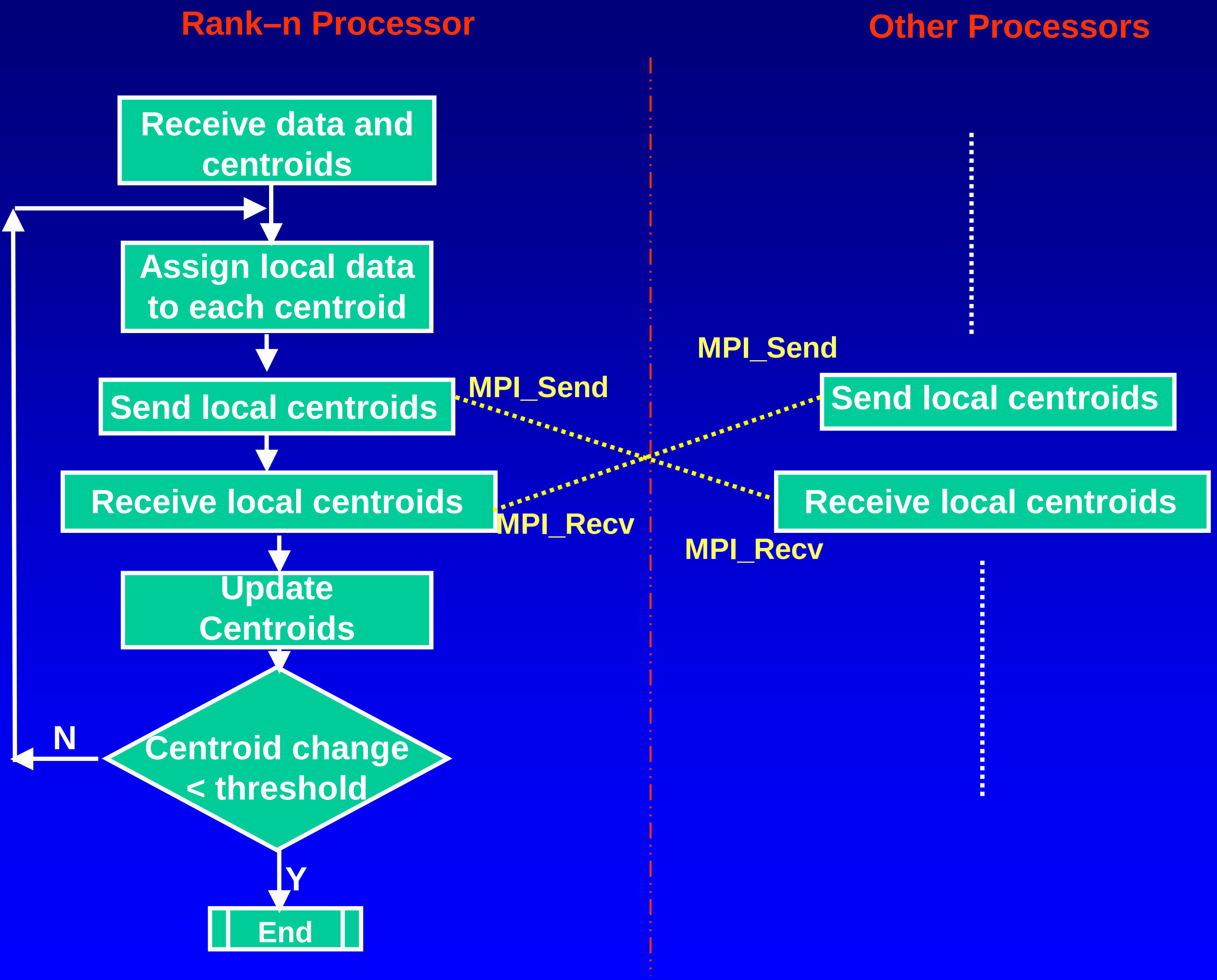
Parallel K-Means Clustering

- We have considered two approaches:
  - **Master- Slave Method**– The rank-0 processor determines when clustering is done.
  - **Peer-to-Peer Method** – All the processing elements communicate among themselves to decide when clustering is done.

Master – Slave Method



Peer-to-Peer Method



Rank–n Processor

Other Processors

Receive data and centroids

Assign local data to each centroid

Send local centroids

MPI_Send

Receive local centroids

MPI_Recv

Update Centroids

Centroid change < threshold

N

Y

End

MPI_Send

Send local centroids

Receive local centroids

MPI_Recv

Communication And Compute Times

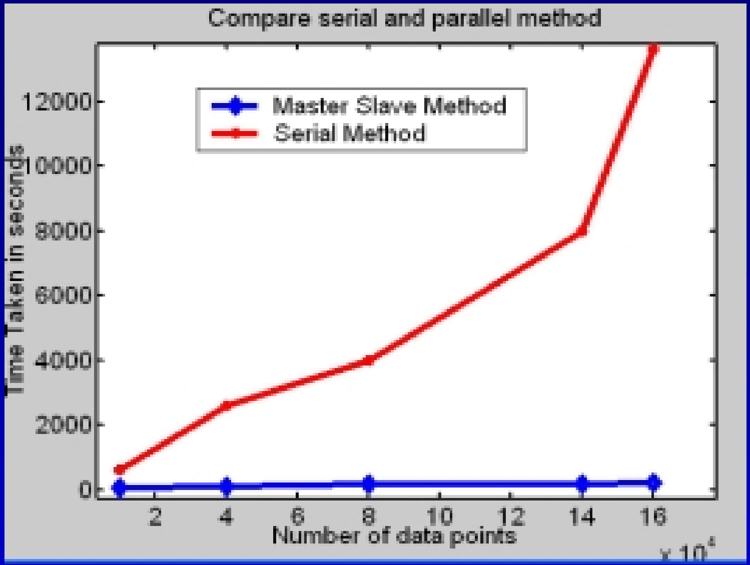
- Consider clustering of N vectors of dimension D into K clusters. Assume that clustering takes L iterations through the data, and P processors are used.
- Serial Method
  - Communication Time – N/A
  - Communication Data Size – N/A
  - Compute Time – O(NKL)
- Master Slave Method
  - Communication Time – (N-1)*(P+1) T_{MPI_Send} + (N-1)*P T_{MPI_Recv}
  - Communication Data size
    - Initial – (N+K)/(P-1)
    - Per loop = K
  - Compute Time / Processor – O((N/(P-1))K)
- Peer-to-Peer Method
  - Communication Time – (N)* (P) (T_{MPI_Send} + T_{MPI_Recv}).
  - Communication Data size
    - Initial – (N+K)/(P-1)
    - Per loop = K
  - Compute Time / Processor – O((N/P)K)

Parallelization Effectiveness

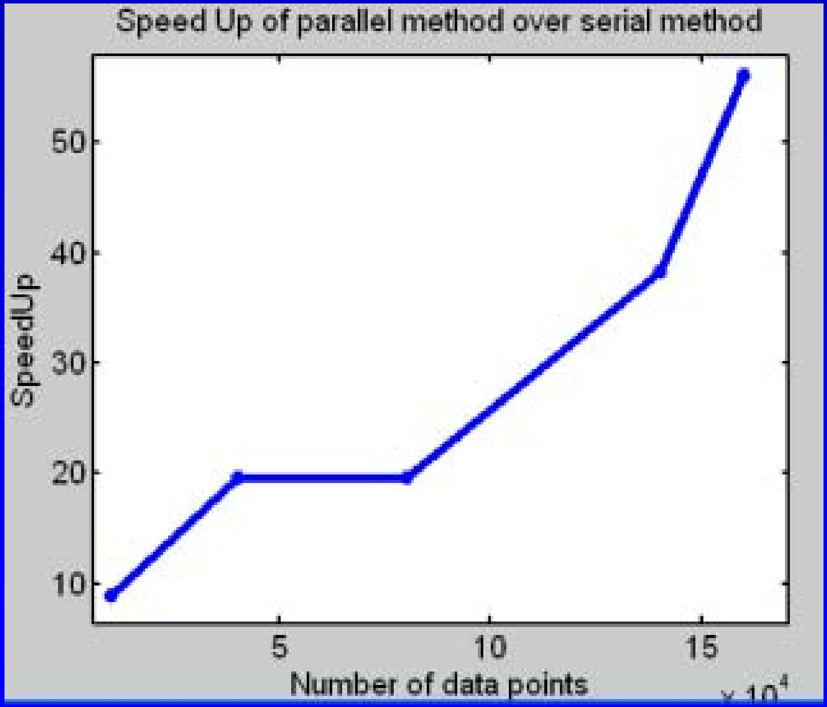
- We studied the effects of following parameter variations on the Master-Slave parallel K-means algorithm
  - Number of data points.
    - To observe the effect of increase in total data size.
  - Number of centroids.
  - Scalability.
    - To observe the effect of change in number of processing elements.

Effect of varying number of data points

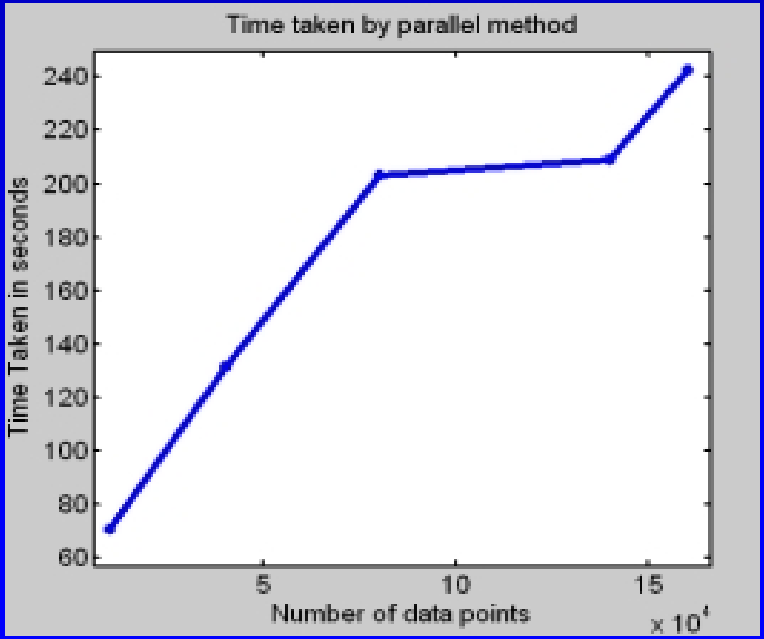
- Data Set
  - Number of data points: 1M – 16M
  - Number of centroids: 30
  - Number of processors: 16
  - Dimensionality of data: 3
- As number of data points is increased speed up of parallel process over serial process increases.



Compare serial and parallel method



Speed Up of parallel method over serial method



Time taken by parallel method

Tested on SUN E10000 - 64 Ultrasparc II

Effect of varying number of centroids

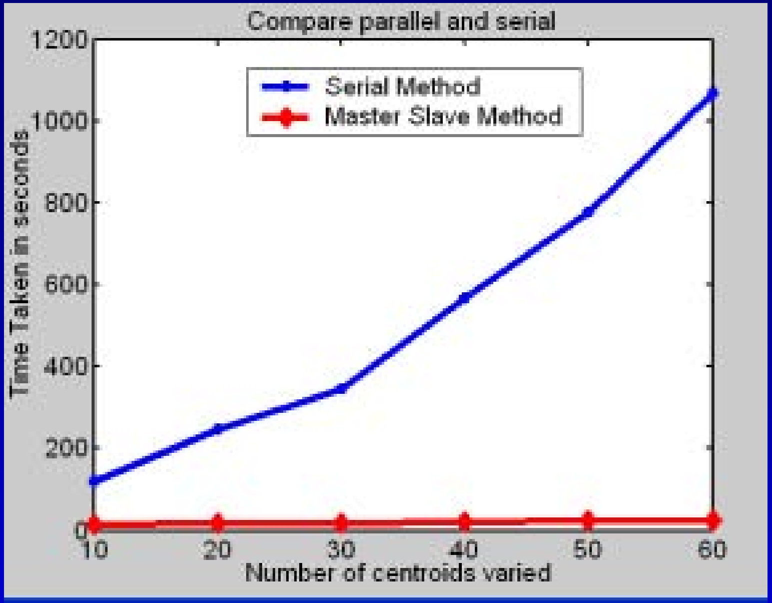
- Data Set

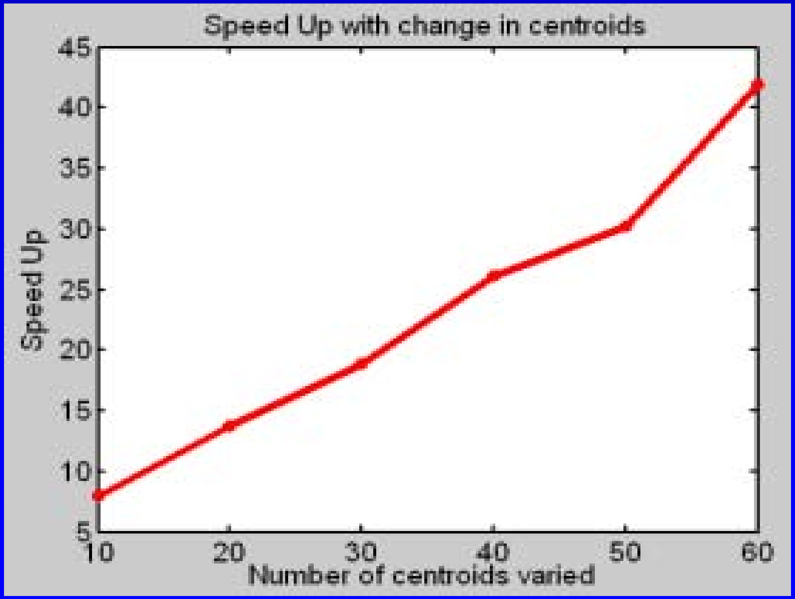
  Number of data points – 0.4M
  Number of centroids – varied
  Number of processors – 16
  Dimensionality - 8

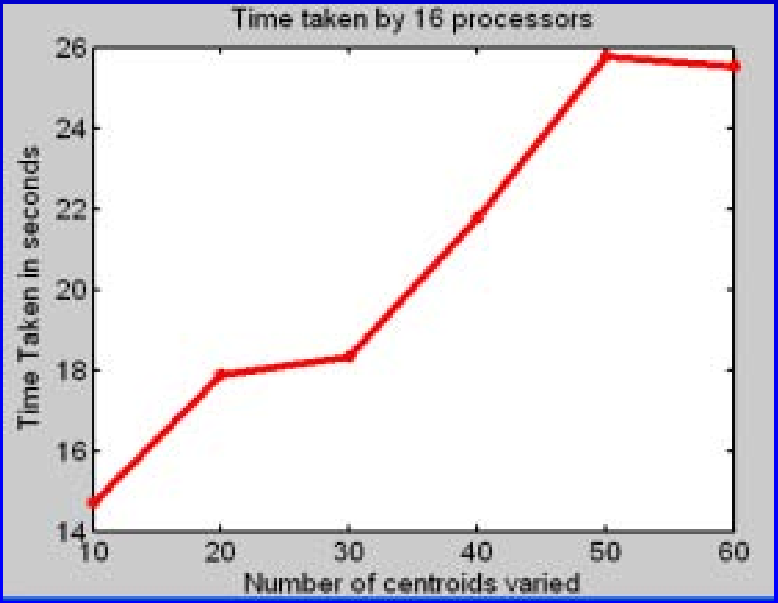
- Effect of increase in number of centroids with constant number of data points

  •The number of data points per process is constant.

  • Speed up observed since compute time is of the order of NK.



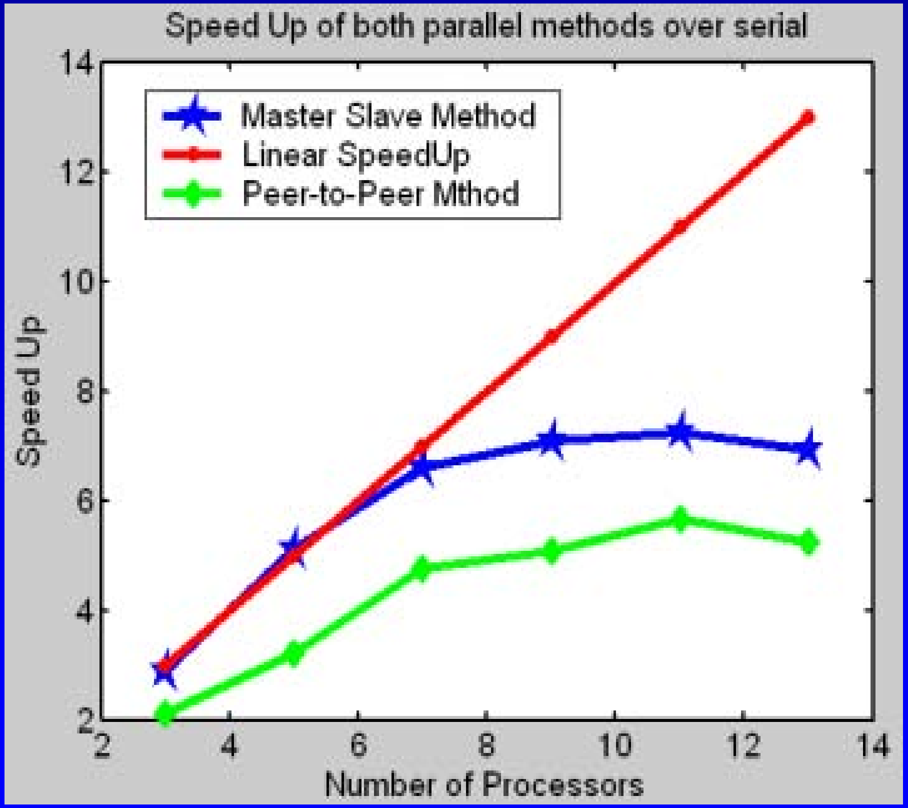


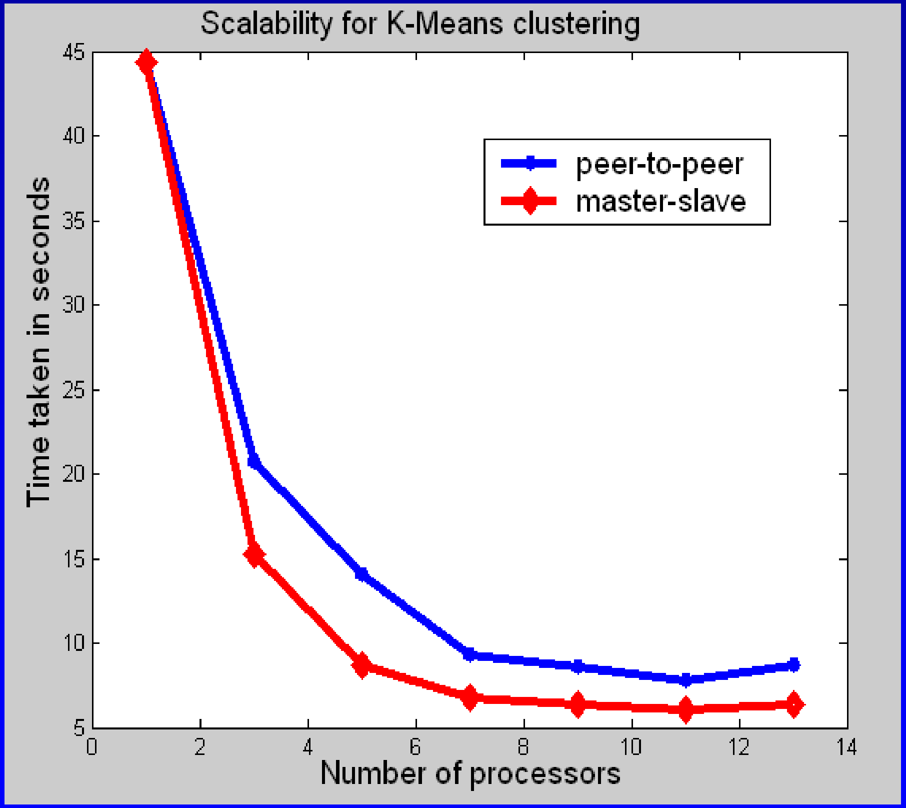


OSC IA32 Cluster distributed/shared memory, 64 compute nodes with two 1.533 GHz AMD Athlon MP processors

Scalability Results

- As number of processors is increased the time taken decreases
  - number of data points: 0.2M
  - number of clusters: 30
  - Dimensionality: 3





Tested on distributed/shared memory hybrid system Dual processor - 1.53GHz AMD Athlon 1800MP CPUs at OSC

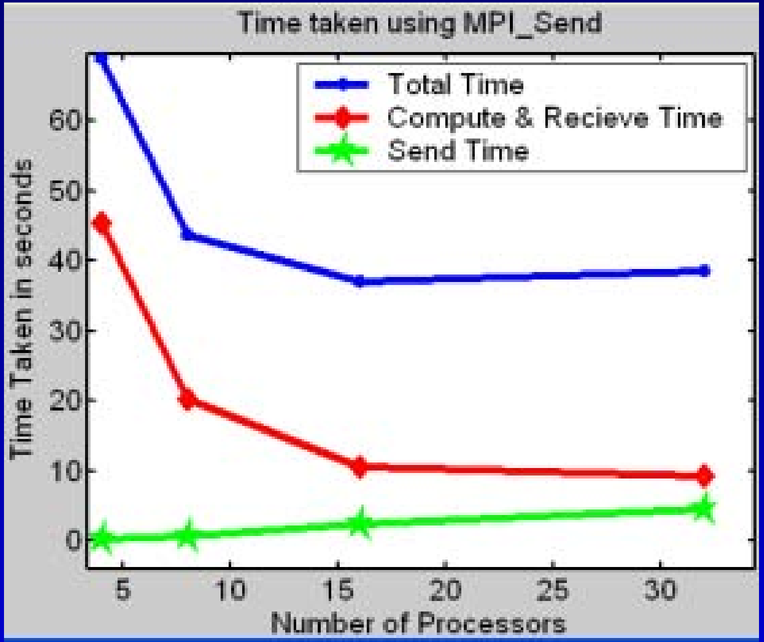
Dependence on data size

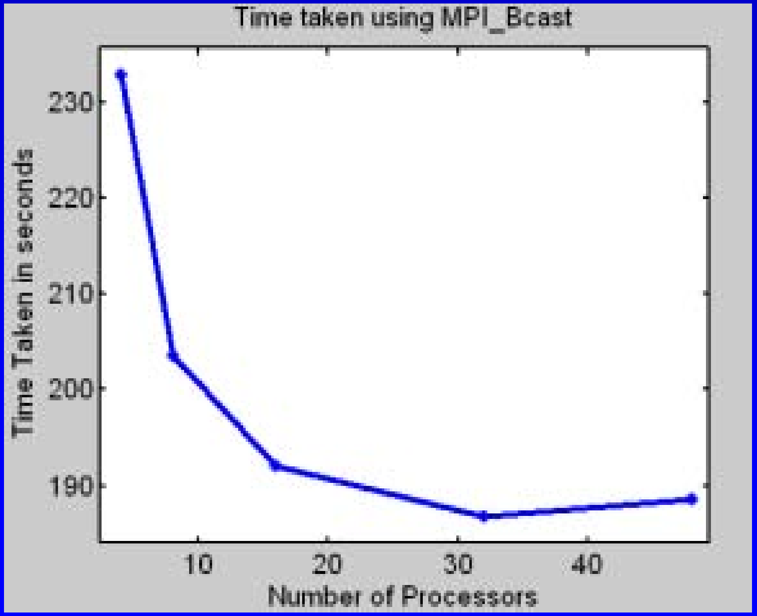
- The decrease in time as the number of processors is increased is not true for all cases

- Data Set for figure :
    - Number of data points: 1M
    - Number of clusters: 16
    - Dimensionality: 8

- For 32 processors increase in time taken to send data is greater than the decrease in computation and receive time.
    - Rank-0 needs to write 31 files to send data to other processors.

- Using MPI_Bcast instead of MPI_Send shows scalability for 32 processors also, but overall time taken is more.

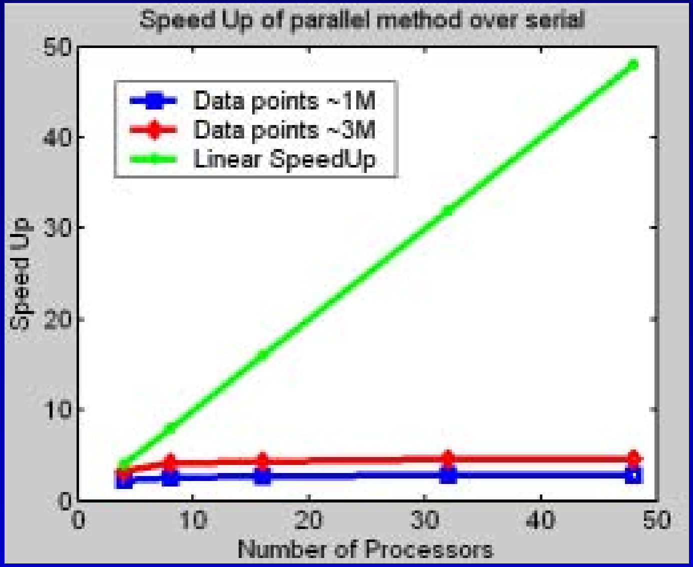


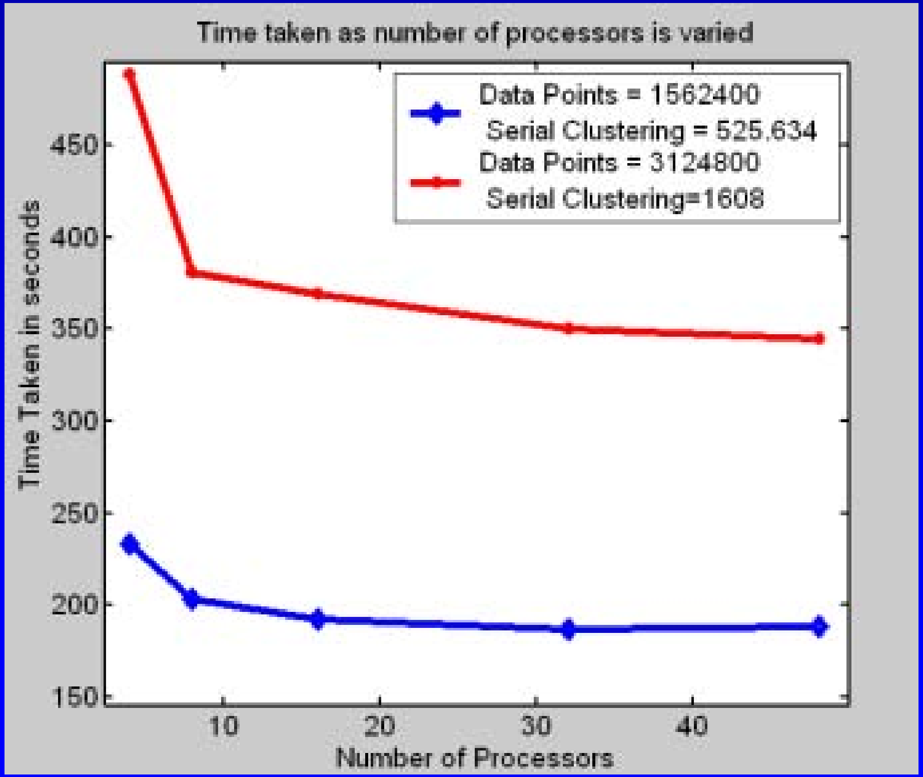


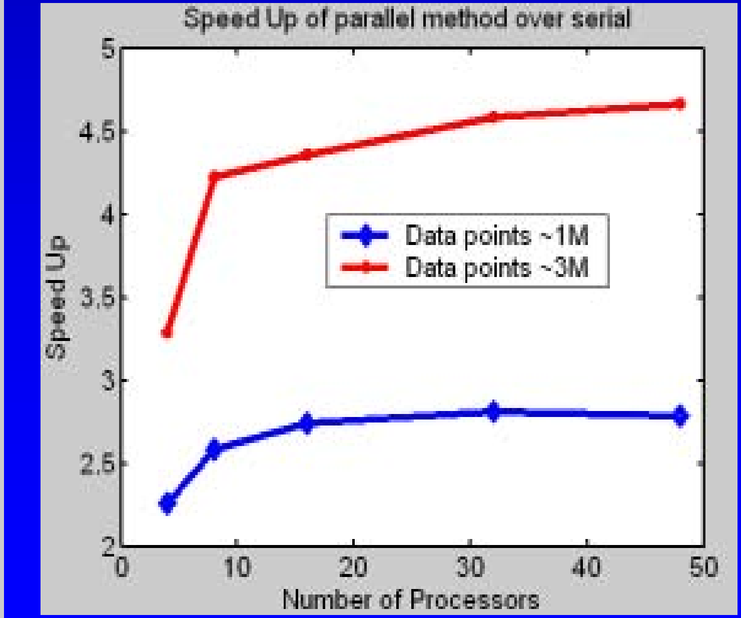
OSC IA32 Cluster distributed/shared memory, 64 compute nodes with two 1.533 GHz AMD Athlon MP processors

Effect of MPI_Bcast

- Time taken for parallel process decreases as number of processors is increased.
- For 3M the time taken decreases as number of processors is increased.
- Observe for ~1M
  - time taken by 48 processors > time taken by 32 processors



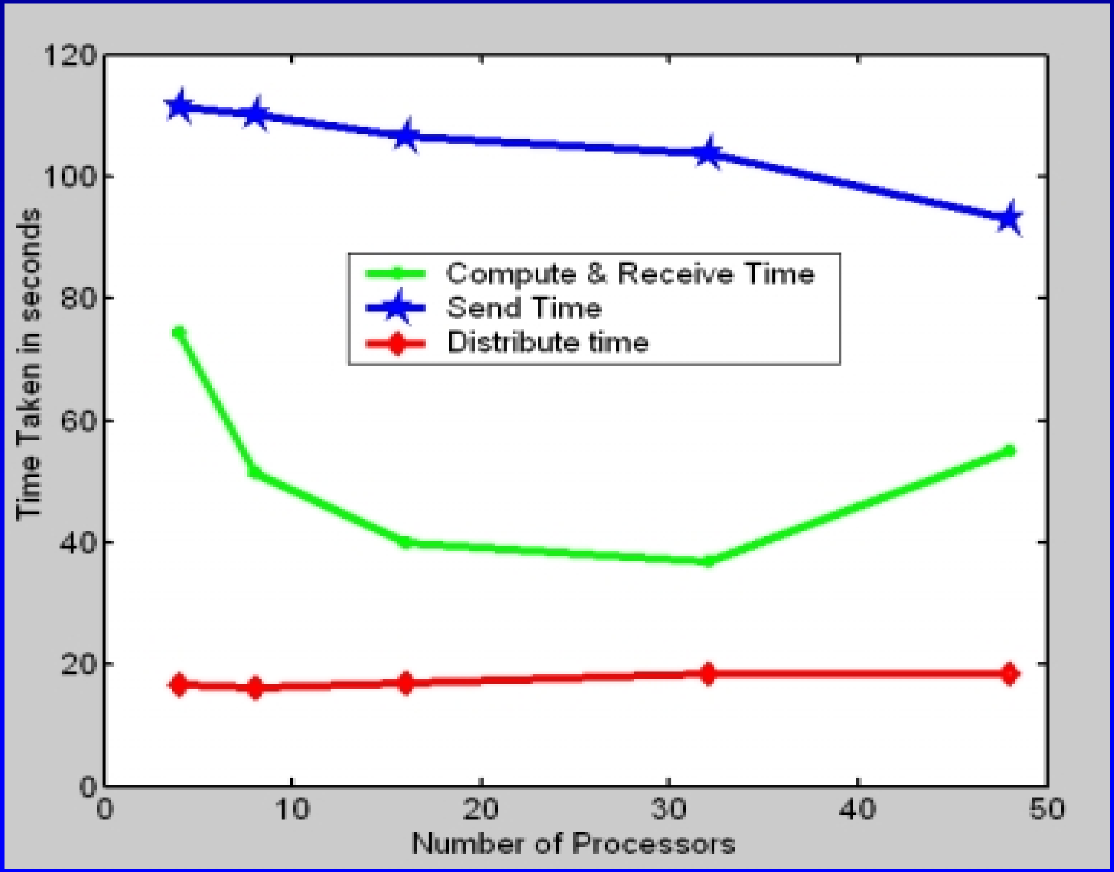




Tested on distributed/shared memory hybrid system Dual processor - 1.53GHz AMD Athlon 1800MP CPUs at OSC

Why this behavior with MPI_Bcast?

- Time taken to read data from 47 processors is reduced
- Time taken to distribute the data is modestly increased.
- But Rank-0 processor receives data from 47 processors and this time increases significantly



Tested on distributed/shared memory hybrid system Dual processor - 1.53GHz AMD Athlon 1800MP CPUs at OSC

Conclusion

- For K-Means Clustering
  - Speedup is observed as number of data points is increased.
  - Speedup is observed as number of centroids is increased
  - For given data size as the number of processors is increased time taken decreases only to the point that the increase in communication cost overshadows the decrease in computation cost
- The advantage of using MatlabMPI is observed if data size is large.

Data Mining Toolbox: Classification

Classification and Regression Tree (CART)

- Classification Tree
  - A tree structured classifier obtained by systematic splitting of training data samples using attribute values.
- Regression Tree
  - A tree structured model to predict values (get function description) of a continuous valued variable based on values of other variables.

Classification Tree

- A tree structured classifier is built in two phases:

  1) Growth Phase : In this the tree is built by recursively partitioning the data until a threshold condition is reached.

  2) Prune Phase : If the tree obtained in the growth phase is too large or too small then the misclassification rate will be high as compared to the right sized tree. The pruning of the tree is done to obtain a right sized tree.

- Only the Growth Phase of CART has been parallelized.

Example

- We explain the steps to build a classification tree using a smaller example.
- Training data
  - Classes – 3
  - Attributes – 3

Attr1	Attr2	Attr3	Class
0	0	0	1
0	1	0	2
1	1	0	3

- Size of training data (Elements per class)
  - Class 1 = 3
  - Class 2 = 5
  - Class 3 = 7

Sequential Classification tree

- Steps:

1. The selection of the splits.
2. The decisions when to declare a node terminal or to continue splitting it.
3. The assignment of each terminal node to a class.

Selection of Splits

- Split Question (X-attribute, C-integer value)
  - continuous attributes : {Is X<C?}
  - categorical attributes : {Is X=C?}
    - In above example Q- {Is X=0?)
- Split Criterion:

  Best split minimizes impurity at a node
  - eg: Gini index is given by:

$$i(t) = 1 - \sum_{j} p_j^2$$

where p_j is the proportion of class ‘j’ at node ‘t’.

- At a node with ‘n’ elements if split ‘S’ divides the data into S_1 (n_1 elements) and S_2 (n_2 elements)

$$gini_{split}(S) = \frac{n_1(gini(S_1)) + n_2(gini(S_2))}{n_1 + n_2}$$

- The split that maximizes $\Delta i(s,t)$ is selected to be the best split.

Splitting the main node

- Gini Index at root node
  - Count matrix for each attribute
  - If attribute value – 0 then data goes to left node
  - Attribute –1

Value	C-1	C -2	C-3
0	3	5	0
1	0	0	7

Gini Index: n1=8,n2=7
gini(s1)=0.46857
Gini(s2)=0
Ginisplit=0.25

  - Attribute - 2

Value	C-1	C -2	C-3
0	0	5	7
1	3	0	0

Gini Index: n1=12,n2=3
gini(s1)=0
gini(s2)=0.486
Ginisplit=0.388

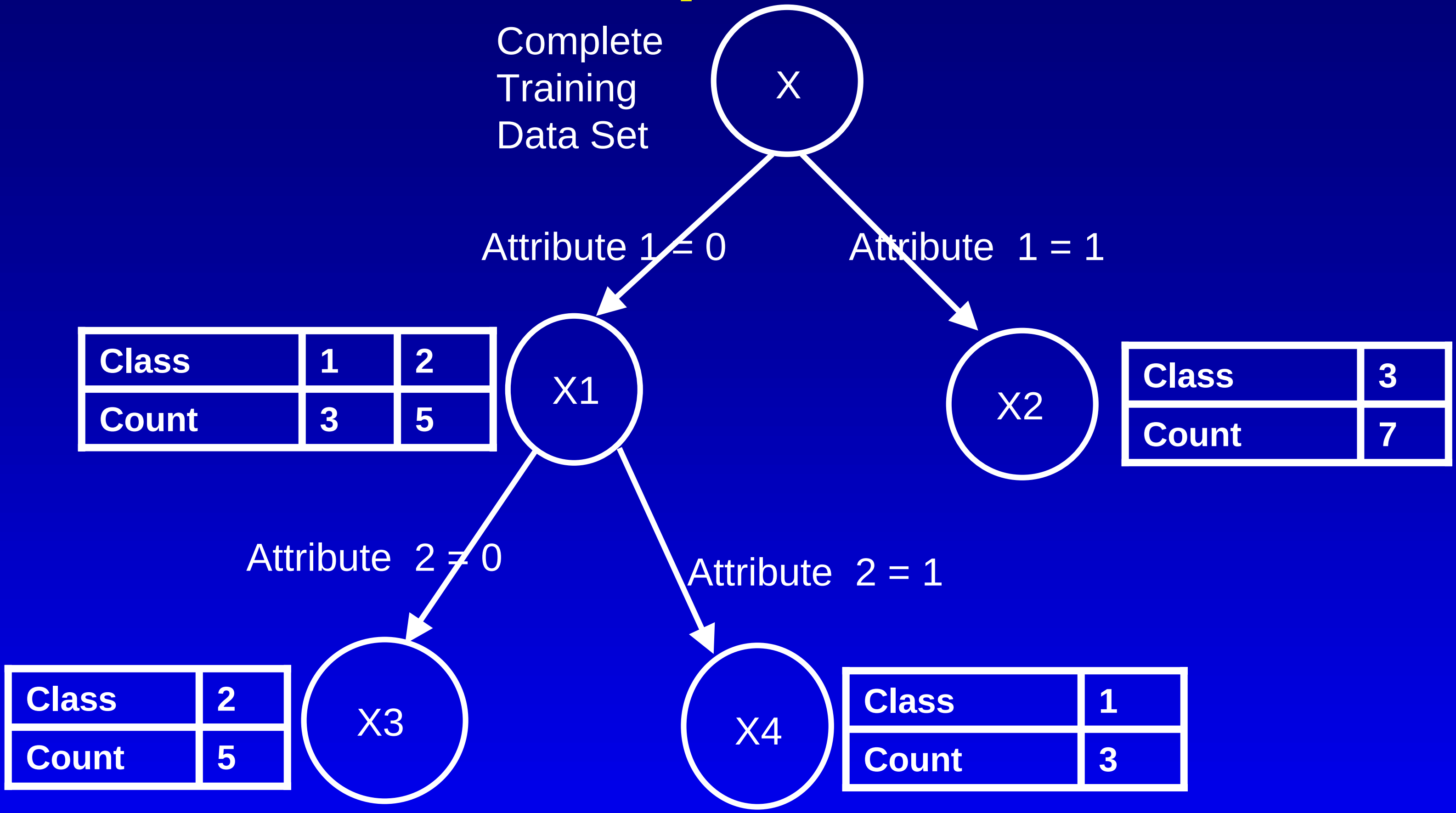
  - Attribute – 3

Value	C-1	C -2	C-3
0	3	5	7
1	0	0	0

No use splitting with this attribute since n2=0

- The best splitting attribute is 1 since it has minimum gini index.

Split Tree



Class	1	2
Count	3	5

Class	3
Count	7

Class	2
Count	5

Class	1
Count	3

Serial Growth Phase - contd.

- Decision to stop splitting
  - A node is decided to be a terminal node if the Gini index is lower than a threshold.
  - Splitting is stopped at node ‘t’ if

    $$\max_{s \in S} \Delta i(s,t) < \beta$$

    Or if the node is pure (as in above example.)

- Assign class to each terminal node.
  - Class *j* is assigned to terminal node $t \in \tilde{T}$ if

    $$j = \arg\max_{i} p(i \mid t)$$

Parallel CART (For Categorical Attributes)

1. Suppose the size of the given data set is N and number of processors is P.
2. The rank-0 processor
   - **Reads the training data**
   - **Distributes the data equally among all the processors.**
3. All other processors
   - **Calculate and send the count matrices for all attributes.**
4. Rank-0 processor
   - **Receives count matrices**
   - **Finds best splitting attribute**

Parallel CART – contd.

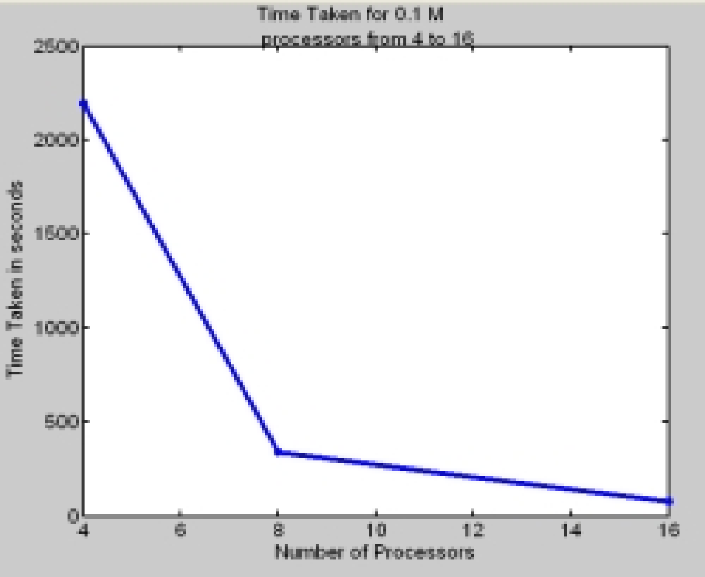
5. Rank-0 process
   - Stops if all terminal nodes are pure.
   - Else sends best splitting attribute to all other processors.
6. All other processors
   - Split the data into the left and right node using the best splitting attribute.
   - Steps 3-6 are repeated for each of the leaves.

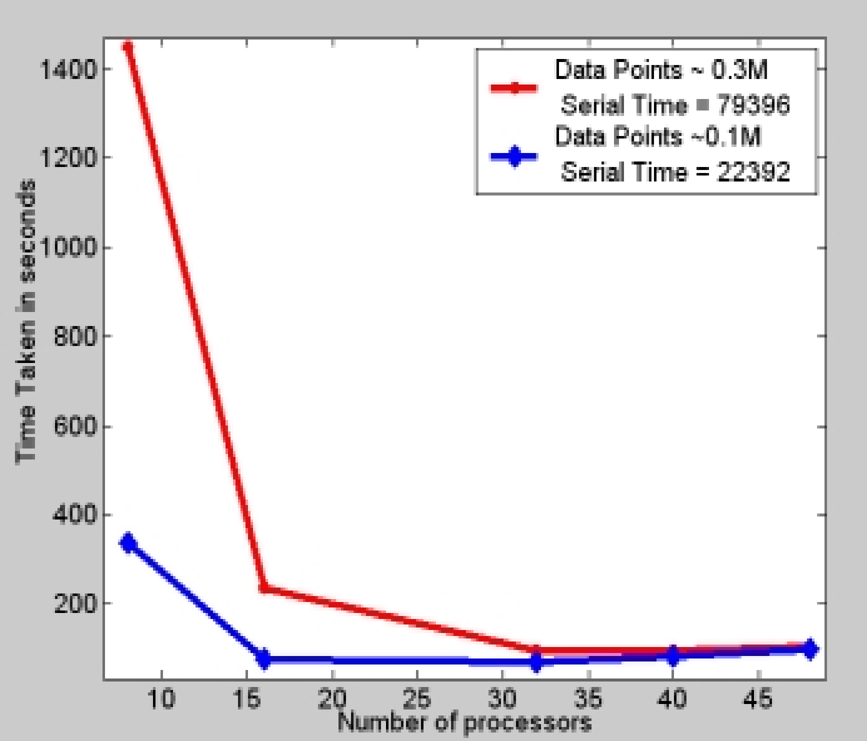
Effects Of Parallelization of categorical CART

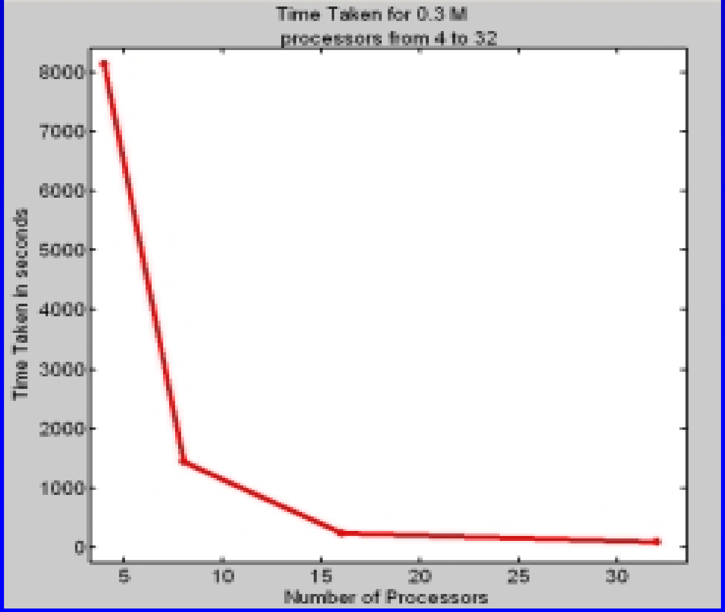
- We studied the performance of the parallel algorithm with the variation in number of processing elements.
  - As the number of processors is increased the number of training samples per processor decreases.
  - Time taken per processor decreases hence total time taken decreases.

Scalability Results

- Time taken to get classification tree using 0.3M and 0.1M training data points. Number of attributes: 7
Number of classes: 10
- Serial process takes very long.
  For 0.3M data points with 32 processors, speedup is about 845
- But for number of processors greater than 32 time taken increases



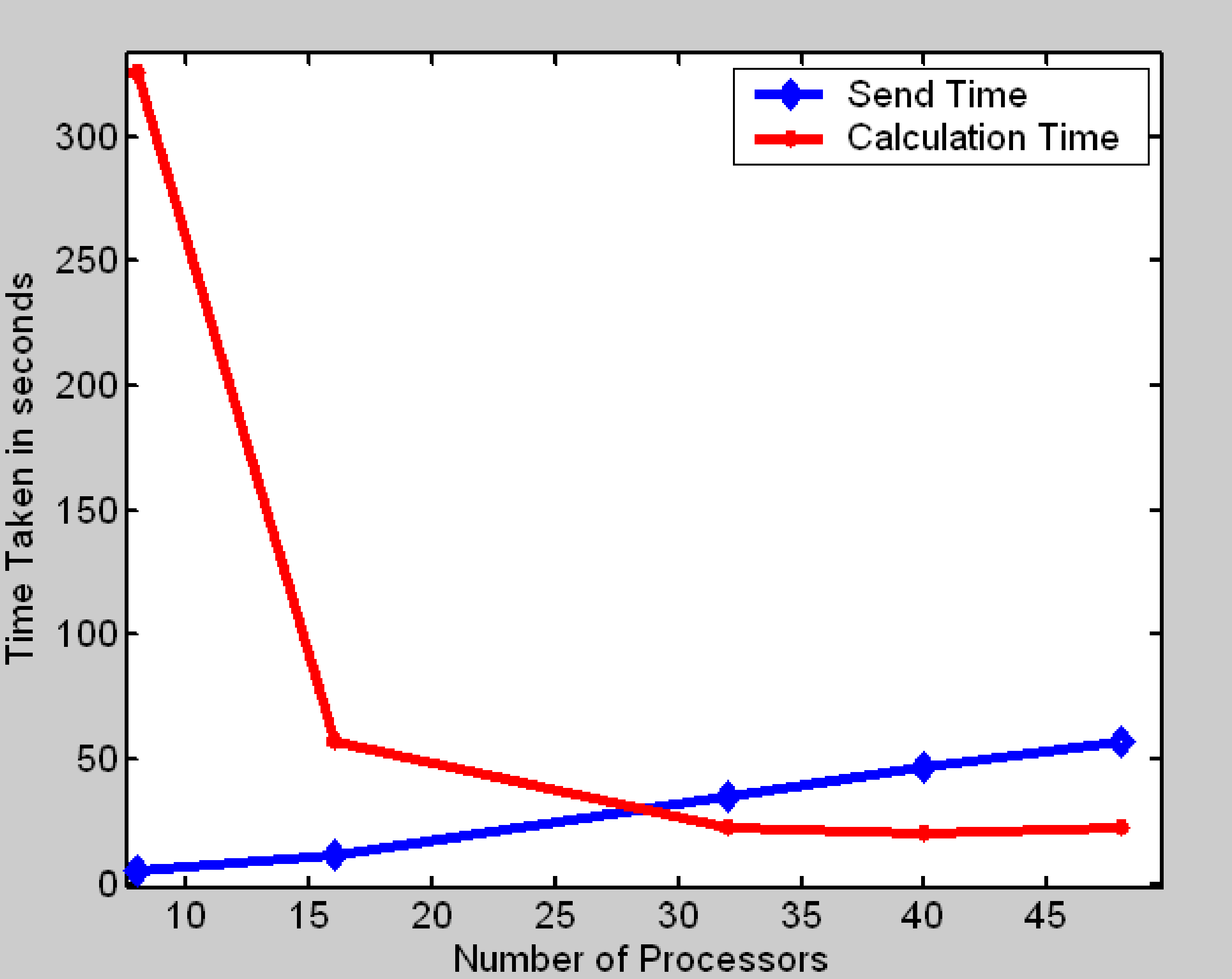




Tested on distributed/shared memory hybrid system Dual processor - 1.53GHz AMD Athlon 1800MP CPUs at OSC

Reason For Increase In Time

Increase in time taken to send messages is greater than decrease in computation time



Tested on distributed/shared memory hybrid system Dual processor - 1.53GHz AMD Athlon 1800MP CPUs at OSC

Conclusions

- Parallel processing takes less time than serial process
- For large data sizes the increase in communication cost is less than the decrease in calculation cost.
- Parallel CART using MatlabMPI can be used with very large data sets

Future Work

- Optimize the use of MPI_Bcast.
- Generalize CART algorithm for continuous type of attributes.
- Parallelize Prune Phase.
- Add Support Vector Machines to the parallel data mining toolbox.