A DISTRIBUTED APPROACH FOR MINING ASSOCIATION RULES IN PARALLEL ON PC CLUSTERS

P.1610

A PROJECT REPORT

Submitted by

D.NITHIYA PRIYA 71202104025
T.SIVA PRIYA 71202104040

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2006

BONAFIDE CERTIFICATE

Certified that this project report **"A DISTRIBUTED AFFROACH FOR MINING ASSOCIATION RULES IN PARALLEL ON PC CLUSTERS"** is the bonafide work of **"D.NITHIYA PRIYA (71202104025), T.SIVAPRIYA (71202104040)"** who carried out the project work under my supervision.

[illegible]

SIGNATURE

Dr. S. Thangasamy

HEAD OF THE DEPARTMENT

Department of
Computer Science and Engineering,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore-641006.

[illegible]

SIGNATURE

Mr. A. MuthuKumar

SUPERVISOR

Department of
Computer Applications,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore-641006.

Submitted for the Viva-voce Examination held on 2/5/2K6

[illegible]

Internal Examiner

[illegible] 02/05/06

External Examiner

We hereby declare that the project entitled "A Distributed Approach for Mining Association Rules in Parallel on PC Clusters" is a record of original work done by us and to the best of our knowledge; a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore

Date: 21-04-2005

[illegible]

(D.Nithiya Priya)

[illegible]

(T.Siva Priya)

We express our hearty gratitude to our beloved correspondent, Professor Dr.K. Arumugam, B.E (Hons), M.S. (U.S.A), M.I.E , for giving us this great opportunity to pursue this course.

We thank, Dr.K.K.Padmanabhan, B.Sc (Engg), M.Tech, Ph.D, Principal, Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facilities to work on this project.

We would like to make a special acknowledgement and thanks to Dr. S.Thangasamy, Ph.D, Professor and Head Of Department of Computer Science and Engineering, for his support and encouragement throughout the project.

We tender our special thanks to Ms Devaki B.E, M.S, Project Co-Ordinator, Department of Computer Science and Engineering for her valuable suggestions.

We express our deep sense of gratitude and gratefulness to our guide Mr. A.Muthukumar, MCA, M.Phil, Ph.D, Assistant Professor, Departmrnt of Computer Applications, for his supervision, tremendous patience, active involvement and guidance.

valuable suggestions and advice.

We would like convey our honest thanks to all the members of staff of the Department for their unlimited enthusiasm and experience from which we have greatly benefited.

We express our profound gratitude to our parents and friends for their moral support.

Above all we thank the CREATOR of this beautiful Planet for his grace throughout our endeavors.

PC clusters have become popular in parallel processing. They do not involve specialized inter processor networks, so the latency of data communications is rather long. And for PC clusters, load balancing among the nodes (clients) becomes a more critical issue in attempts to yield high performance.

Data mining normally involves processing large voluminous data. Doing all the mining processes in a single system consumes a lot of CPU time. Hence in order to reduce the computation time, we distribute the data to other systems. These are the client systems. We also have a server which co-ordinates the activities between the clients.

When more clients are involved, using them efficiently is of primary importance. In our project the idle time of the clients during processing is kept minimal by considering the CPU utilization on every node. We also implement the Apriori algorithm for mining association rules.

CPU – Central Processing Unit.

I/O - Input/Output.

PC – Personal Computer.

KDD – Knowledge Discovery in Databases.

DM – Data Mining

DDM – Distributed Data Mining.

INTRODUCTION

INTRODUCTION

This chapter gives an introduction to the data mining process, distributed data mining and software intelligent agents and its uses in different fields. In addition, this chapter clearly specifies the objective of this project work and gives brief and clear description about the statement of the problem.

1.1. DATAMINING

With the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the "essence" of information stored, and the discovery of patterns in raw data.

DEFINITION:

Data Mining, also popularly known as **Knowledge Discovery in Databases (KDD)**, refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases

1.1.1.What kinds of information are we collecting?

We have been collecting a myriad of data, from simple numerical measurements and text documents, to more complex information such as spatial data, multimedia channels, and hypertext documents. Here is a non-exclusive list[3] of a variety of information collected in digital form in databases and in flat files.

- Business transactions
- Scientific data
- Medical and personal data
- Surveillance video and pictures
- Satellite sensing Games
- Digital media
- CAD and Software engineering data
- Virtual Worlds
- Text reports and memos (e-mail messages)
- The World Wide Web repositories

1.1.2. Knowledge Discovery

The Knowledge Discovery in Databases[3] process comprises of a few steps leading from raw data collections to some form of new knowledge[4].

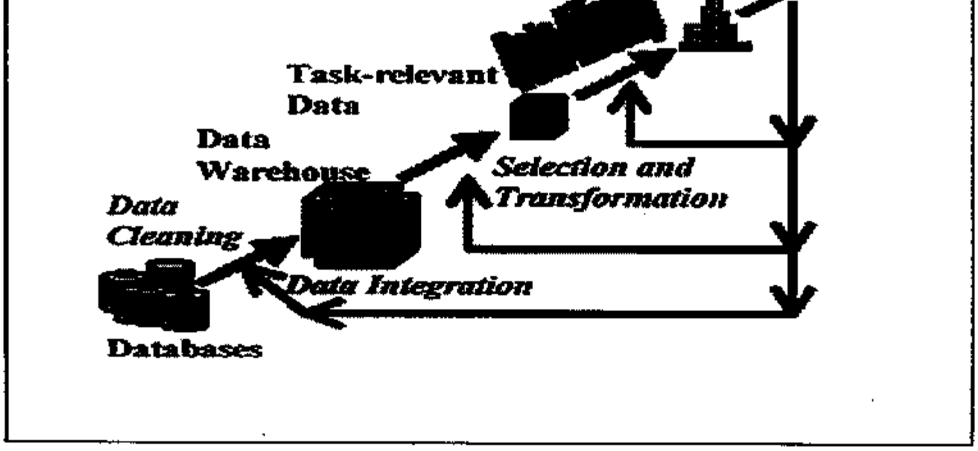


Figure 1.1 Knowledge Discovery Process

The iterative process consists of the following steps:

- **Data cleaning**: also known as data cleansing, it is a phase in which noise data and irrelevant data are removed from the collection.
- **Data integration**: at this stage, multiple data sources, often heterogeneous, may be combined in a common source.
- **Data selection**: at this step, the data relevant to the analysis is decided on and retrieved from the data collection.
- **Data transformation**: also known as data consolidation, it is a phase in which the selected data is transformed into forms appropriate for the mining procedure.
- **Data mining**: it is the crucial step in which clever techniques are applied to extract patterns potentially useful.
- **Pattern evaluation**: in this step, strictly interesting patterns representing knowledge are identified based on given measures.

visualization techniques to help users understand and interpret the data mining results.

It is common to combine some of these steps together.

1.1.3.What kind of Data can be mined?

Data mining is not specific to one type of media or data. Data mining should be applicable to any kind of information repository. However, algorithms and approaches may differ when applied to different types of data[4]. Data mining is being put into use and studied for databases, including relational databases, object-relational databases and object-oriented databases, data warehouses, transactional databases, unstructured and semi structured repositories such as the World Wide Web, advanced databases such as spatial databases, multimedia databases, time-series databases and textual databases.

1.1.4.What can be discovered?

The kinds of patterns that can be discovered depend upon the data mining tasks employed. By and large, there are two types of data mining tasks: **descriptive data mining** tasks that describe the general properties of the existing data, and **predictive data mining** tasks that attempt to do predictions based on inference on available data. The data mining functionalities and the variety of knowledge they discover are briefly presented in the following list:

- **Characterization**: Data characterization[5]is a summarization of general features of objects in a target class, and produces what is called **characteristic rules**. The data relevant to a user-specified class are normally retrieved by a

- **Discrimination**: Data discrimination produces what are called **discriminate rules** and is basically the comparison of the general features of objects between two classes referred to as the **target class** and the **contrasting class**
- **Association analysis**: Association analysis[4] is the discovery of what are commonly called **association rules**. It studies the frequency of items occurring together in transactional databases, and based on a threshold called **support,** identifies the frequent item sets. Another threshold, **confidence**, which is the conditional probability than an item appears in a transaction when another item appears, is used to pinpoint association rules.
- **Classification**: Classification analysis[14] is the organization of data in given classes. Classification approaches normally use a **training set** where all objects are already associated with known class labels. The classification algorithm learns from the training set and builds a model. The model is used to classify new objects
- **Prediction**: Prediction has attracted considerable attention given the potential implications of successful forecasting in a business context. There are two major types of predictions: one can either try to predict some unavailable data values or pending trends, or predict a class label for some data. The latter is tied to classification. Once a classification model is built based on a training set, the class label of an object can be foreseen based on the attribute values of the object and the attribute values of the classes. Prediction is however more often referred to the forecast of missing numerical values, or increase/ decrease trends in time related data. The major idea is to use a large number of past values to consider probable future values.

unknown and it is up to the clustering algorithm to discover acceptable classes. Clustering is also called **unsupervised classification**, because the classification is not dictated by given class labels. There are many clustering approaches all based on the principle of maximizing the similarity between objects in a same class (intra-class similarity) and minimizing the similarity between objects of different classes (inter-class similarity).

- **Outlier analysis**: Outliers are data elements that cannot be grouped in a given class or cluster. Also known as exceptions or surprises, they are often very important to identify. While outliers can be considered noise and discarded in some applications, they can reveal important knowledge in other domains, and thus can be very significant and their analysis valuable.
- **Evolution and deviation analysis**: Evolution and deviation analysis[3] pertain to the study of time related data that changes in time. Evolution analysis models evolutionary trends in data, which consent to characterizing, comparing, classifying or clustering of time related data. Deviation analysis, on the other hand, considers differences between measured values and expected values, and attempts to find the cause of the deviations from the anticipated values. It is common that users do not have a clear idea of the kind of patterns they can discover or need to discover from the data at hand. It is therefore important to have a versatile and inclusive data mining system that allows the discovery of different kinds of knowledge and at different levels of abstraction. This also makes interactivity an important attribute of a data mining system.

location. Recently, distributed data mining systems have exploited wide area, high performance networks to mine large amounts of distributed scientific and health care data.

Distributed Data mining is expected to perform partial analysis of data at individual sites and then to send the outcome as partial results to other sites where it is sometimes required to be aggregated to the global result. The important issues to be considered while performing distributed data mining includes,

- ✓ **Nature of data sets**

The data sets to be used for processing will be massive and it is necessary to perform required transformation before these data sets are mined in distributed environment. These data sets are inherently distributed both geographically and physically.

- ✓ **Networks**

The distributed data mining process is capable of working under limited bandwidth and with the limited computing resources at nodes

- ✓ **Privacy and security**

Sensitive data can be handled effectively and extra measures should be taken to secure the data. Also, we should keep in mind that our goals should be shared and not the entire data under distributed data mining environment.

The main objective of this project work is to implement the distributed data mining approach for mining data with the maximum resource utilization of the existing CPU idle time.

1.4. PROBLEM STATEMENT :

Data mining usually involves large voluminous of data. Accomplishing the task in a single machine generally consumes large amount of time. In order to minimize the utilization time we distribute the data to other systems (client systems) and get the work done. When the data are distributed to various systems for processing, it is necessary to use the resources of these systems efficiently. Here, maximal utilization of idle time of these systems is in question. Henceforth, it is essential to identify some techniques to maximize the usage of these computers and to reduce the idle time.

This distributed data mining activities will help in forecasting the business trends and analyzing the customer's behavior in any shop or market.

SYSTEM ANALYSIS

SYSTEM ANALYSIS

EXISTING SYSTEM

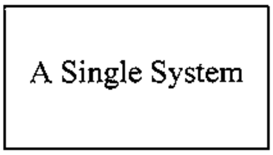


Figure 2.1 Single System

- All data are processed on a single machine.
- More time consumption.
- Decrease in speed.

PROPOSED SYSTEM

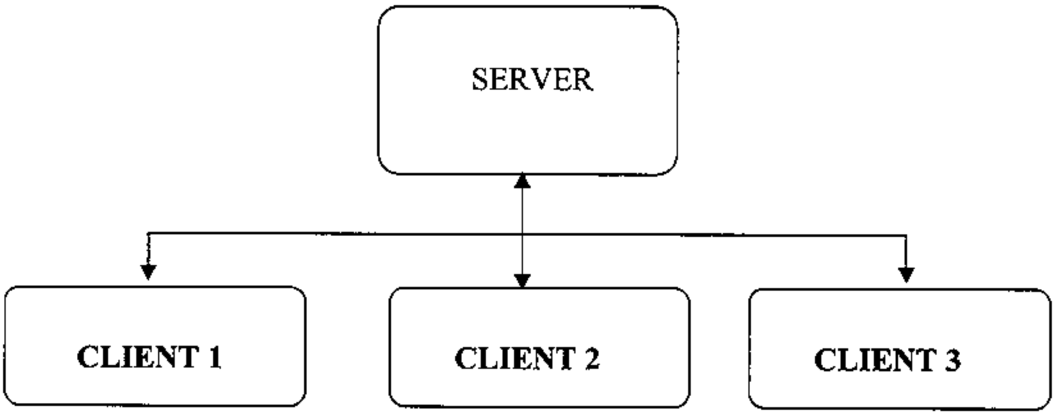


Figure 2.2. Proposed system

- The task is shared between three systems.
- Decrease in time consumption.
- Increase in processing speed.

SYSTEM ARCHITECTURE

The proposed system consists of two logical components.

- ✓ **Client** - There may be in N-number of clients connected in intranet. Here the client receives data sets and mining operation to be performed as parameters from the server agent. Local agent running in the client machine takes care performing specified operation and storing the final results. In the beginning, the client side agent sends resource utilization information to the server machine. client agent along with the mining operation to be performed.

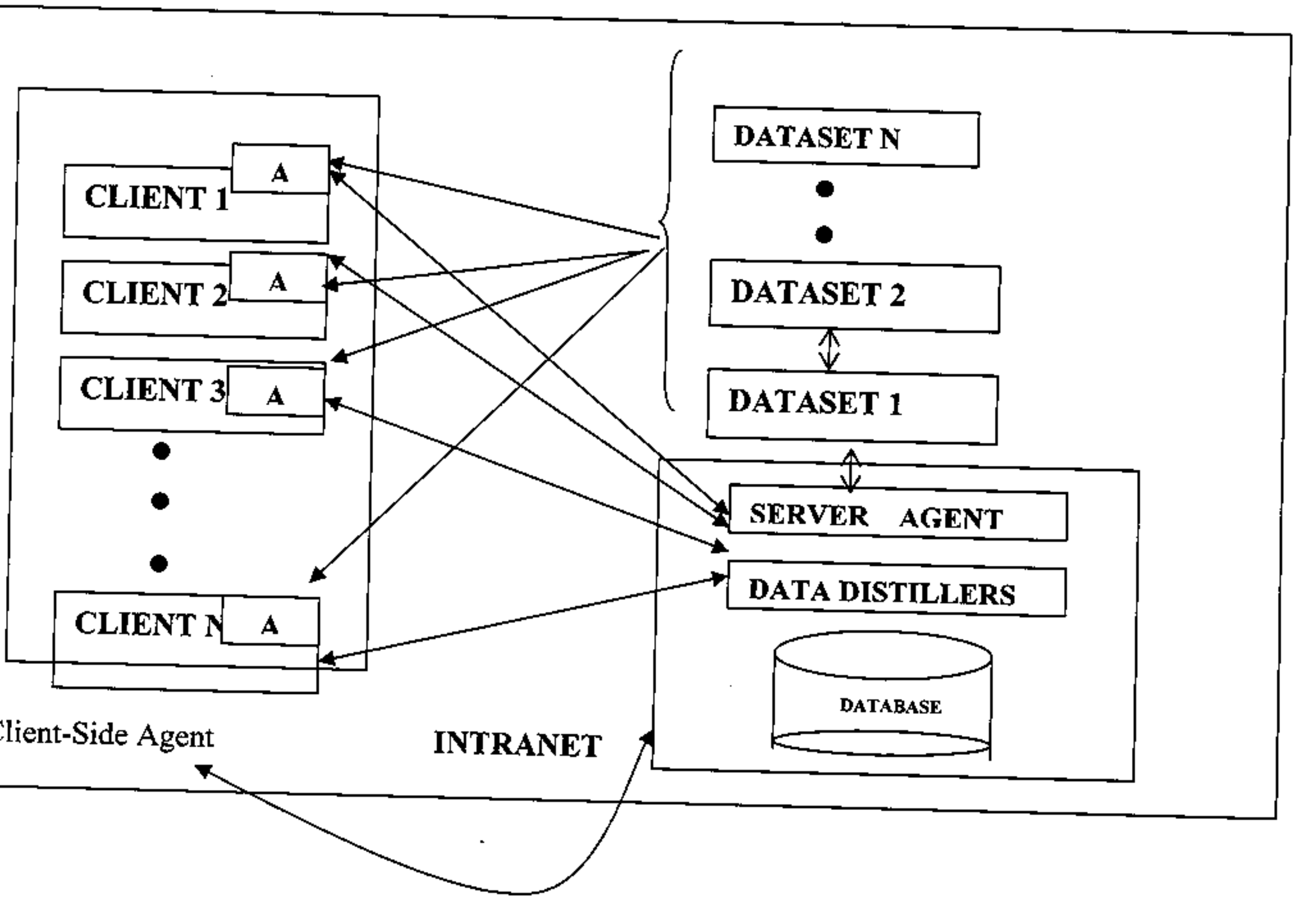


Figure 3.1. System Architecture

having following responsibilities

- ✓ Monitoring the load on the connected client machines
- ✓ Distributing the data sets based on this load information from the client agent along with the mining operation to be performed.
- ✓ Integrating the collected results from the various clients, for global result.

SYSTEM DESIGN

The system is considered as a set of components with clearly defined behavior. The focus is on the identification of the modules and how the modules should be interconnected .The list of modules in the system are as follows.

4.1 SERVER SIDE MODULES:

- File split
- Main module
- File merge

4.1.1 FILE SPLIT

Based on the CPU usage calculate the load that can be sent to clients.The input file is split into 3 & written in 3 separate files .These files are then sent to clients. Clients begin the appropriate operation.

clients[i] = [line count/total] *[1/ cpuload[i]

where total = sum of (1/cpuload[i])

Files used :

Cpuload.txt $\rightarrow$ gives the CPU usage of each client.

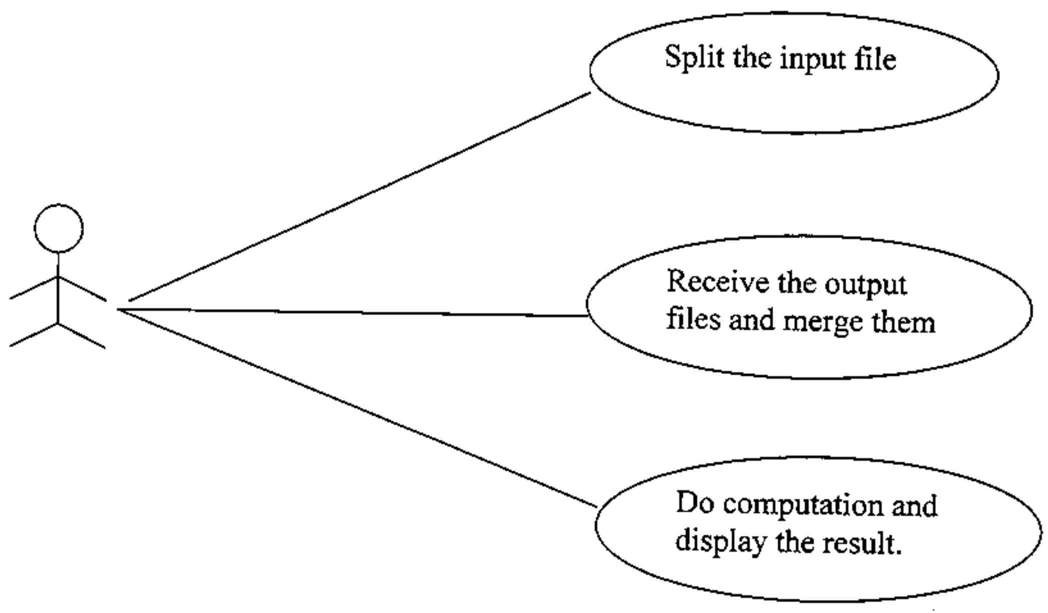
4.1.2 MAIN MODULE

The cpu usage received from the clients are written to a file (cpuload.txt). After getting the choice from the user, the appropriate itemset is invoked on the client and files are created according to the choice (Any one the itemsets).
The output files are received from the clients and merging is done.

output files. Create a common file which will hold the merged output. Do merging accordingly.

4.1.4 USE CASE DIAGRAM

Based on the calculated CPU idle time, the Server splits the data among the workstations. Finally, accepts the processed results from the clients and display them in the required format. The figure best illustrates the identified responsibilities of the server.



Split the input file
Receive the output files and merge them
Do computation and display the result.

Figure4.1 USE CASE Diagram – Server Side

- Sending CPU usage to the server (This is done by the agent).
- Implementation of Apriori algorithm.
- Sending results to the server.

4.2.1 SENDING CPU USAGE TO THE USER

By calling the method getCPUUsage() of sysInfo object, the CPU usage is obtained .
This utilization information is sent to the server.

4.2.2 IMPLEMENTATION OF APRIORI ALGORITHM

The implementation of Apriori algorithm [2] basically consists of the following steps

a. Join Step: Ck is generated by joining Lk-1with itself
b. Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

PSEUDO CODE

```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k
L1 = {frequent items};
for (k = 1; Lk !=∅;  k++) do begin
    Ck+1 = candidates generated from Lk;
  for each transaction t in database do
```

```
    end
  return $\cup_k L_k$;
```

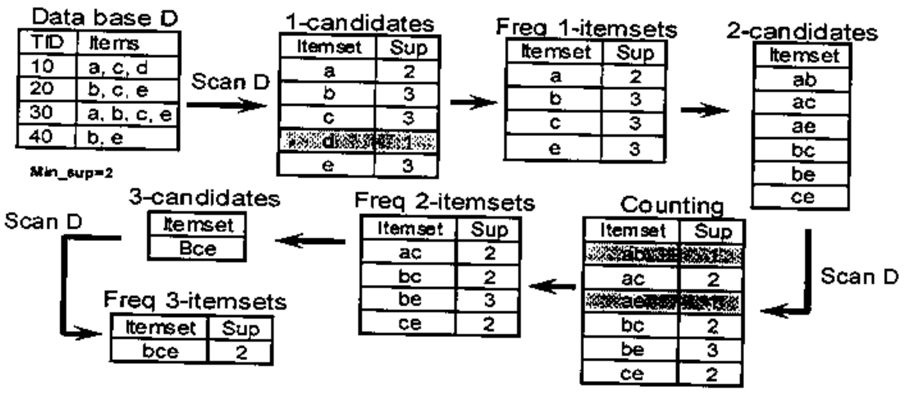


Figure 4.2 Aproiri Algorithm

GENERATION OF CANDIDATE SET

```
Suppose the items in Lk-1 are listed in an order
Step 1: self-joining Lk-1
        insert into Ck
        select p.item1, p.item2, ..., p.itemk-1, q.itemk-1 from Lk-1 p, Lk-1q
         where   p.item1=q.item1, ..., p.itemk-2=q.itemk-2,
        p.itemk-1 < q.itemk-1
Step 2: pruning
        forall itemsets c in Ck do
                forall (k-1)-subsets s of c do
                        if (s is not in Lk-1) then delete c from Ck
```

4.2.3 SENDING RESULTS TO THE SERVER

After the computation of the itemset, the results are sent to the client for merging.

4.2.4 USE CASE DIAGRAM

The basic responsibilities of clients include accepting the data sets, perform association rule mining and send the processed results to the server using agents. Apriori algorithm is used for performing the association rule mining. The figure reveals the responsibilities of the clients.

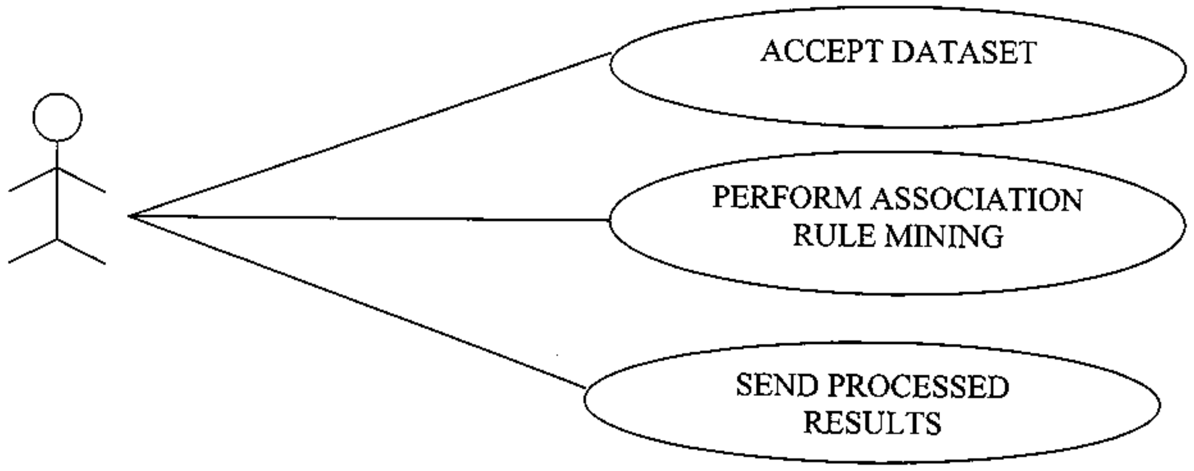


Figure 4.3 USE CASE Diagram – Client Side

The primary responsibility of the agent is to obtain the CPU usage time of clients and send this information to the server. The figure portrays these responsibilities clearly

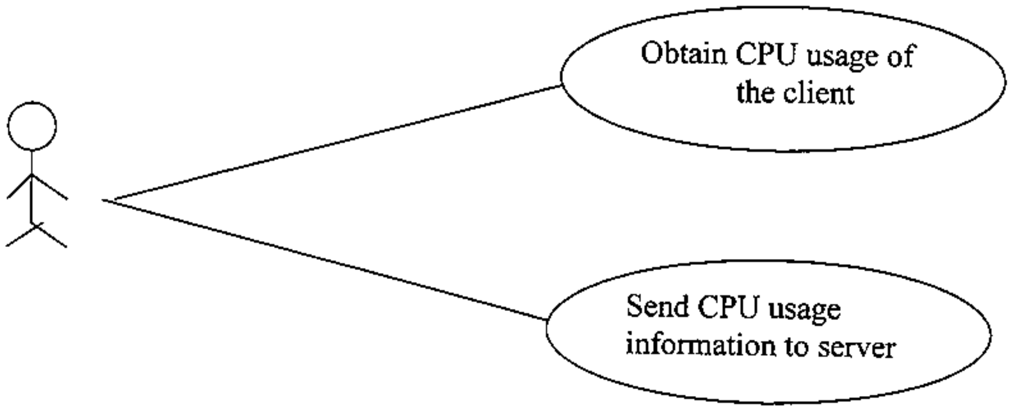

Obtain CPU usage of the client
Send CPU usage information to server

Figure 4.4 USE CASE Diagram – Agent

server requests the CPU usage time from the client. On this request, the agent on the client side obtains the CPU usage information and sends it to the client. Based on this information the server splits the input file and sends it to the clients. The client does the required operation and returns the processed data to the server.

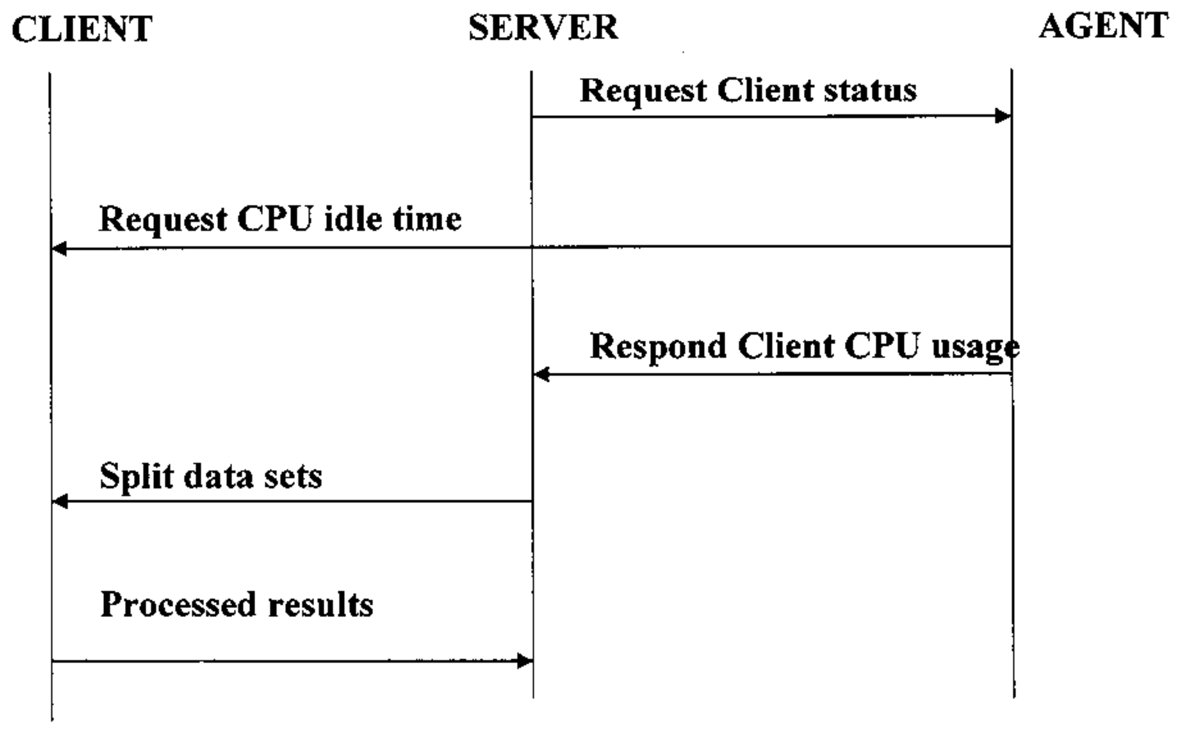


FIGURE 4.5. Interaction Diagram for the proposed system

IMPLEMENTATION

This chapter elucidates the software and hardware requirements of this project, experimental results and sample screen shots.

5.1. CHOICE OF HARDWARE AND SOFTWARE

5.1.1 SOFTWARE:

This project is implemented in JDK 1.4 version. The Apriori algorithm and the intelligent agent program that monitors the CPU usage time in the identified workstations and all the server side modules is implemented using JDK 1.4. This project requires JDK 1.4 to be installed in all the systems.

5.1.2 HARDWARE:

This project does not impose any special hardware requirements. Thus, the hardware requirements for this project work is very limited to have the following configuration

- ❖ PROCESSOR: PIV
- ❖ CLOCK: 2.8 GHZ
- ❖ CACHE MEMORY: 512 KB
- ❖ 256 MB RAM, 40 GB HARD DISK

PERFORMANCE ANALYSIS

6.1 DISTRIBUTED SYSTEM VS SINGLE SYSTEM

We do the mining process with input file of various sizes. First the size of the input is kept as 250KB. 3 ITEM SET is run for this input file on a single system. The time taken for producing the final output is noted. The same input file is used and 3ITEM SET is run on a distributed system with three clients. The time taken for producing the final output is noted. Then the size of the input file is changed to 500 KB and then 750 KB and finally 1000 KB and the above process is repeated.

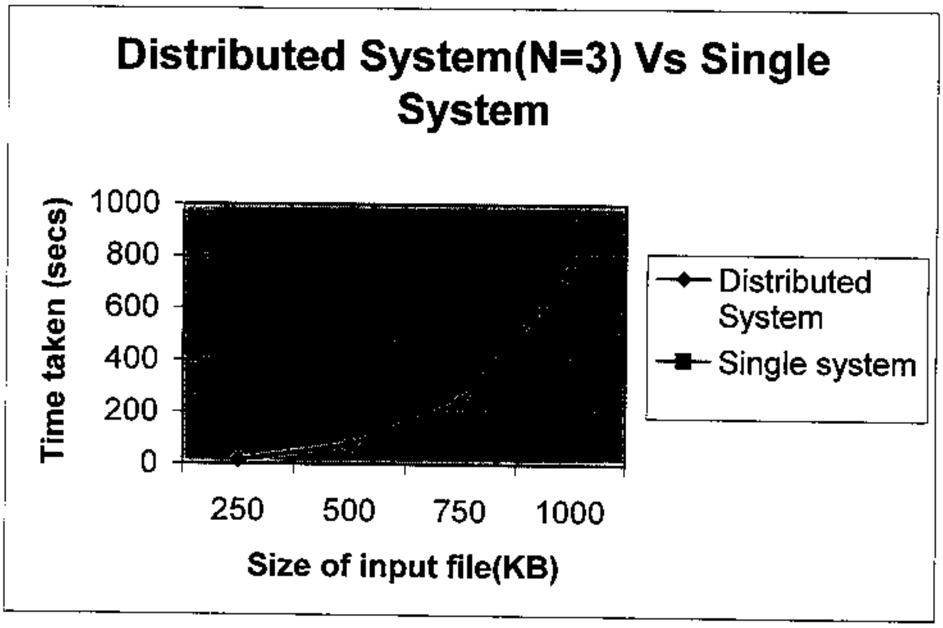


Figure 6.1. Distributed System Vs Single System

It can be easily concluded that the distributed system yields better performance than single system. The performance also greatly depends on the size of the input used. We can see that the performance difference between the systems is high when the size of the input is 500 KB and 750 KB.

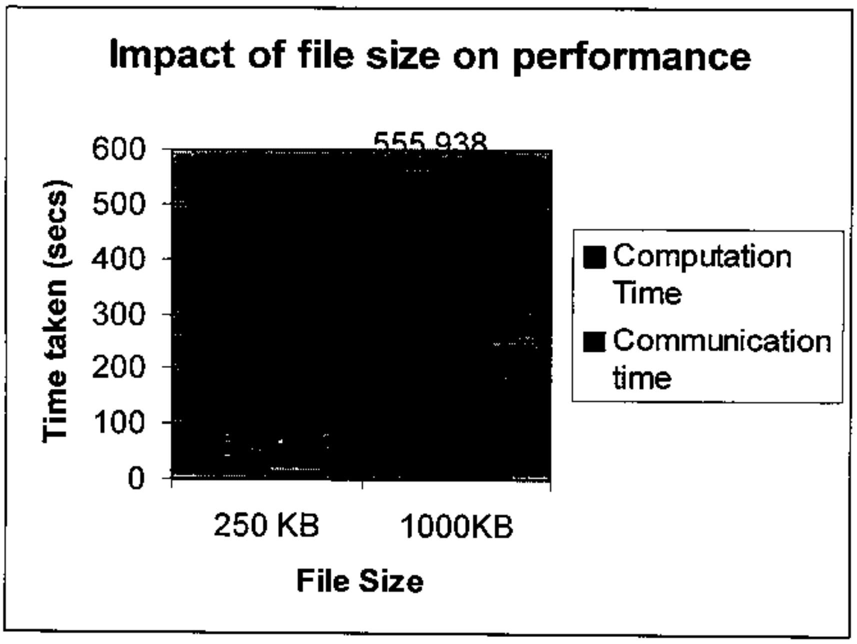


Figure 6.2 Impact of File Size On Performance

When the size of the input file is very small or large then the difference is not highly pronounced. If the input size is small the communication overhead will be longer than the computation time.

On the other hand if the size of the input file is too large it inadvertently affects the performance due to cache memory limitations. Here, computation time will be longer than the communication overhead.

Now we conduct experiments to analyze the performance by statically distributing the file and distributing the file based on CPU usage.

We distribute to each client 125KB of the input file at a time. (No. of clients =3). So after first distribution to all the clients, the size of input file that is remaining is 625KB. The clients process their share of data and return the results to the server. Again we distribute 125KB to all the clients and the above process is repeated. This is done till end of the input file is reached and the time taken for the entire process is calculated.

Secondly we distribute the input file based on the CPU usage to the clients. The clients process the data and return the results. The time taken for this is also noted. From the obtained data we plot the following graph.

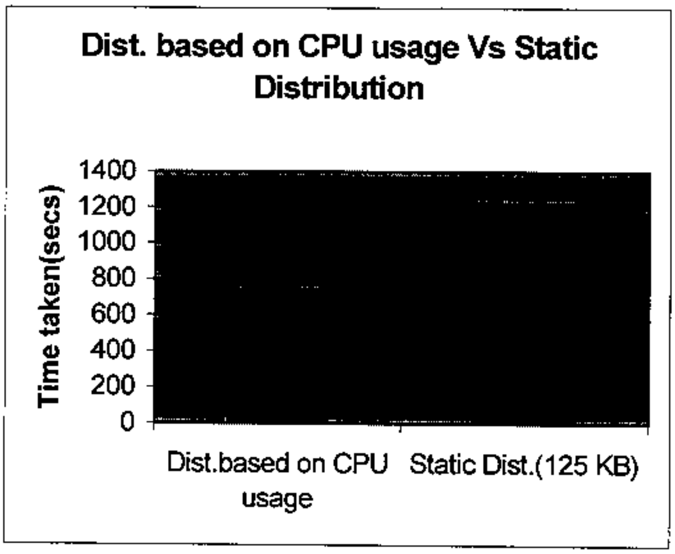


Figure 6.3 Distribution based on CPU Usage Vs Static Distribution

From the graph it can be clearly concluded that distribution based on CPU usage yields better performance than static distribution.

I. Distributed system gives better performance than single system.

II. When input file size is neither very large nor very small optimal performance is obtained.

III. Distribution based on CPU usage gives comparatively better performance than static distribution.

FUTURE SCOPE

FUTURE SCOPE

7.1 FUTURE SCOPE

❖ This project can be extended to N number of workstations to perform distributed data mining activities and to achieve efficient overall performance.

❖ This project used association rule mining technique. This can be extended to use other data mining techniques like Cluster Analysis, Classification Analysis and Neural networks etc.,

❖ The intelligent agent designed for monitoring CPU time can be extended to move assigned data sets from one machine to other automatically, when the CPU usage time exceeds the threshold value.

❖ This project can be extended to other types of data such as scientific data, satellite data etc., as per the requirement.

CONCLUSION

CONCLUSION

8.1. CONCLUSION

The data mining activity for mining association rules on PC clusters is done by distributing the load among three clients. This project also achieves the objective of effective utilization of computing resources in efficient manner. Furthermore this project forecasts and analyses the behaviour of customers.

APPENDICES

side and the two main modules of the server side.

Server side:

FILE SPLIT;

```
public class FileSplit
 {
   FileSplit()
      {
        try
            {
                String inputline,outputline;
                int count=0;
                double tot=0.0,tot1=0.0,lineCount=0.0;
                String line;
              int c=0;
              String s="Input1.txt", data;
              String cpaddr[] = new String[5];
              int inc=1;
                double cpload[] = new double[5];
              double cpload1[] = new double[5];
                double trans[] = new double[5];
               int cl=3;

               File f= new File(s);
               StringTokenizer stoken;
               BufferedReader bbr = new BufferedReader(new
                                                FileReader("cpuload.txt"));
               BufferedReader d = new BufferedReader(new FileReader(s));
               BufferedReader d1 = new BufferedReader(new FileReader(s));
                while (d1.readLine()!=null)
                      {
                            lineCount++;
                      }

                while((data = bbr.readLine()) !=null)
                      {
```

```
                                        cpaddr[inc] = stoken.nextToken();
                                        cpload1[inc] =
                              Double.parseDouble(stoken.nextToken());
                                         inc++;
                                     }

                    }

          for(int j=1; j<=cl; j++)
                    tot = tot+(1/cpload1[j]);

          for(int j=1; j<=cl; j++)
                    trans[j] = (lineCount/tot)*(1/cpload1[j]);
                    trans[1]=trans[1]/2;
                    trans[cl]=trans[cl]+trans[1];

          for(int i=1;i<=cl;i++)
               {
                    File f1=new File("file"+(i)+".txt");
                    FileOutputStream fout=new FileOutputStream(f1);
                    PrintStream ps=new PrintStream(fout);

                    while((line=d.readLine())!=null)
                         {

        ps.println(line);
        c++;
        if(c>trans[i])
                {
                        ps.close();
                        c=0;
                        break;
                }

        }
      }
}
```

```
        }
    }
}
```

MERGING ONE ITEM SET

```
public class merge_one
{
  merge_one(int cl1,double rd_th)
       {
         System.out.println("Entering merge_one");
         try
             {
              DecimalFormat dfd = new DecimalFormat("0.00");
              int count[]=new int[8000];
             double TH_value[] = new double[8000];
             double dd=3300.00;
             String data,data1;
             StringTokenizer st,st1;
             BufferedReader br;

             int k=0,l=0,cl;
             double  ratio =0.0;
             FileWriter fw = new FileWriter("one_out.txt");
             FileWriter final_one = new FileWriter("final_one.txt");

             for(int i=0;i<8000;i++)
                   count[i]=0;
                    System.out.print("Reading files...");
             for(int g=1;g<=cl1;g++)
                    {
                          br=new BufferedReader(new
                                        FileReader("one_out"+g+".txt"));
                          while((data=br.readLine())!=null)
                              {
                               st=new StringTokenizer(data,"     ");
                                    while(st.hasMoreTokens())
                                 {
```

```
                          {
                         k=Integer.parseInt(st1.nextToken());
                         l=Integer.parseInt(st1.nextToken());
                         count[k]=count[k]+l;
                        }
                     }
                  }
              }
      System.out.println("File closed. Producing output...");
      for(int i=1;i<8000;i++)
       {
          if(count[i]!=0)
           TH_value[i] = (double) (count[i] / dd)*100.0;
       }
     for(int d=0;d<8000;d++)
      {
            if(TH_value[d]>=rd_th)
           fw.write(""+d+"\r\n");
      }
    fw.close();
    for(int d=1;d<8000;d++)
      {
           if(TH_value[d]>=rd_th)
            {
                 final_one.write(""+d+"\t("+dfd.format(TH_value[d])+"%
                                                                  )"+"\r\n");
            }
      }
   final_one.close();
   }
   catch(Exception e)
     {
     }
   System.out.print("Process completed.");
   }
}
```

FINAL CLASS

```
public class Final
{
public static void main(String arg[])
{
	Thread t=Thread.currentThread();
	double supp, conf;

	try
	{
		server2 ss2 = new server2();
		Thread.sleep(10000);

		FileWriter z=new FileWriter("one_out1.txt");
		FileWriter z1=new FileWriter("one_out2.txt");
		FileWriter z2=new FileWriter("one_out3.txt");

		long startTime;
			long endTime;
		double time;
		 int choice, cl=1;



		ServerSocket se2 = new ServerSocket(91);
		ServerSocket se21 = new ServerSocket(92);
		ServerSocket se22 = new ServerSocket(93);
		do
		{
		Socket plug,plug1,plug2;

		System.out.println("\n\n1-> One   ItemSet\n2-> Two   ItemSet\n3->
Three ItemSet\n4-> Four  ItemSet\n5-> Five  ItemSet\n6-> Exit ");
		System.out.print("\nEnter the Choice:");
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		choice = Integer.parseInt(br.readLine());
```

```
          plug1= se21.accept();
          plug2= se22.accept();

          PrintWriter PW=new PrintWriter(plug.getOutputStream(),true);
          PrintWriter PW1=new PrintWriter(plug1.getOutputStream(),true);
          PrintWriter PW2=new PrintWriter(plug2.getOutputStream(),true);

          PW.println(choice);
          PW1.println(choice);
          PW2.println(choice);

          startTime = System.currentTimeMillis()-10000;
          switch (choice)
          {
               case 1:
                         System.out.print("Enter the minimum support threshold
value :");
                         BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
                         supp= Double.parseDouble(br1.readLine());

                         server ob11=new
server(choice,"file1.txt","file2.txt","file3.txt","one_out1.txt","one_out2.txt","one_o
ut3.txt");
                         Thread.sleep(10000);
                         merge_one ob11m=new merge_one(cl,supp);
                         Thread.sleep(10000);
                         break;

               case 2:
                         System.out.print("Enter the  minimum support threshold
value :");
                         BufferedReader br2 = new BufferedReader(new
InputStreamReader(System.in));
                         supp= Double.parseDouble(br2.readLine());
                         System.out.print("Enter the confidence threshold value
:");
                         conf= Double.parseDouble(br2.readLine());
ut3.txt");
                        Thread.sleep(10000);
                        merge_one ob21m=new merge_one(cl,supp);
                        Thread.sleep(10000);



                        server ob22=new
server(choice,"one_out.txt","one_out.txt","one_out.txt","two_out1.txt","two_out2.t
xt","two_out3.txt");
                        Thread.sleep(10000);
                        merge_two ob22m=new merge_two(supp,conf);
                        Thread.sleep(10000);
                        break;

                case 3:
                        System.out.print("Enter the minimum support threshold
value :");
                        BufferedReader br3 = new BufferedReader(new
InputStreamReader(System.in));
                        supp= Double.parseDouble(br3.readLine());
                        System.out.print("Enter the confidence threshold value
:");
                        conf= Double.parseDouble(br3.readLine());

                        server ob31=new
server(choice,"file1.txt","file2.txt","file3.txt","one_out1.txt","one_out2.txt","one_o
ut3.txt");
                        Thread.sleep(10000);
                        merge_one ob31m=new merge_one(cl,supp);
                        Thread.sleep(10000);



                        server ob32=new
server(choice,"one_out.txt","one_out.txt","one_out.txt","two_out1.txt","two_out2.t
xt","two_out3.txt");
                        Thread.sleep(10000);
                        merge_two ob32m=new merge_two(supp,conf);
```

```
server(choice,"two_out.txt","two_out.txt","two_out.txt","three_out1.txt","three_out
2.txt","three_out3.txt");
                        Thread.sleep(10000);
                        merge_three ob33m=new merge_three(supp,conf);
                        Thread.sleep(10000);
                        break;

                  case 4:
                        System.out.print("Enter the minimum support threshold
value :");
                        BufferedReader br4 = new BufferedReader(new
InputStreamReader(System.in));
                        supp= Double.parseDouble(br4.readLine());
                        System.out.print("Enter the confidence threshold value
:");
                        conf= Double.parseDouble(br4.readLine());

                        server ob41=new
server(choice,"file1.txt","file2.txt","file3.txt","one_out1.txt","one_out2.txt","one_o
ut3.txt");
                        Thread.sleep(10000);
                        merge_one ob41m=new merge_one(cl,supp);
                        Thread.sleep(10000);

                        server ob42=new
server(choice,"one_out.txt","one_out.txt","one_out.txt","two_out1.txt","two_out2.t
xt","two_out3.txt");
                        Thread.sleep(10000);
                        merge_two ob42m=new merge_two(supp,conf);
                        Thread.sleep(10000);


                        server ob43=new
server(choice,"two_out.txt","two_out.txt","two_out.txt","three_out1.txt","three_out
2.txt","three_out3.txt");
                        Thread.sleep(10000);
                        merge_three ob43m=new merge_three(supp,conf);
                        Thread.sleep(10000);
```

```
server(choice,"three_out.txt","three_out.txt","three_out.txt","four_out1.txt","four_
out2.txt","four_out3.txt");
                              Thread.sleep(10000);
                              merge_four ob44m=new merge_four(supp,conf);
                              Thread.sleep(10000);
                              break;

                        case 5:
                              System.out.print("Enter the minimum support threshold
value :");
                              BufferedReader br5 = new BufferedReader(new
InputStreamReader(System.in));
                              supp= Double.parseDouble(br5.readLine());
                              System.out.print("Enter the confidence threshold value
:");
                              conf= Double.parseDouble(br5.readLine());

                              server ob51=new
server(choice,"file1.txt","file2.txt","file3.txt","one_out1.txt","one_out2.txt","one_o
ut3.txt");
                              Thread.sleep(10000);
                              merge_one ob51m=new merge_one(cl,supp);
                              Thread.sleep(10000);



                              server ob52=new
server(choice,"one_out.txt","one_out.txt","one_out.txt","two_out1.txt","two_out2.t
xt","two_out3.txt");
                              Thread.sleep(10000);
                              merge_two ob52m=new merge_two(supp,conf);
                              Thread.sleep(10000);

                              server ob53=new
server(choice,"two_out.txt","two_out.txt","two_out.txt","three_out1.txt","three_out
2.txt","three_out3.txt");
                              Thread.sleep(10000);
                              merge_three ob53m=new merge_three(supp,conf);
```

```
                                server ob54=new
server(choice,"three_out.txt","three_out.txt","three_out.txt","four_out1.txt","four_
out2.txt","four_out3.txt");
                                Thread.sleep(10000);
                                merge_four ob54m=new merge_four(supp,conf);
                                Thread.sleep(10000);

                                server ob55=new
server(choice,"four_out.txt","four_out.txt","four_out.txt","five_out1.txt","five_out
2.txt","five_out3.txt");
                                Thread.sleep(10000);
                                merge_five ob65m=new merge_five(supp,conf);
                                Thread.sleep(10000);
                                break;

                        case 6:
                                System.exit(0);
                                break;

                }
                endTime = System.currentTimeMillis()-(choice*20000);
                time = (endTime - startTime) / 1000.0;
                System.out.println("Run time for "+choice+"-ItemSet :
"+time+"seconds");
                plug.close();
                plug1.close();
                plug2.close();
                }while(choice!=6);

        }
        catch(Exception e)
        {       }
}
}


public class server
```

```
                                                    outfile2,String outfile3)
    {
       Server1 s1=new Server1("localhost",81,infile1,outfile1);
       Server1 s2=new Server1("localhost",83,infile2,outfile2);
       Server1 s3=new Server1("localhost",85,infile3,outfile3);

       SS ss1=new SS(ch);
     }
}



class Server1 implements Runnable
  {
     Thread t;
     InetAddress ip;
     int port;
     String filename,outfilename;
     Server1(String ipaddr,int portaddr,String fnname, String outfnname)
       {
       try
       {
              filename=fnname;
              outfilename = outfnname;
               ip=InetAddress.getByName(ipaddr);
        }
       catch(Exception e)
       {
               System.out.println(e);
       }
       System.out.println( "Server starting" );
       port=portaddr;
       t=new Thread(this);
       t.start();
       }

public void run()
  {
        Socket sock = new Socket(ip, port);
        File fp1;
         if(port==81)
        {
              fp1 = new File(filename);
        }
         else if(port==83)
        {
                fp1 = new File(filename);
        }
         else if(port==85)
        {
                fp1 = new File(filename);
        }
        else
        {
                fp1 = new File("temp.txt");
        }

        FileInputStream fins = new FileInputStream(fp1);
          int bsize = 1024000;
          byte[] bdata = new byte[bsize];
          DataOutputStream o = new DataOutputStream(sock.getOutputStream());
          long fsize = fp1.length();
          int n=0;
          while (fsize>0)
        {
            n = fins.read(bdata,0,bsize); // read 1mb from file
            o.write(bdata,0,n); // write it to the socket
            fsize-=n;
        }
         o.close();
        System.out.println("File Sent"); sock.close();
     }
catch(Exception e)
    {
        System.out.println(e);
    }
```

```
        ServerSocket server2 = new ServerSocket( (port+1));
        System.out.println( "Server Waiting...\n" );
         Socket socket = server2.accept();
         if(port==81)
              System.out.println( "Response from Client1 received. " );
         else if(port==83)
              System.out.println( "Response from Client2 received. " );
         else if(port==85)
              System.out.println( "Response from Client3 received. " );
         else
              System.out.println( " " );
         File fp;
          if(port==81)
                {
                 fp = new File(outfilename);
                 }
          else if(port==83)
                {
                  fp = new File(outfilename);
                  }
         else if(port==85)
               {
                  fp = new File(outfilename);
                }
         else
                {
                 fp = new File("e:\\temp.txt");
               }
FileOutputStream fout = new FileOutputStream(fp);

DataInputStream i = new DataInputStream(socket.getInputStream());

int n=0;
long totalbytes = 0;
int bsize = 1024000;
byte[] bdata = new byte[bsize];
System.out.println("Receiving file..");
while ((n = i.read(bdata,0,bsize)) > 0)
```

```
 totalbytes += n;
 }
 fout.close();server2.close();
  }
 catch(Exception ioex)
 {
 System.out.println("Error occurred while trying to service request.");
 System.out.println("Server stopped.");
 ioex.printStackTrace();
 System.exit(0);
 }
 System.out.println("Files Received. File Closed");
}
}


class SS
{
	SS(int ch)
	{
	try
	{

	if(ch>=2)
	{	FileWriter FW1 = new FileWriter("two_temp.txt",false);
		BufferedReader br = new BufferedReader(new
FileReader("two_out1.txt") );
		BufferedReader br1 = new BufferedReader(new
FileReader("two_out2.txt") );
		BufferedReader br2 = new BufferedReader(new
FileReader("two_out3.txt") );
		StringTokenizer stD;
		String Ddata;
		while ((Ddata =br.readLine())!=null)
		{
			stD = new StringTokenizer(Ddata,"	");
			while(stD.hasMoreTokens())
			{
            }
            while ((Ddata =br1.readLine())!=null)
            {
                  stD = new StringTokenizer(Ddata,"     ");
                  while(stD.hasMoreTokens())
                  {
                        FW1.write(stD.nextToken()+"\r\n");
                  }
            }
            while ((Ddata =br2.readLine())!=null)
            {
                  stD = new StringTokenizer(Ddata,"     ");
                  while(stD.hasMoreTokens())
                  {
                        FW1.write(stD.nextToken()+"\r\n");
                  }
            }
            FW1.close();
      }

      if(ch>=3)
      {
            BufferedReader br = new BufferedReader(new
FileReader("three_out1.txt") );
            BufferedReader br1 = new BufferedReader(new
FileReader("three_out2.txt") );
            BufferedReader br2 = new BufferedReader(new
FileReader("three_out3.txt") );
            StringTokenizer stD;
            FileWriter ffw = new FileWriter("three_temp.txt",false);
            String Ddata;
            while ((Ddata =br.readLine())!=null)
            {
                  stD = new StringTokenizer(Ddata,"     ");

                  while(stD.hasMoreTokens())
                  {
                        ffw.write(stD.nextToken()+"\r\n");
```

```
		while ((Ddata =br1.readLine())!=null)
		{
			stD = new StringTokenizer(Ddata,"	");
			while(stD.hasMoreTokens())
			{
				ffw.write(stD.nextToken()+"\r\n");

			}

		}

		while ((Ddata =br2.readLine())!=null)
		{
			stD = new StringTokenizer(Ddata,"	");
			while(stD.hasMoreTokens())
			{
				ffw.write(stD.nextToken()+"\r\n");
			}

		}
		ffw.close();
	}

	if(ch>=4)
	{
		BufferedReader br = new BufferedReader(new
FileReader("four_out1.txt") );
		BufferedReader br1 = new BufferedReader(new
FileReader("four_out2.txt") );
		BufferedReader br2 = new BufferedReader(new
FileReader("four_out3.txt") );
		StringTokenizer stD;

		FileWriter ffw = new FileWriter("four_temp.txt",false);
		String Ddata;
		while ((Ddata =br.readLine())!=null)
		{
```

```
                while(stD.hasMoreTokens())
                {
                        ffw.write(stD.nextToken()+"\r\n");
                }
        }

        while ((Ddata =br1.readLine())!=null)
        {
                stD = new StringTokenizer(Ddata,"    ");
                while(stD.hasMoreTokens())
                {
                        ffw.write(stD.nextToken()+"\r\n");

                }

        }

        while ((Ddata =br2.readLine())!=null)
        {
                stD = new StringTokenizer(Ddata,"    ");
                while(stD.hasMoreTokens())
                {
                        ffw.write(stD.nextToken()+"\r\n");
                }

        }
        ffw.close();
    }

    if(ch==5)
    {
        BufferedReader br = new BufferedReader(new
FileReader("four_out1.txt") );
        BufferedReader br1 = new BufferedReader(new
FileReader("four_out2.txt") );
        BufferedReader br2 = new BufferedReader(new
FileReader("four_out3.txt") );
        StringTokenizer stD;
        String Ddata;
        while ((Ddata =br.readLine())!=null)
        {
                stD = new StringTokenizer(Ddata,"    ");

                while(stD.hasMoreTokens())
                {
                        ffw.write(stD.nextToken()+"\r\n");
                }
        }

        while ((Ddata =br1.readLine())!=null)
        {
                stD = new StringTokenizer(Ddata,"    ");
                while(stD.hasMoreTokens())
                {
                        ffw.write(stD.nextToken()+"\r\n");

                }

        }

        while ((Ddata =br2.readLine())!=null)
        {
                stD = new StringTokenizer(Ddata,"    ");
                while(stD.hasMoreTokens())

                {
                        ffw.write(stD.nextToken()+"\r\n");
                }

        }
        ffw.close();

}
}
```

```
        }
}

class server2
{
public server2() throws UnknownHostException
{
        try
        {
                Server11 s1=new Server11("localhost",1021);
                Server11 s2=new Server11("localhost",1022);
                Server11 s3=new Server11("localhost",1023);
        }
catch(Exception e)
{}

}
}

class Server11 implements Runnable
{

Thread t;
InetAddress ip;
int port;
String str1;
Server11(String ipaddr,int portaddr)
{
try{

 ip=InetAddress.getByName(ipaddr);
 }
 catch(Exception e){
 System.out.println(e);
 }

 port=portaddr;
 t=new Thread(this);
```

```
}

public void run()
{
 try
{
       Socket sock = new Socket( ip, port );
       BufferedReader br1=new BufferedReader(new
InputStreamReader(sock.getInputStream()));
       String str1=br1.readLine();
       FileWriter fw = new FileWriter("cpuload.txt",true);
       System.out.println("Client   :"+str1);
       fw.write(""+str1+"\r\n");
       fw.close();
  }

  catch(Exception ee)
        {
        }

FileSplit fsp = new  FileSplit();



}

}
```

SENDING CPU TO SERVER:

```
class Clientcpu
{

public Clientcpu() throws Exception
{
try {
            String arg;
            ServerSocket ser2 = new ServerSocket(1022);
            Socket plug;
            plug= ser2.accept();
            PrintWriter PW=new PrintWriter(plug.getOutputStream(),true);
            System.out.print("Sending CPU load to server...");
            sysInfo cpuusage=new sysInfo();
           arg=String.valueOf(cpuusage.getCPUUsage());
            PW.println(""+plug.getLocalAddress()+":"+Integer.parseInt(arg));
            Thread.sleep(20000);
      }
catch(IOException ioex)
{
         System.out.println("Error occurred while trying to service request.");
         System.out.println("Server stopped.");
         ioex.printStackTrace();
         System.exit(0);
}
}
}
```

NETWORKING ONE ITEM SET

```
class OneItemset_networking
{

OneItemset_networking(String sernam)
{
try
{
```

```
Socket socket = server2.accept();

System.out.println( "Response from server received." );
try
{
        File fp = new File("file1.txt");
        FileOutputStream fout = new FileOutputStream(fp);

        DataInputStream i = new DataInputStream(socket.getInputStream());

        int n=0;
        long totalbytes = 0;
        int bsize = 1024000;
        byte[] bdata = new byte[bsize];
        System.out.println("Reading file...");
        while ((n = i.read(bdata,0,bsize)) > 0)
        {
        fout.write(bdata,0,n); // write it to the socket's stream
        totalbytes += n;
        }
        System.out.println("File Read Completed. ");
        fout.close();
        System.out.println("File Closed");
}
catch(IOException ioex)
{
        System.out.println("Error occurred while trying to service request.");
        System.out.println("Server stopped.");
        ioex.printStackTrace();
        System.exit(0);
}
server2.close();socket.close();
try
{
         oneitemset ob=new oneitemset();
         Socket sock = new Socket( sernam, 84 );
         File fp1 = new File("one.txt");
         FileInputStream fins = new FileInputStream(fp1);
        DataOutputStream o = new DataOutputStream(sock.getOutputStream());

        long fsize = fp1.length();
        int n=0;
                while (fsize>0)
                 {
                  n = fins.read(bdata,0,bsize); // read 1mb from file
                  o.write(bdata,0,n); // write it to the socket
                  fsize-=n;
                 }
        o.close();sock.close();
        System.out.println("Process completed.File Sent.");
 }
 catch(Exception e)
 {
            System.out.println(e);
 }


}
catch(Exception ece)
{
       System.out.println("Error " +ece);
}
}
}
```

ONE ITEM SET

```
class oneitemset
{
            oneitemset()
      {
            try
            {
            int count[]=new int[8000];
            double TH_value[]=new double[8000];

            String file,data,source;
      file="file1.txt";
      BufferedReader LINE;
      LINE=new BufferedReader(new FileReader(file));

      System.out.println("Producing C1...");
      FileWriter fwr = new FileWriter("one.txt",false);
      for(int c=0;c<8000;c++)
      {
      count[c]=0;
      TH_value[c]=0.0;
      }
      boolean flag;
   while((data=LINE.readLine()) !=null)
                  {
                  sub++;

                  flag=false;
                              st = new StringTokenizer(data," ");
                              while(st.hasMoreTokens())
                              {

data1=Integer.parseInt(st.nextToken());
                                   if (data2==data1)
                                   {
                                   }
                                   else
                                   {
                                   count[data1]=count[data1]+1;
                                   data2=data1;
                                   }
                              }


                  }

               for(int d=0;d<8000;d++)
               {
               if (count[d]!=0)
```

```
                    }
                    }

                     fwr.close();
            }
            catch(Exception e)
            {
            }

        }
}
```

FINAL CLIENT:

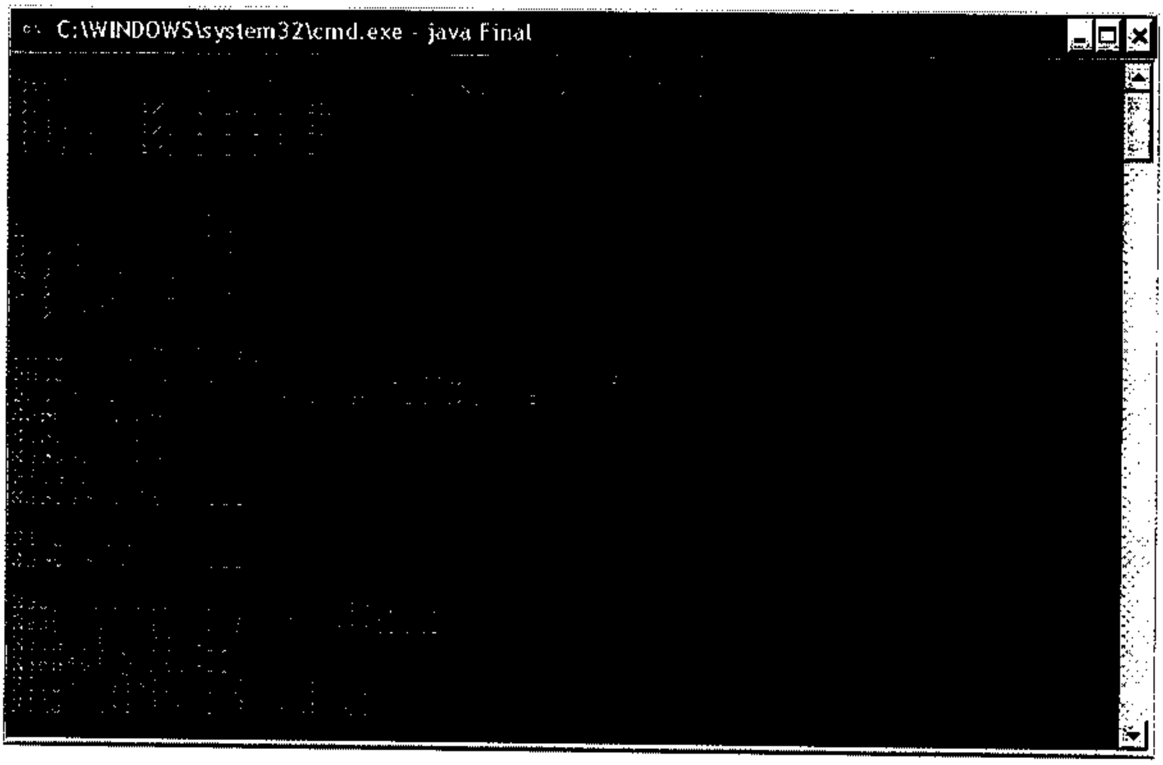
```
class final_client
{
        public static void main(String args[])
        {
                try
                {
                Clientcpu ccpu = new Clientcpu();

              do
            {
      String sernam="localhost";
                Socket sock = new Socket(sernam,92);
                BufferedReader br1=new BufferedReader(new
InputStreamReader(sock.getInputStream()));
                String str1=br1.readLine();
                int ch = Integer.parseInt(str1);
                System.out.println("received"+ch);
                switch(ch)
                {
                        case 1:
                                OneItemset_networking one1 = new
OneItemset_networking(sernam);
                                System.out.println("Client waiting...");
```

```
                    case 2:
                              OneItemset_networking one2 = new
OneItemset_networking(sernam);
                              TwoItemset_networking two2 = new
TwoItemset_networking(sernam);
                              System.out.println("Client waiting...");
                              break;

                    case 3:
                              OneItemset_networking one3 = new
OneItemset_networking(sernam);
                              TwoItemset_networking two3 = new
TwoItemset_networking(sernam);
                              ThreeItemset_networking three3 = new
ThreeItemset_networking(sernam);
                              System.out.println("Client waiting...");
                              break;
                    case 4:
                              OneItemset_networking one4 = new
OneItemset_networking(sernam);
                              TwoItemset_networking two4 = new
TwoItemset_networking(sernam);
                              ThreeItemset_networking three4 = new
ThreeItemset_networking(sernam);
                              FourItemset_networking four4 = new
FourItemset_networking(sernam);
                              System.out.println("Client waiting...");
                              break;
                    case 5:
                              OneItemset_networking one5 = new
OneItemset_networking(sernam);
                              TwoItemset_networking two5 = new
TwoItemset_networking(sernam);
                              ThreeItemset_networking three5 = new
ThreeItemset_networking(sernam);
                              FourItemset_networking four5 = new
FourItemset_networking(sernam);
```

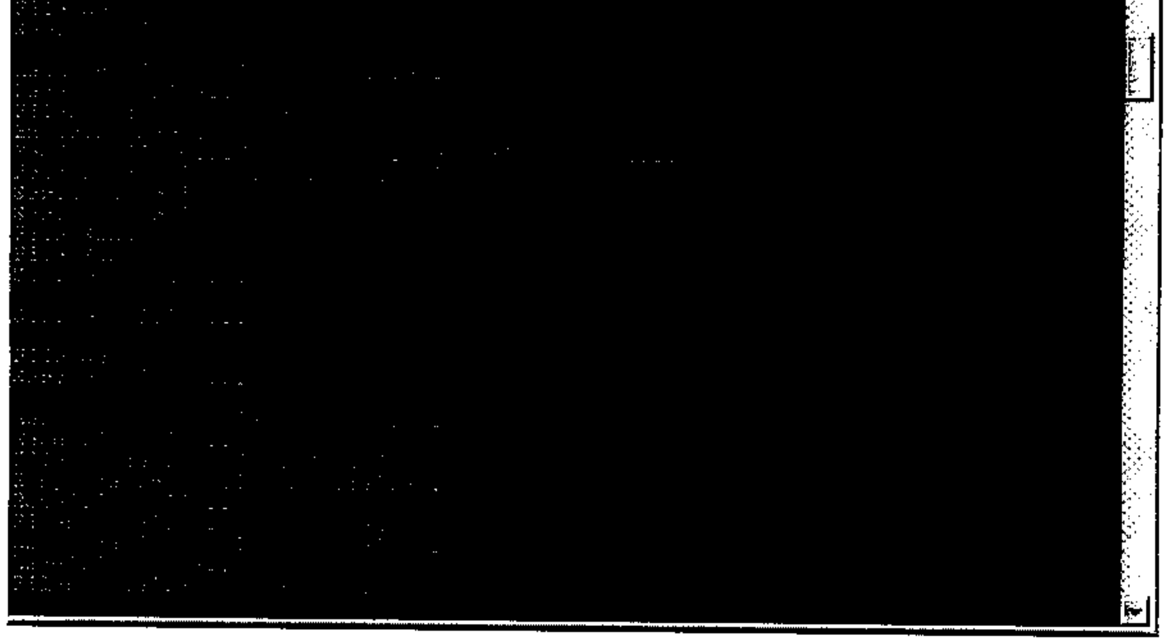
```
                    System.out.println("Client waiting...");
                    break;
               case 6:
                    System.out.println
                    System.exit(0);
                    break;

          }
        sock.close();
          }while(true);
     }

     catch(Exception e)
     {
          System.out.println("error : "+e);
     }
     }

}
```

Appendix 2-SAMPLE SCREENSHOTS

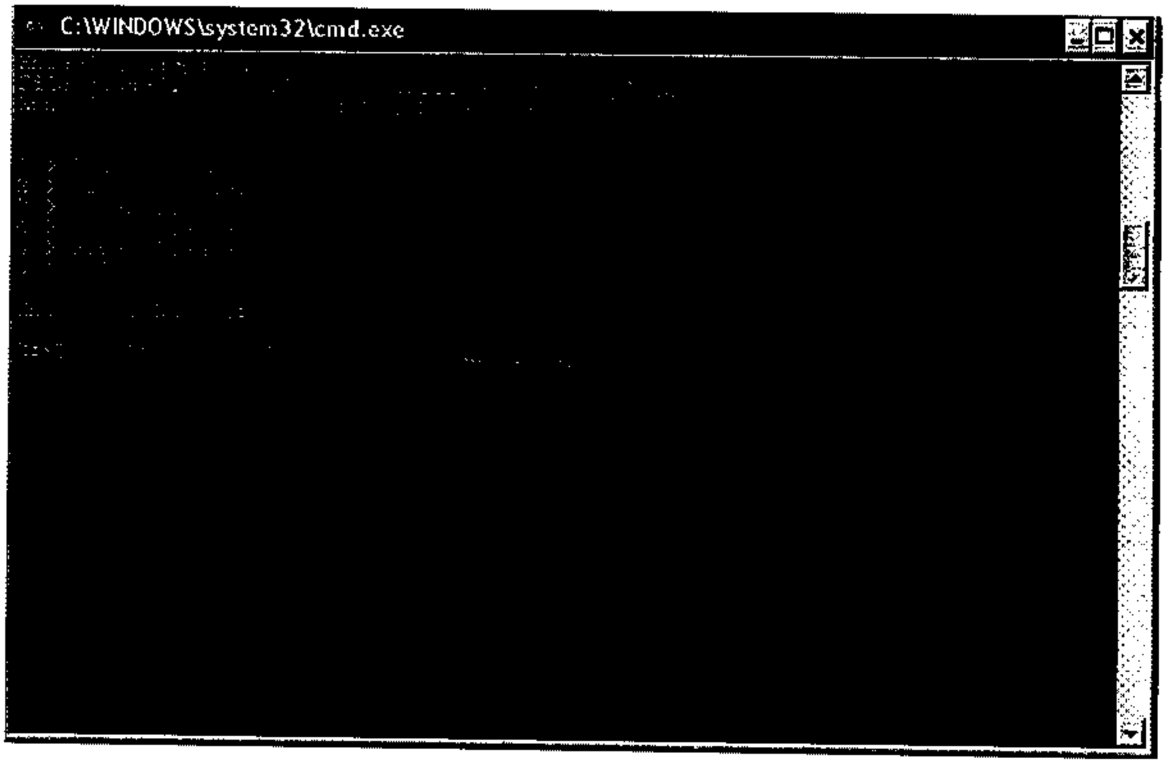
SERVER SIDE



```
[illegible]
```



```
[illegible]
```



C:\WINDOWS\system32\cmd.exe

```
[illegible]
```

C:\WINDOWS\system32\cmd.exe

```
[illegible]
```

REFERENCES

10.1.REFERENCES

[1] Agent Working Group. *Agent Technology Green Paper*. OMG Document ec/99-12-02.Version 0.9. 24 December 1999.

[2] R. Agrawal, T. Imielinski, A. Swamy, "Mining association rules between sets of items in large databases". *ACM SIGMOD Conf.* 1993

[3] Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.,.and Verkamo, I. 1996. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining, 3rd* ed., 307–328, AAAI Press

[4] Agrawal, R., and Psaila, G. 1995. Active Data Mining. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), 3–8. AAAI Press

[5] Apte, C., and Hong, S. J. 1996. Predicting Equity Returns from Securities Data with Minimal Rule Generation. In *Advances in Knowledge Discovery and Data Mining,2nd* ed. , 514–560. AAAI Press

[6] Chan P, Fan W, Prodromidis A, and Stolfo s, "Distributed data mining in credit card fraud detection,", *IEEE Intelligent Systems*, 14(6):67-74, 1999.

[7] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, Michael Y. Zhu ,Purdue University.",Tools for privacy preserving Data mining." In proceedings of the *ACM SIGKDD Explorations*, v.4 n.2, p.28-34, December 2002

[8] Kargupta H,IILkar Hamzoaglu, Brian Stafford., Scalable, "Distributed data mining using an agent based architecture." In the proceedings of *KDD'97.*

[9] Kargupta, H, Park, Hershberger, Johnson ,E., "Collective Data mining: A New perspective toward Distributed Data mining" . *In: Advances in*

[10] Langley, P., and Simon, H. A. 1995. Applications of Machine Learning and Rule Induction. Communications of the ACM 38:55–64.

[11] Matthias Klusch, Stefano Lodi and Gianulco Moro.,"Agent based Distributed Data mining: The KDEC Scheme." (2003), http://citeseer.ist.psu.edu , 2004.

[12] Parthasarathy S, "Towards Network-Aware Data Mining", 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops. *IEEE Computer Society,* p. 30157b

[13] Rakesh Agrawal and Ramakrishnan Srikant., "Privacy-preserving data mining". In Proceedings of the *1997 ACM SIGMOD Conference on Management of Data,* Dallas, TX, May 14-19 2000. ACM.

[14] Salvatore Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Dave W. Fan, et al. "JAM: Java Agents for Meta-Learning over Distributed Databases." In *International Conference on Knowledge Discovery and Data Mining* -1997(KDD-97)

[15] M. Shaw , D. Garlan. *Software architecture: perspectives on an emerging discipline*, 3rd edition, Prentice Hall, New Jersey, 1996.

[16] http://portal.acm.org/, 2004.

[17] www.cs.bme.hu/~bodon/en/apriori , 2004

[18] www.csc.liv.ac.uk/~frans/ KDD ,2004

[19] www.agentland.com , 2005

[20] www.dcs.elf.stuba.sk/emg/kdd.htm, 2004

[21] www.kdnuggets.com, 2005