Abstract — Compute Server is a generalized computing engine which can consist of tens, hundreds, or even thousands of machines on a network, all performing computational tasks [1]. This paper presents ways to integrate Machine Learning with Compute Server in order to build intelligent servers which use machine learning techniques such as Prediction, Clustering, Association rule and Event notification. Built-in intelligence makes server administration automated and leads to better performance. Another addition is that instead of having fixed pre-defined tasks which get dynamically loaded and processed by the slave machines, where user is only able to send the input parameters to the Compute Server; the system developed is a solution using which the user will be able to send both, the input parameters as well as the logic to execute on the parameters. Data dependency analysis can lead to faster computations and data analytics will enable to draw inferences and discover patterns for machine learning. Learning mechanisms will make the server more adaptive to the run-time environment and user-defined tasks will help make the power of Compute Server available to anyone.

Keywords — Distributed Computing, Parallel Processing, Machine Learning, Data Mining, Java.

I. INTRODUCTION

There are many scenarios where organizations need to process huge amount of data within the limited time constraints. For getting such work done, specialized high performance machine is required or the job can be out-sourced. In some other case, the data processing job may require high computations, which many a times, the client may not be able to carry out due to the low system configurations. Such jobs can now be carried out with the use of specialized services provided by certain websites. However, another issue here is that with the increase in the number of clients, the site may get over loaded and its response may become low.

Compute Server will prove to be an effective solution to the above mentioned problems. The use of Compute Server will be beneficial to the users in various ways. Companies often have accounting jobs at the end of every month or during the financial year end. Or users may not be able to compute DFT or any such task due to low system configuration. Compute server will be able to perform such tasks efficiently and relieve the user from the pressure of such jobs involving high computations.

Compute Server has been designed to apply the concepts of Distributed Computing for processing the client’s request. The system consists of a Master machine that will manage the entire network and multiple workers seeking some work. The Dynamic Class Loading feature of Advanced RMI is used to allocate jobs to the Workers. The benefit of this feature is that all the workers are generic and not hard coded to perform any specific task. This allows programming them on the fly. Moreover, all machines including the Master don’t require any special software configuration. This feature made it feasible to work upon the development of Compute Server. And especially, the real world application of this system will be a successful one, and hence proved the main motivation behind developing the Compute Server [2].

Talking about making the Compute Server “smart”; having a system that does the job as instructed is not the best way to do something. The capability of adapting to environment and learning something from it would be the best performance provided by the system.

The efforts were to make the Compute Server intelligent in order to tackle concerns like: how to achieve adaptive fault tolerance, how to monitor unusual activities using statistics,
how to minimize computation without losing its real value or how to find some common things in some notable events.

II. TECHNOLOGIES USED

A. Distributed Computing

Distributed Computing is a collection of individual computers that appears to its user as a single coherent system.

This definition has two aspects. The first one deals with the hardware: the machines are autonomous. And the second one deals with the software: the users think that they are dealing with the same system [3].

This system has been developed using Java technology, so that the server is not dependent on any specific hardware or software platform. End user is not aware that he is dealing with a group of machines as he never knows how the server is going to process his request, and thus the server appears as a single coherent system to the user.

1) Hardware configuration - Heterogeneous Multicomputer Systems: Here each CPU has a direct connection to its own local memory. The computers that form part of the system may vary widely with respect to, for example, processor type, memory sizes, I/O bandwidths and so on. In fact some of the computers may be high performance parallel systems, such as multiprocessors or homogeneous multicomputer [4].

Since the system doesn’t use any specific hardware, all computers forming the compute server may have varying hardware configuration. Compute Server uses Heterogeneous Multicomputer Systems Hardware configuration.

2) Software Configuration - Middleware: Middleware provides the scalability and openness of the network operating system, as well as the transparency and the related ease of use of distributed operating systems. This can be achieved by an additional layer of software that is used in network operating systems to hide the heterogeneity of the collection of underlying platforms but also to improve the distribution transparency [5].

B. Parallel Processing

Parallel Processing involves multiple processes which are active simultaneously and solving given problem, generally on multiple processors.

For example, one unit compiling file1.c and another compiling file2.c will not be considered as parallel processing, because these two are absolutely independent tasks according to the definition of C language. But, if one unit is compiling a statement from file1.c and other is compiling the next statement from the same file, they need to share variable declaration and scoping information properly between them. Hence they are solving a single problem and not two separate problems. Therefore, it can be considered as Parallel Processing [6].

1) Loop Splitting: Loop splitting involves splitting a single loop into a multiple separate loops. Consider the following example –

\[
\begin{align*}
\text{DO i} \\
a[i] &= b[i] + c[i] \\
c[i] &= a[i-1] \\
\end{align*}
\]

Note that a[i-1] is referring to an earlier index and, therefore, to the value computed by the a=b+c formula, in the previous iteration. The second statement does not require the old value of array a at all. Therefore, we can consider rewriting the loop as follows –

\[
\begin{align*}
\text{DO i} \\
a[i] &= b[i] + c[i] \\
\end{align*}
\]

\[
\begin{align*}
\text{DO i} \\
c[i] &= a[i-1] \\
\end{align*}
\]

This produces the same effect as the code above, for sequential and parallel executions.

Now consider the code –

\[
\begin{align*}
\text{DO i} \\
a[i] &= b[i] + c[i] \\
c[i] &= a[i+1] \\
\end{align*}
\]

The values referred to in the second statement are the values before the update as per statement 1. There is a combination of ideas here – copy and split. This transformation can be done in multiple ways – we can copy either c or a. Here is one solution -

\[
\begin{align*}
\text{DO i} \\
x[i] &= c[i] \\
c[i] &= a[i+1] \\
\end{align*}
\]

\[
\begin{align*}
\text{DO i} \\
a[i] &= b[i] + x[i] \\
\end{align*}
\]

2) Loop Interchange: Loop Interchange is a technique which is concerned with the parallelisation of the nested loops. Let’s assume that I is the counter variable for the outer loop and J is the counter variable for the inner loop, now if after interchanging the values of I and J, and the output is same and
if this interchange provides the opportunity to parallelise the nested loop, then the Loop Interchange can be done.

C. Machine Learning and Data Mining

Machine learning addresses the question of how to build computer programs that improve their performance at some task through experience. Machine Learning algorithms have proven to be of great practical value in a variety of application domains. They are especially useful in 1) data mining problems 2) poorly understood domains 3) domains where the program must dynamically adapt to changing conditions. Machine learning draws on ideas from a diverse set of disciplines, including artificial intelligence, probability and statistics, computational complexity, information theory, psychology and neurobiology, control theory, and philosophy [8].

Data Mining is the task for discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses or other information repositories. It is an interdisciplinary field, drawing from areas such as database systems, data warehousing, statistics, machine learning, data visualization, information retrieval, and high performance computing [9].

These two terms are commonly confused, as they often employ the same methods and overlap significantly. They can be roughly defined as follows:

- Machine learning focuses on prediction, based on known properties learned from the training data.
- Data mining (which is the analysis step of Knowledge Discovery in Databases) focuses on the discovery of (previously) unknown properties on the data.

The two areas overlap in many ways: data mining uses many machine learning methods, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a pre-processing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in Knowledge Discovery and Data Mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by supervised methods, while in a typical KDD task; supervised methods cannot be used due to the unavailability of training data [10].

We will be using following Machine Learning or Data Mining techniques-

1) Classification: Classification maps data into pre-defined groups or classes. It is often referred to as the supervised learning because the classes are determined before examining the data [11].

2) Prediction: Prediction is also a supervised learning technique like classification, whereas classification predicts categorical (discrete, unordered) labels, prediction models continuous-valued functions. That is, it is used to predict missing or unavailable numerical data values rather than class labels. The term prediction may refer to both numeric prediction and class label prediction [12].

3) Clustering: Clustering is similar to classification except that the groups are not pre-defined but rather defined by the data alone. Clustering is alternatively referred to as unsupervised learning or segmentation. The clustering is usually accomplished by determining the similarity among the data on pre-defined attributes. The most similar data are grouped into cluster [13].

4) Association: Association refers to the data mining task of uncovering relationships among the data. The best example of this type of application is to determine association rules. An association rule is a model that identifies specific types of data associations [14].

5) Time Series Analysis: With time series analysis, the value of an attribute is examined as it varies over time. The values usually are obtained as evenly spaced time points (daily, weekly, hourly, etc.). There are three basic functions performed in time series analysis. In one case, distance measures are used to determine the similarity between different time series. In second case, the structure of the line is examined to determine its behaviour. The third application would be to use the historical time series plot to predict future values [15].

6) Sequence Discovery: Sequence Discovery is used to determine sequential patterns in data. These patterns are based on a time sequence of actions. These patterns are similar to associations in that data are found to be related, but the relationship is based on the time [16].

D. Java

The Java Programming Language is a general-purpose, concurrent, strongly typed, class-based object-oriented language. It is normally compiled to the bytecode instruction set and binary format defined in the Java Virtual Machine Specification [17].

1) RMI - Giving Networking an Object Oriented view: There are many ways to design applications using a set of objects distributed over a network. Most of these applications are called distributed applications. The idea is that system processing is not performed at one single computing resource
but is instead performed by an orchestration of machines across a network. A well designed distributed system can yield the following highly desirable benefits.

i) Fault–tolerance: if some machine in the network fails, the system as a whole still continues to function (maybe at a reduced capacity).

ii) Scalability: more client workload can be handled by simply adding computing resources at the server location.

iii) High throughput: achieved by combining the processing resources of many machines within a network working in parallel.

It is possible to design an RMI-based system that exhibits all of the above properties. Good distributed system design demands a lot of efforts, time and experience [18].

• Dynamic Class Loading In RMI

One of the most significant capabilities of the Java platform is the ability to dynamically download Java software from any Uniform Resource Locator (URL) to a Virtual Machine (VM) running in a separate process, usually on a different physical system. The result is that a remote system can run a program, for example an applet, which has never been installed on its disk [19].

III. ARCHITECTURE

Before Java, Advanced RMI, most Compute Servers were built using dedicated, massively parallel processing hardware (Cray Computers, for example) and software supporting dedicated tasks had to be created and installed. With Advanced RMI, the network becomes an interconnection between processing resources (individual machines) and the Java VM becomes the generic processor that can be programmed on the fly by loading the Java objects from Remote VMs. The real benefit of using Advanced RMI in this scenario is its ability to store behaviour with the data. This means that the tasks performed by individual machines do not have to be fixed in advance [20].

Compute Server’s architecture consists of two modules: Master module & Worker module.

A. Master Module

This module acts as an entry point and controller for the compute server. Main sub-modules of the Master module are: Request Handler, Work Object Pool, Worker Handler, Work Scheduler, and Logger.

Whenever any request comes in, it is handed over to the Request Handler, it then puts that request into Work Object Pool, and notifies the Work Scheduler, Master then notifies the respective Worker Handler, Worker handler then picks up the job from Work Object Pool and allocates it to the respective Worker.

1) Request Handler: The main job of the Request Handler is to instantiate the respective Work Object based on the request type. And then add that work object into Work Object Pool. It then keeps waiting till Master notifies him about completion of the job. Then, it fetches that job from the Work Object Pool and retrieves the Output Parameters and sends them back to the client.

2) Work Object Pool: This pool consists of work queues. Basically there are three queues. First queue stores the work objects which are ready to be allocated to the worker but not yet allocated i.e. these are ‘waiting-in-queue’ jobs. This is the queue from which Worker Handler picks up the new work. The second queue stores the work objects which are allocated to a particular worker but not yet returned by them i.e. these are ‘work-in-progress’ jobs. This queue is used to re-allocate the job to another worker when any worker doesn’t return allocated job in expected time. The third queue stores the work objects that have completed processing but the results have not been sent back to the client. This is the queue from which Request Handler picks up the processed job.

3) Worker Handler: For every worker which is waiting for work has a respective Worker handler. Worker Handler waits for some user defined time and if it is not allocated a job by Master, it returns to the worker. In case it has been allocated a job, then it fetches the job from the Work Object Pool and sends it to the respective worker. The main aim of reconnecting to Master after some time is a way to make the network self-configurable.

4) Work Scheduler: This module stores Worker machine’s details and its resource utilization parameters. Worker machine’s details include: UUID, OS name, OS Architecture, OS Version, Available Processors, Total Physical Memory, Total Swap space, Hostname, IP Address. Resource utilization parameters include: System Load Average (SLA), Free Physical Memory (FPM), and Free Swap Space (FSS). The main responsibility of this module is to sort and select the best worker such that worker with minimum average SLA and maximum average FPM and FSS, should be considered as the best worker to get the job done.

5) Logger: The main task of this module is to log the important events with timestamp to the log file.

B. Worker Module

This module forms a Generic Worker which is able to dynamically load class files from the Class Server and use the input parameters provided by the Master to execute the allocated job. This module has another sub-module named Machine Monitor.
1) Machine Monitor: This module constantly monitors the Resource utilization parameters of worker machine, which includes System Load Average, Free Physical Memory, Free Swap Space. The module records such parameters at a user defined time gap and store in the array. This module stops recording the parameters when the worker is suspended or disconnected from the master machine so as to limit the resource utilization of the worker on the worker machine.

Fig. 1 Compute Server Architecture
IV. SERVERS AND INTELLIGENCE

In order to provide intelligence to the Compute Server, the following machine learning mining techniques have been considered: Prediction, Clustering and Association Rules.

A. Prediction

Prediction in Compute Server will assist the Master machine to deal with fault tolerance in more adaptive and an intelligent way. As an example, assume that there is a task to find roots of a given equation using bisection method and there are many Worker machines which are heterogeneous in nature. The system will maintain the processing time that each Worker took when it was allocated a job to solve an equation using bisection method. Let’s assume that for current request of bisection method, Work Scheduler has scheduled Worker1 and previously it has received the request to solve the bisection method four times and its details are as follows:

| Worker Readings Table (Table I) shows the average SLA, FPM, and FSS values of the Worker1, when previously it was chosen by the Master’s Work Scheduler to process bisection method problem. Hence, by referring to the Coefficients of Multiple Regression Table (Table II), the equation to predict maximum time that Worker1 should take to execute the bisection method can be find out by substituting the current readings of SLA, FPM, FSS in following equation:

\[
\text{Time} = 0.96115166 \times \text{SLA} + 0.00000035 \times \text{FPM} + 58.35845566 \times \text{FSS} - 2506.5012207
\]

So, Master will wait for the time predicted by this linear equation and some more time to adjust network delays and other delays. But after that time period has been elapsed, if Worker 1 is not returning the work then Master will assume that Worker 1 has crashed or it has encountered some error, and then Master will re-allocate the current bisection method request to other Worker.

In such a scenario, how much time a Master should wait before re-allocating the job to other Worker is decided through the experience that the Master gained through his previous processing of same type of job to the same Worker. This is how prediction can be used for Adaptive Fault Tolerance in the Master machine which is specific to every request type and every Worker.

B. Clustering

In Compute Server, Clustering optimizes the performance while minimizing the computation done by the Work Scheduler and ranking the Worker based on their Resource Utilization Measures i.e. based on SLA, FPM, and FSS. Let’s assume that there are three Workers which are connected to the Master machine and are configured to take SLA, FPM, FSS readings every five seconds. Master machine’s Scheduler queue size is set to 300.

Now consider a situation in which there are 300 readings for every Worker in the scheduler queue. Hence to calculate the average SLA – 300 readings, FPM – 300, readings, FSS – 300 readings, for every Worker, Master machine will have to do the summation of the 900 numbers. And for three workers, 900 * 3 = 2700, master’s Work Scheduler will have to add 2700 numbers. So Work Scheduler will have to do much computation to rank the best workers and this process is time consuming.

Now consider another issue, what will be done when 301st reading arrives at the Work Scheduler? Since the system uses a queue to store these readings, the natural behaviour in case of queue after reaching to its maximum capacity will be to discard the element at its head and add the new element at its tail, i.e. to discard the first reading and add new 301st reading at 300th place. Here there is a loss in the aggregate effect of the first reading which will be discarded while adding a new reading.

Now there is a need to answer these two questions: by reducing the computation as well as maintaining the aggregate impact of the all reading, i.e. to reduce the reading count without affecting the average. We can solve this problem using Clustering. Consider following tables:

<table>
<thead>
<tr>
<th>TABLE I Worker Readings</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLA</td>
</tr>
<tr>
<td>9.61</td>
</tr>
<tr>
<td>4.06</td>
</tr>
<tr>
<td>4.06</td>
</tr>
<tr>
<td>1.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II Coefficients of Multiple Regressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Variables</td>
</tr>
<tr>
<td>Constant Term</td>
</tr>
<tr>
<td>SLA</td>
</tr>
<tr>
<td>FPM</td>
</tr>
<tr>
<td>FSS</td>
</tr>
</tbody>
</table>
reduce the computation for Work Scheduler but will also help in maintaining the aggregate impact.

C. Association

The system will keep track of the unusual activities occurring at the Master and also at the Worker machine. These unusual activities will be recorded in the log and the Association rule algorithm will be implemented to find out the resulting association between the various activities present in the log.

At the Worker side, there can be two major possible circumstances where unusual activity may occur. In the first case, the problem may arise due to wrong input provided by the user at runtime which may lead to wrong output generation or may increase the SLA and in turn would reduce the throughput of the system. In the second case, the problem may arise due to some improper system configuration or some fault at the Worker machine.

At the Master side too, there might be situations where the CPU usage might suddenly increase or the response time may become too low. All such activities need to be monitored continuously in order to ensure efficient utilization of the system.

In both the cases, there is a need for mechanism which will monitor such problems and record them. Maintaining a log of such unusual activities is one solution. This log will enable the system to implement the Association rule algorithm and help Compute Server to decide the future course of action on such anomalous activities.

V. USER DEFINED LOGIC, DATA DEPENDENCY ANALYSIS AND DATA ANALYTICS

A. User Defined Logic

As specified in the architecture, all workers in the Compute Server are generic workers. In other words, the Master machine can delegate any job to the Worker machine. Workers are not hard coded to do any particular job.

This ability of delegating the logic to the other machine at runtime can be further extended to allow users to send the logic to the compute server along with the data to be processed by it. This will expose the high computing capability of Compute Server to everyone. In case of pre-defined logic, user will be able to use the power of Compute Server to process the tasks which are already defined. Here user is only able to send the input parameters required for the pre-defined tasks. The problem with this approach is that user will not be able to process the task he wishes to be processed by the Compute Server. This consideration might be useful in the scenario where user is having some tasks which require

Worker Readings table shows twelve records of SLA, FPM, FSS readings. Clustering Results table shows output of the K-means clustering algorithm, when it is applied on the Worker Reading records where the output clusters were four. And the averages of both the tables are same. So twelve records of the Worker Readings table (Table III) can be represented by four records of the Clustering Results Table (Table IV). So making use of clustering technique to reduce the reading count when Work Scheduler’s queue reaches to its capacity, will not only
high computing power and user is not having enough resources to process this task. With the help of user defined logic, user will be able to send both-input parameters as well as the logic to process those input parameters.

To achieve this, the system provides the user with a Java interface which users have to implement. That interface will have general structure of any data processing task, i.e. Input-Process-Output. Basically it has three methods. First method takes the input data and initialize the data members. The second method includes all the processing logic which is executed by the Worker machine. And the third method is used to return the output parameters of the task. To get the user defined input, XML Schema will be provided to the user, where he will put his input data to be processed by his own logic.

B. Data Dependency Analysis

Now there is a class having three methods with user defined logic and input data to be processed by this class. There are three basic types of logic: Sequential Logic, Selection Logic, and Iterative Logic. Almost all programs will involve Iterative Logic, i.e. loops of some type. And much of the time spent by the program is in the loops. Also note that loops are the direct source of the parallelism – one could try executing the different iteration on the different machines. Therefore, one needs to be particularly concerned about the Loop Parallelism. However, as with the simple statements, the question of data dependency arises here also. Different iterations of the loop may not be independent to ensure the correct results under parallel execution [21]. Given the importance of Parallel Processing, the system will perform the Loop Dependency Analysis which will check for the dependency across the iterations in the loop. After analysis if any loop dependency is found, the system will allocate that job as a whole to the worker, otherwise it will implement the concepts of Loop Splitting and Loop Interchange wherever possible, and achieve the performance gains that parallel processing offers to us.

C. Data Analytics

So far, the system has user-defined logic, user-defined input data, and loop dependency analysis to check for the parallel execution of the user’s request. All the things have been done to make Compute Server’s high computing power available to everyone. How about some value added service?

Let’s consider an example of processing the data-Airline Ticket Transactions to build some kind of report whose logic is being provided by the user. So, by the time this report is being prepared, compute server will apply Data Mining techniques which will include Classification, Prediction, Time series Analysis, Clustering, Association Rules, and Sequence Discovery on the data provided by the user. Using these data mining techniques Compute Server will try to draw inferences, make predictions, discover patterns, and find common features between these Airline Ticket Transactions. When the request processing has completed, along with the results of the request, in our example along with the report, the report generated by the mining techniques will also be sent to the user. This will enable user to get some more insights into his own data, which he can use for his business purpose. Compute Server will store report of mining techniques into its database. Next time whenever the user defined request comes with Airline Ticket Transaction data, say for building the report for the next quarter or next year; the system will again mine the new data and try to discover hidden information. And while sending report back to the user, it will compare the earlier reports with current one and again build some new patterns which will be sent to the user and can be used to optimize the processing of similar requests the next time.

VI. EVENT NOTIFICATION FRAMEWORK

Event Notification Framework is a mechanism which will define whom to notify about every event happening in the Compute server. The event can be any activity happening in the Compute Server. Examples for events in the Compute Server can be: New request comes in, work object added to work pool, request successfully processed by the worker, scheduler queue has reached to the maximum capacity, any exception occurred due to improper or invalid input parameter provided by the user, worker crash, master machine crash and so on.

This framework is broadly specified using four levels of events. These four levels are very generic levels and they can be extended to fulfill any server specific requirements.

The first and the lowest level of events are, Events only to be logged. These events might involve logging very general server specific activity. Ex. Time out occurred at the Worker Handler and Worker Handler returned.

The second level events are those which require some programmatic action to be taken. Ex. New request has been received – now call the Request Handler, job has been processed by the worker and it has been added to the result queue – now notify the respective Request Handler to pick up his work object, scheduler queue has reached to its maximum capacity – now call the clustering algorithm to cluster the scheduler queue readings and so on.

The third level is Customer Level Events; these are the events which require notification to be sent to the customer. Ex. Successful completion of user defined task, user defined class is not following the constraints specified, exception occurred while processing the user defined task which was due to invalid arguments provided by the user, etc. For these kinds of events, a SMS and/or E-Mail will be sent to the customer, which will contain details of the event.
The next level of event will be Administrative Level Events. These are the events which are fired by internal functioning of the Compute Server. Ex. Disconnection of the Worker, Worker crash while it was processing the job given by the Master, exception occurred at the Worker while communicating with the Master, Thread synchronization exception occurred at the Master machine, exception occurred at the Master machine, Master machine crash and so on. For these kind of events SMS and / or E-Mail will be sent to the administrator of the Compute Server which will contain details of the event.

**VII. PROPOSED SYSTEM ARCHITECTURE**

The proposed system will have two types of requests, Pre-defined and User-defined. Pre-defined tasks will be executed as they are implemented. Whenever any user-defined task will arrive at compute server, Request Handler will ask the Data Dependency Analyzer to check whether this task can be executed in parallel, otherwise task is allocated as a whole to the Worker. If Data Dependency Analyzer detects no dependencies then request is executed in parallel. By the time request is being processed by the Worker(s), Data Analytics will run different Data Mining algorithms on the data provided by the user to find out hidden patterns in the data and it will be stored in the database. Server Intelligence will mine on the data generated by the server itself to optimize the processing. System will use Event Notification Framework to notify appropriate persons and/or trigger programmatic action whenever any respective event happens.

![Fig. 2 Proposed Compute Server Architecture](image-url)
VIII. CONCLUSION

The inclusion of intelligence in the Compute Server increases the performance of the system considerably. The throughput of the system can be increased and the time required for the computation and CPU usage can be reduced effectively. Moreover, the run-time environment for the user helps making the system generic in nature and the system becomes useful for applications other than just computations. Also, the intelligence introduced in the system will help perform the data mining tasks efficiently. This in turn, will lead to better machine learning and improve the overall performance of the system.

ACKNOWLEDGEMENT

We would like to acknowledge and extend our heartfelt gratitude to the following significant advisors and contributors who have made the completion of this project possible:

First and foremost, we would like to thank Patkar-Varde College for giving us an opportunity to make this research paper a success. Also to the College Library and Computer Laboratory staff for providing us with all the facilities.

Second, I would like to thank Mr. Nilesh Patil to read our paper and to provide valuable advices.

And finally, to our parents for the support and encouragement they provided.

REFERENCES

[7]. M. Sasikumar, Dinesh Shikhare, P. Ravi Prakash, Introduction to Parallel Processing, Prentice-Hall of India, 2006, Pg. 60.
[12]. Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006, Pg. 5.
[15]. Margaret H. Dunham, Data Mining: Introductory and Advanced Topics, Pearson Education, 2005, Pg. 5.
[21]. M. Sasikumar, Dinesh Shikhare, P. Ravi Prakash, Introduction to Parallel Processing, Prentice-Hall of India, 2006, Pg. 52.