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.

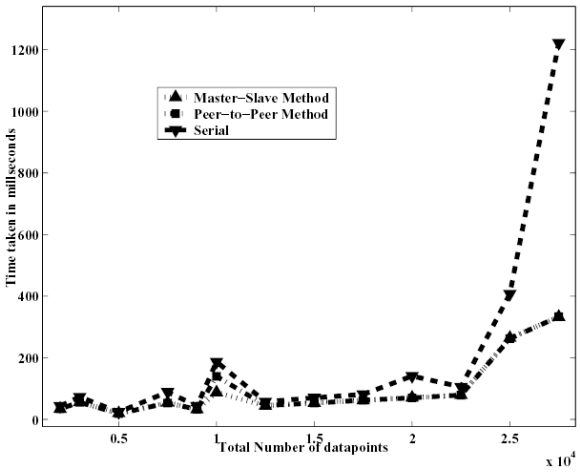
We present two parallel implementations of K-Means clustering using MatlabMPI in this poster.

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.

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.