A MPI-MapReduce Based Parallel Engine For Clustering Complex Biological Ligand

[1]Vishal Prasad Sharma, [2]Udit Kamboj
[1]Dept. of Computer Science and Engineering (I.S),
[1] NIT Sikkim, Sikkim, India

Abstract *:* Driven by the consistent expanded in "conceived advanced" information, information driven registering gives new difficulties in information administration and investigation in the life sciences and in numerous different controls extending from coordination's (activity examination) to sociology(online networking). These difficulties are altogether different than the customary re-enactment challenges since managing a large number of individual records and terabytes of information. Cutting edge investigation techniques in light of bunching and orders for conventional re-enactments regularly require the examination of complex information records in an iterative procedure. In this paper we present MPI-MapReduce based clustering of complex biological dataset using octree clustering method. We performed experiments on a distributed environment using MapReduce and MPI. The results outperforms the existing clustering method which uses Euclidian distance. The proposed L2 norm based octree clustering shows more levels of separated ligand.

Keywords: Biological Ligand, MPI, MapReduce, Octree Based Clustering.

I. INTRODUCTION

As of late, novel patterns towards parallel architectures have offered better reaction time and execution for parallel computation applications. Multi-core groups demonstrate discernible changes in equipment architectures amid HPC ages, for example, symmetric multi-processors (SMP shared memory frameworks) or single core hubs. Multi-core groups influence this attainable to blend two highlights of parallel programming dialects; to message going in shared memory and multi-cores utilizing shared memory.

Executing parallel calculation brought numerous ideas which need to consider just, memory progressive system [1]. It will influence execution by two noteworthy parameters: memory inertness; that is the time passed from point a bit of information is required until the point that information wind up plainly accessible, memory transfer speed; that is the speed which information are sent from memory to processors.

This exploration concentrated on how multi-core bunches architectures can influence the execution of message passing and shared memory while running parallel computation application, and the approaches to enhance handling time and execution by mix of two basic parallel programming dialects: unadulterated MPI and OpenMP.

This exploration concentrated on how multi-center group's architectures can influence the execution of message passing and shared memory while running parallel computation application, and the approaches to enhance handling time and execution by blend of two basic parallel programming dialects: unadulterated MPI and OpenMP.

MESSAGE PASSING INTERFACE

MPI furnishes parallel equipment sellers with a plainly characterized base arrangement of routines that can be effectively executed. Thus, equipment merchants can expand upon this gathering of standard low-level routines to make more elevated amount routines for the distributed memory correspondence condition provided with their parallel machines. MPI gives an easy to-utilize convenient interface for the essential client, yet one sufficiently intense to enable software engineers to utilize the superior message passing operations accessible on cutting edge machines.

A. Functionality

The MPI interface is intended to give fundamental virtual topology, synchronization, and correspondence usefulness between an arrangement of procedures (that have been mapped to hubs/servers/PC occurrences) in a dialect autonomous manner, with dialect particular linguistic structure (ties), in addition to a couple of dialect particular highlights.

B. Concepts

MPI gives a rich scope of capacities. Some of them are presented below.

1. Communicator: Communicator objects interface gatherings of procedures in the MPI session. Every communicator gives each contained procedure a free identifier and orchestrates its contained procedures in a requested topology.

2. Point-to-point rudiments: Various critical MPI capacities include correspondence between two particular procedures. A prevalent case is MPI_Send, which enables one indicated procedure to make an impression on a moment determined process.

3. Aggregate fundamentals: Aggregate capacities include correspondence among all procedures in a procedure gathering (which can mean the whole procedure pool or a program-characterized subset). A run of the mill work is the MPI_Bcast call (another way to say "communicate").

HADOOP – MAPREDUCE

The Apache Hadoop is a framework that considers the distributed handling of extensive informational collections crosswise over groups of PCs utilizing basic programming models. It is intended to scale up from single servers to a huge number of machines, each offering neighborhood calculation and capacity [4]. The basic MapReduce operation are presented in fig. 1.

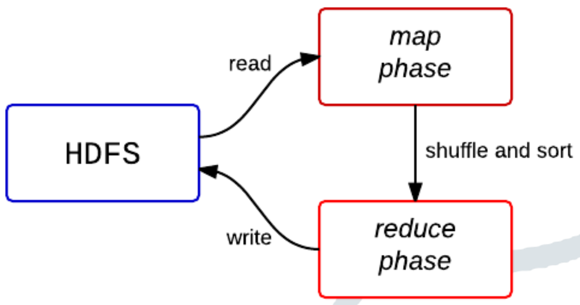

Fig. 1. Shows basic operation of MapReduce Model

Hadoop empowers the application to work in a distributed environment. There might be a great many distributed part cooperating to finish a solitary errand. For the most part, the colossal log records are distributed over different bunches known as HDFS group [12] (Hadoop distributed document framework). Hadoop separates the records into the quantity of pieces. These pieces are distributed over different bunches and handled in every framework in a parallel design. The execution of the Hadoop framework is picked up by working documents in a parallel environment.

II. RELATED WORK

The MapReduce paradigm has been used in the past to analyse large data. For example, the paradigm was adapted to analyse long MD trajectories in parallel with the tool HiMach [6]. Work in [6] includes the design and use of a MapReduce-style programming interface to exclusively analyse statistical data of long trajectories. Our approach looks at more general similarities in large datasets. Other efforts have investigated well-known clustering methods, such as k-means clustering and hierarchical clustering, which were adapted to fit into the MapReduce framework [7], [8]. However, the resulting implementations suffer from the limitations of the clustering algorithms, which do not scale, despite being formulated in the MapReduce paradigm. A similar clustering approach based on the density of single points in an N-dimensional space was presented in the work of Cordeiro el al. [9]. The three algorithms presented in Cordeiro’s work rely on local clustering of sub-regions and the merging of local results into a global solution (which can potentially suffer from accuracy issues), whereas our proposed approach considers the whole dataset and performs a single-pass analysis on it. Contrary to [9], when using the LG variation we no longer observe a correlation between space density and clustering efficiency. Milletti and Vulpetti [10] perform a geometry based abstraction of molecular structures. Specifically, they profile protein-binding pockets by using a shape-context-based method. This method describes the coarse distribution of the pocket shape with respect to a given point in its geometry. In [11] author has explored five different methods for missing value imputation namely Row Average Imputation, Mean Imputation, Median Imputation, k-Nearest Neighbour Imputation and combination of kNN based feature selection (kNNFS) and kNN-based imputation. In [12] author proposed a technique which uses crossover and mutation operation to getting better clustering result. It conduct experiments on seven datasets that are available in UCI machine learning repository. Two evaluation criteria namely silhouette coefficient and DB index are used.

III. PROBLEM IDENTIFICATION

In investigations of disease forms, a typical issue is to look for little atoms (ligands) that can cooperate with a bigger particle, for example, a protein when the protein is engaged with a disease state. More specifically, when ligands dock well in a protein, they can conceivably be utilized as a medication to stop or avert infections related with the protein's glitch. This is a typical kind of investigation done in medicate advancement examine. In this paper, for instance of its appropriateness, we apply our technique to basic science datasets containing substantial quantities of ligand adaptations. This paper aims to cluster all the related genes via clustering and MapReduce algorithm which smartly selects the ligands of similar types.

Customarily, a naive approach used to cluster similar structural biology compliances is through geometry-based clustering. Such strategy requires that information is put away in a concentrated area keeping in mind the end goal to apply the Root-Mean Square Deviation of every ligand with the various ligands in the dataset. The investigation requires the atomic dataset to be moved whole into a central server. This completely concentrated approach isn't adaptable and can bring about some serious storage and bandwidth problem on the server side.

IV. METHODOLOGY

In this section we present the proposed methodology for implementing octree based clustering using distributed vector distance calculation with Map Reduce and MPI.

We proposed a novel mechanism for analysing large amount of data in short period of time with highest accuracy. We have considered biological genes data. The experiments are performed on a distributed environment. The proposed methodology tries to overcome the existing mechanism in which the data movement took place for processing huge volume of data.
Proposed mechanism smartly selects small set of data and transfer between nodes and cluster those using octree based clustering mechanism. The major step comprises of a data reshaping that is connected to every individual data record simultaneously. This progression ventures applicable separated data properties into a kind of metadata. We execute a unique strategies to oversee and examine the reshaped data.
In the principal variety, called Metadata Movement, the removed properties are moved crosswise over hubs to assemble a worldwide perspective of the dataset; these properties are iteratively and privately broke down while hunting down some class or bunch union. The proposed method depends on the general subject of moving calculation to the data. We incorporate our algorithm into the MapReduce worldview [2] and utilize them to think about a complex basic science dataset of a great many ligand particles.

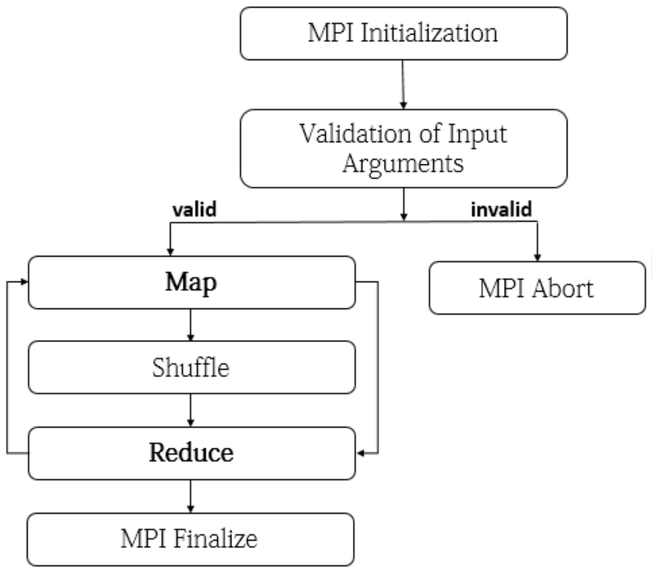

Fig. 2. Basic Workflow of MPI-MapReduce Clustering Framework

Step 01: Initialization of MPI Parameters

The MPI Parameters are prepared for execution in parallel environment. The function such as:

1. MPI_INIT()
2. MPI_Comm_Rank()
3. MPI_Comm_Size()

Step 02: Map Phase

Map phase used for octree key for each ligand.

Key → Octree Key

Value → Id for each ligand

It calculates the octree key for each ligand by taking input from the biological data which is the collection of various molecules or ligands.

Step 03: Shuffling Phase

At this point, the key, value pairs are partitioned among the r reduce processes based on the first couple of digits of the key. Next the reducer perform local reduction, and decides which level to go next. The distance for each of the properties are calculated here and passed to the reducer for different level processing.

Step 04: Reducing Phase

The reducer main task is to perform reduction of data and selecting best level for partitioning of data points. It iterated through whole document and parallel cluster all of them based on local properties. Finally, it combines result and stored into files with level information.

Algorithm 1: Octree Key Construction for each ligand

```
Input: Point coordinates $(\beta_x, \beta_y)$
Output: Octree Key's (K)

1: FOR j = 1 to K key's DO
2:    $x_{mid} = \left\lfloor \frac{1}{2} (x_{min} + x_{max}) \right\rfloor$
3:    $y_{mid} = \left\lfloor \frac{1}{2} (y_{min} + y_{max}) \right\rfloor$
4:    IF $\beta_x < x_{mid}$ $THEN$
5:        $\delta_0 = 0, x_{min} = x_{mid}$
6:    ELSE
7:        $\delta_0 = 1, x_{max} = x_{mid}$
8:    END IF
9:    IF $\beta_y < y_{mid}$ $THEN$
10:       $\delta_1 = 0, y_{min} = y_{mid}$
11:    ELSE
12:       $\delta_1 = 1, y_{max} = y_{mid}$
13:    END IF
14: Octree key calculated as:
15: Octree key[j] = $2^0 \beta_0 + 2^1 \beta_1$
16: END FOR
```

The algorithm-1 presents the octree key generation phases. Each point in the dataset represents a ligand points. To construct the octree keys we first determine the edge size for all ligand. The ligand confirms to x and y coordinates. The process shown in algorithm-1 repeated arbitrary number of times to generate complete octree keys $octree - key_{[1 \dots N_{key}]}$.

Algorithm 2: Octree Clustering

```
1: Input: Octree Key
2: Output: best octant with density $\leq Threshold_{density}$
3: $\underset{dept}{\min} = 1$
4: $\underset{dept}{\max} = K\ (keys)$
5: $\omega = \left\lfloor \frac{\underset{dept}{\max} - \underset{dept}{\min}}{2} \right\rfloor$ where, $\omega = octree\ dept$
6: WHILE
7:    FOR each octant in octree with depth ω do
8:        count octant density
9:    END FOR
10:   IF $density \leq Threshold_{density}$ THEN
11:       $\underset{dept}{\min} = \omega$
12:    ELSE
13:       $\underset{dept}{\max} = \omega$
14:    END IF
15    $\omega = \left\lfloor \frac{\underset{dept}{\max} - \underset{dept}{\min}}{2} \right\rfloor$
16: LOOP UNTIL $\underset{dept}{\max} = \underset{dept}{\min}$
```

The algorithm-2 shows the clustering of biological ligands. This step finds out the deepest octant whose number of confirmations should be greater than equal to $Threshold_{density}$. In octree based clustering technique, the algorithm keeps on searching for most dense octant. It first starts by looking for all nodes that is octants of certain depth ω. The ω is initializes to $\left\lfloor \frac{1}{2} N_{key} \right\rfloor$.

Algorithm 3: Parallel Octree Based Clustering Using MPI and Hadoop

```
1: Master node, $m_0$ assigns various number of slaves which are worker nodes as (P-1)
2: $m_0 \rightarrow$ splits the biological ligands into various smaller subsets.

        $\frac{D}{(P-1)}\ subsets$
   Where D is biological ligand dataset.

3: $m_0 \rightarrow$ transfer the spitted data to the worker nodes i.e. slaves.
4: For all worker nodes $m_i, m_i \in \{m_1, m_2, \ldots, m_{P-1}\}$ do in parallel
5: Worker nodes receives dataset
6: Apply Octree based clustering
7: Send the local centroid data to $m_0$
8: End loop
9: Master node, $m_0$ receives Centroid.
10: $m_0$: centroid are sorted.
11: Calculate global centroids by applying L2 NORM by formula:

        $|x| = \sqrt{\sum_{k-1}^{n} |x_k|^2}$

   Where $|x_k|$ denotes the complex modulus.
```

L2-norm is also known as least squares. Intuitively speaking, since a L2-norm squares the error (increasing by a lot if error > 1), the model will see a much larger error (e vs e2) than the L1-norm, so the model is much more sensitive to this example, and adjusts the model to minimize this error. If this example is an outlier, the model will be adjusted to minimize this single outlier case, at the expense of many other common examples, since the errors of these common examples are small compared to that single outlier case.

We have utilized L2-Norm method for calculating the global centroid distance in this project.

C. Properties Exchange

Given a ligand with p atomic coordinates (xi, yi, zi, with i from 1 to p), we perform a projection of the coordinates in their respective 3 planes (x, y), (y, z), and (z, x). Each projection results in a set of 2D points on the associated 2D plane.

Extracted properties can be unpredictably distributed across the 3-dimensional space of property points and across the nodes of the distributed memory system.

In Metadata movement, the extracted properties are exchanged among nodes across the distributed memory system to rebuild a global view of the information content in the scientific data so that similar properties eventually reside on the same node or nodes that are topologically close to each other as shown in fig. 3.

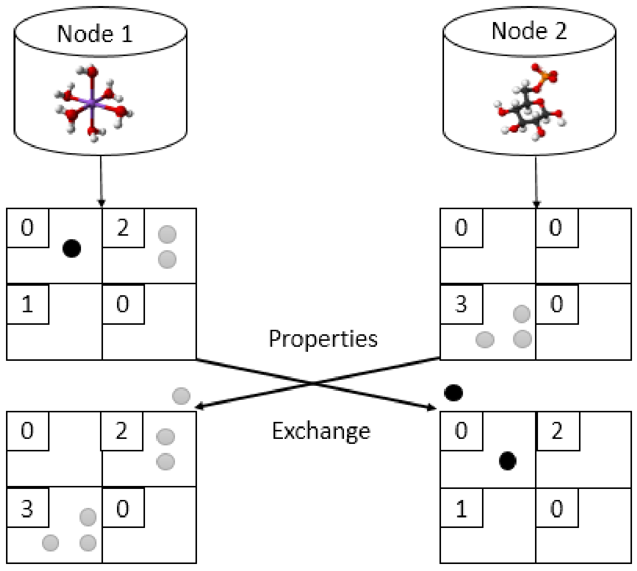

Fig. 3. Shows the Exchange of metadata properties in nodes

V. Results and Discussions

We have performed experiments using Biological dataset which consists of 76,889 ligand molecules. The system configuration are presented in table I.

Table I. System Configuration

S.NO	Attributes	Values
1	Software Used	Open-MPI MapReduce
2	Language Used	C++
3	Algorithm	Octree Based Clustering, L2 Norm Distance Function
4	Dataset	Biological Dataset

Using following system configuration the clustering of ligands are done. For clustering we have considered various threshold values such as 300, 500 and 1000 value for clustering of molecules. The threshold used to restrict the output of the cluster to contain specific ranges of molecules and density.

We have performed experiment while considering Euclidian distance (existing method) and L2 Norm for distributed computation (proposed method) for comparison.

The execution time and clustered records are shown in table II.

Table II. Clustered Records and Convergence for Execution Comparing Euclidian and L2 Norm Calculation

Threshold	Convergence Levels	Clustered Records	Level Information
300_eu	4	71	Level 4 Only
300_l2	9	64	Level 8 and 9
500_eu	4	31	Level 4 Only
500_l2	9	30	Level 8 and 9
1000_eu	4	3	Level 4 Only
1000_l2	9	11	Level 8 and 9

The outcomes gives clear indication that, for threshold value of 300 with Euclidian distance the convergence rate is 4 and records clustered are 71 but with L2 norm its converged at level 9 with 64 data records. But the data are uniformly distributed throughout level 8 and 9. Unlike with Euclidian distance the data points are present at level 4 only. But in fact the biological data points are to be distributed somewhere around level 8 and 9. Hence the work of distribution is smartly done by L2 norm distance vector calculation. Hence L2 is improved version of Euclidian for calculating the centroid points of distributed node clusters.

REFERENCES

[1] T. Estrada, B. Zhang, P. Cicotti, R. Armen, and M. Taufer, “Accurate analysis of large datasets of protein-ligand binding geometries using advanced clustering methods,” Computers in Biology and Medicine, vol. 42, no. 7, pp. 758–771, 2012.

[2] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Communic'ations of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[3] [3] S. J. Plimpton and K. D. Devine, “MapReduce in MPI for large-scale graph algorithms,” Parallel Computing, vol. 37, no. 9, Sep. 2011.

[4] A. Jain, “Bias, reporting, and sharing: Computational evaluations of docking methods,” Journal of Computer-Aided Molecular Design, vol. 22, no. 3-4, pp. 201–212, 2008.

[5] S. S. Shende and A. D. Malony, “The TAU parallel performance system,” International Journal of High Performance Computing Applications, vol. 20, no. 2, pp. 287–311, 2006.

[6] T. Tu, C. A. Rendleman, D. W. Borhani, R. O. Dror, J. Gullingsrud, M. O. Jensen, J. L. Klepeis, P. Maragakis, P. Miller, K. A. Stafford, and S. E. Shaw, “A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories,” in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC), 2008, pp. 1–12.

[7] H. G. Li, G. Q. Wu, X. G. Hu, J. Zhang, L. Li, and X. Wu, “K-means clustering with bagging and mapreduce,” in Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), 2011, pp. 1–8.

[8] A. Ene, S. Im, and B. Moseley, “Fast clustering using mapreduce,” in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011, pp. 681–689.

[9] R. L. F. Cordeiro, C. T. Jr, A. J. M. Traina, J. Lopez, U. Kang, and ´C. Faloutsos, “Clustering very large multi-dimensional datasets with MapReduce,” in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011, pp. 690–698.

[10] F. Milletti and A. Vulpetti, “Predicting polypharmacology by binding site similarity: From kinases to the protein universe,” Journal of Chemical Information and Modeling, vol. 50, no. 8, pp. 1418–1431, 2010

[11] H. L. Shashirekha and A. H. Wani, "Analysis of imputation algorithms for microarray gene expression data," 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Davangere, 2015, pp. 589-593.

[12] A. H. Beg and M. Z. Islam, "Clustering by genetic algorithm- high quality chromosome selection for initial population," 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, 2015, pp. 129-134.