CHAPTER 2
REVIEW OF LITERATURE

Data mining is a process of nontrivial extraction of implicit, previously unknown, and potentially useful information (such as knowledge rules, constraints, and regularities) from data in databases [PF, 91]. In fact, the term "knowledge discovery" is more general than the term "data mining." Data mining is usually viewed as a step towards the process of knowledge discovery [HK, 01], although these two terms are considered as synonyms in the computer literature. The entire life cycle of knowledge discovery includes steps such as data cleaning, data integration, data selections, data transformation, data mining, pattern evaluation, and knowledge presentation.

![Figure 2a: Life Cycle of knowledge presentation](image)

Data cleaning is to remove noise and inconsistent data. Data integration is to combine data from multiple data sources, such as a database and data warehouse. Data selection is to retrieve data relevant to the task. Data transformation is to transform data into appropriate forms. Data mining is
to apply intelligent methods to extract data patterns. Pattern evaluation is to identify the truly interesting patterns based on some interestingness measures. Knowledge evaluation is to visualize and present the mined knowledge to the user. There are many data mining techniques, such as association rule mining, classification, clustering, sequential pattern mining, etc.

Since this thesis focuses on parallel and distributed data mining with remote memory utilization, the related work associated with this is summarized.

2.1 Distributed Data Mining

Databases in today's information age are inherently distributed. Organizations that operate in global markets need to perform data mining on distributed data sources (homogeneous / heterogeneous) and require cohesive and integrated knowledge from this data. Such organizational environments are characterized by a physical/geographical separation of users from the data sources. This inherent distribution of data sources and large volumes of data involved inevitably leads to exorbitant communications costs. Therefore, it is evident that traditional data mining model involving the co-location of users, data and computational resources is inadequate when dealing with environments that have the characteristics outlined earlier. The development of data mining along this dimension has led to emergence of DDM [ZaP, 02] [Fu, 01] [Kam, 01] [KKC, 99] [KRS, 01]. The need to address specific issues associated with the application of data mining in distributed
computing environments is the primary objective of distributed data mining. Broadly, data mining environments consist of users, data, hardware and the mining software (this includes both the mining algorithms and any other associated programs). Distributed data mining addresses the impact of distribution of users, software and computational resources on the data mining process. There is general consensus that distributed data mining is the process of mining data that has been partitioned into one or more physically/geographically distributed subsets. Data mining algorithms deal pre-dominantly with simple data formats (typically flat files). There is an increasing amount of focus on mining complex and advanced data types such as object-oriented, spatial and temporal data [HNK, 94] [AnO, 01] [RHS, 01] [HKS, 97]. Another aspect of this growth and evolution of data mining systems is the move from stand-alone systems using centralized and local computational resources towards supporting increasing levels of distribution. As data mining technology matures and moves from a theoretical domain to the practitioner’s arena there is an emerging realization that distribution is very much a factor that needs to be accounted for.

The significant factors, which have led to the emergence of distributed data mining, are as follows:

- The need to mine distributed subsets of data, the integration of which is non-trivial and expensive.
- The performance and scalability bottle necks of data mining.
- Distributed data mining provides a framework for scalability, which allows the splitting up of larger datasets with high
dimensionality into smaller subsets that require less computational resources individually [FrL, 98].

Distributed Data Mining (DDM) is a branch of the field of data mining that offers a framework to mine distributed data, paying careful attention to the distributed data and computing resources. In the DDM literature, one of two assumptions is commonly adopted as to how data is distributed across sites: homogeneously and heterogeneously. Both viewpoints adopt the conceptual viewpoint that the data tables at each site are partitions of a single global table. In the homogeneous case, the global table is horizontally partitioned. The tables at each site are subsets of the global table; they have exactly the same attributes. In the heterogeneous case the table is vertically partitioned, each site contains a collection of columns (sites do not have the same attributes). However, each tuple at each site is assumed to contain a unique identifier to facilitate matching. It is important to stress that the global table viewpoint is strictly conceptual. It is not necessarily assumed that such a table was physically realized and partitioned to form the tables at each site. This distributed environment is considered in this research work where the dataset is in a centralized location but the workstations are distributed globally to perform the mining process.

Andrei L. Turinsky et al presents that distributed data mining is emerging as a fundamental computational problem [AR, 04]. A common approach with distributed data mining is to build separate models at geographically distributed sites and then to combine the models. At the other extreme, all the data can be moved to a central site and a single model
built. There are a variety of intermediate strategies in which some of the data is moved and some of the data is left in place, analyzed locally, and the resulting models are moved and combined. These intermediate cases are coming to be of practical significance with the explosion of fiber and the emergence of high performance networks. In this work, an intermediate case is examined in the context in which high performance networks are present and the cost function represents both computational and communication costs. The communication cost is also a major task considered in this research work because the dataset which is partitioned has to be allocated to the distributed workstations. Hence the communication overhead involved is also considered.

Tao Li et al says Distributed Data Mining (DDM) has been very active and enjoying a growing amount of attention since its inception [TSM, 02]. Current DDM techniques regard the distributed data sets as a single virtual table and assume there exists a global model which could be generated if the data were combined/centralized. This paper proposes a similarity based distributed data mining (SBDDM) framework which explicitly takes the differences among distributed sources into consideration. A new similarity measure is introduced and its effectiveness is then evaluated and validated. Similarity measures between homogeneous datasets are used for deviation detection data quality mining, distributed mining and trend analysis. Also, a new similarity measure is introduced. The new measure is calculated from support counts using a formula inspired by information entropy.
Shonali Krishnaswamy et al say that there is an emerging interest in optimization strategies for distributed data mining in order to improve response time [SSA, 04]. Optimization techniques are operated by first identifying factors that affect the performance in distributed data mining, computing/assigning a "cost" to those factors for alternate scenarios or strategies and then choosing a strategy that involves the least cost. In this paper the use of application run time estimation is proposed as solution to estimating the cost of performing a data mining task in different distributed locations. A priori knowledge of the response time provides a sound basis for optimization strategies, particularly if there are accurate techniques to obtain such knowledge. In this paper a novel rough sets based technique is presented for predicting the run times of applications. Also an experimental validation of the prediction accuracy of this technique is presented for estimating the run times of data mining tasks. There is an emerging focus on efficiency and optimization of response time in distributed data mining (Parthasarathy and Subramonian 2001, Turinsky and Grossman 2000, Krishnaswamy et al. 2000). This can be attributed to two reasons. Firstly, DDM aims to improve the scalability of mining large data sets. Scalability in turn is intrinsically linked with better performance in terms of response times. Secondly, the emergence of several Internet-based data mining service providers is driving the need to improve the efficiency of the distributed data mining process (Krishnaswamy et al. 2002). Response time is an important Quality of Service (QoS) metric in web-based data mining services for both clients and service providers.
In this research work the homogeneous dataset is considered and data is allocated based on the CPU idle time to the distributed workstations. The workstations found on the geographically distributed locations perform the mining process in parallel. The response time is taken into account and optimization strategies are used to obtain an optimized response time. Various factors are considered for the processing of response time.

2.2 Association Rules

Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [AIS, 93]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc.

Sotiris Kotsiantis et al say that Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database [SD, 03]. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is $L_k$, $L_k = \{i_1, i_2, \ldots, i_k\}$, association rules with these itemsets are generated in the following way: the first rule is $\{i_1, i_2, \ldots, i_{k-1}\} \Rightarrow \{i_k\}$, by checking the confidence this rule can be
determined as being interesting or not. Then other rules are generated by deleting the last items in the antecedent and inserting them to the consequent, further the confidences of the new rules are checked to determine their interestingness. That process is iterated until the antecedent becomes empty. Since the second sub-problem is quite straightforward, most of the researches focus on the first sub-problem. The first sub-problem can be further divided into two sub-problems: candidate large itemsets generation process and frequent itemsets generation process. These are known as itemsets whose support exceed the support threshold as large or frequent itemsets and those itemsets that are expected or have the hope to be large or frequent are called candidate itemsets.

In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. Further, the association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several strategies have been proposed to reduce the number of association rules, such as generating only "interesting" rules, generating only "nonredundant" rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength. Hegland [Heg, 03] reviews the most well known algorithm for producing association rules - Apriori and discuss variants for distributed data, inclusion of constraints and data taxonomies. The review ends with an outlook on tools which have the potential to deal with long itemsets and considerably reduce the amount
of (uninteresting) itemsets returned. In this paper, the most recent existing association rule mining techniques are surveyed.

A. Demers et al says that association rule has a set of relationship among a set of objects [DJV*, 01]. Mining association rules may require iterative scanning of large transactions. Efficient mining of association rules in transaction and relational databases are dealt with here.

Cornelia Györödi et al give a comparative study of the algorithms involved in association rules [CRS, 01]. The algorithms such as Apriori, FP Tree Growth Algorithm, and Dyn FP-Growth algorithm are clearly compared and the factors such as scalability etc are clearly specified.

Mafruz Zaman Ashrafi et al talk of association rule mining as an active data mining research area [MDK, 04]. However, most ARM algorithms cater to a centralized environment. In contrast to previous ARM algorithms, ODAM is a distributed algorithm for geographically distributed data sets that reduces communication costs. Modern organizations are geographically distributed. Typically, each site locally stores its ever increasing amount of day-to-day data. Using centralized data mining to discover useful patterns in such organizations' data isn't always feasible because merging data sets from different sites into a centralized site incurs huge network communication costs. Data from these organizations are not only distributed over various locations but also vertically fragmented, making it difficult if not impossible to combine them in a central location.

*Distributed data mining* has thus emerged as an active subarea of data mining research. A significant area of data mining research is association
rule mining. Unfortunately, most ARM algorithms focus on a sequential or centralized environment where no external communication is required. Distributed ARM algorithms, on the other hand, aim to generate rules from different data sets spread over various geographical sites; hence, they require external communications throughout the entire process. DARM algorithms must reduce communication costs so that generating global association rules costs less than combining the participating sites' data sets into a centralized site. However, most DARM algorithms don't have an efficient message optimization technique, so they exchange numerous messages during the mining process. A new distributed algorithm is developed, called Optimized Distributed Association Mining, for geographically distributed data sets. ODAM generates support counts of candidate itemsets quicker than other DARM algorithms and reduces the size of average transactions, data sets, and message exchanges.

2.3 Dynamic Remote Memory Utilization

Load balancing for distributed systems represents mapping or remapping of work to different processors with the intent of assigning each processor with an equal amount of work. The heaviest use of load balancing techniques is found in the domain of distributed systems. However, most of the work is carried out on computational tasks and not in the storage systems area [RN, 95]. Load balancing of data is already more efficient in Distributed File Systems than in standard non distributed file systems. One reason is the use of replication, which allows for “read-only” filesets to be replicated on multiple machines. Requests for files from frequently used
"read-only" filesets are then spread across different machines, preventing any one of them from becoming overburdened with data requests.

IBM Informix talks about the load balancing techniques widely employed in the domain of distributed systems. However, most of their applications work on redistributing the work load between multiple processors to speed up computational tasks and not in the storage systems area. Whenever a system resource, such as a CPU or a particular disk, is occupied by a transaction or query, it is unavailable for processing other requests. Pending requests must wait for the resources to become available before they can complete. When a component is too busy to keep up with all its requests, the overused component becomes a bottleneck in the flow of activity. The higher the percentage of time that the resource is occupied, the longer each operation must wait for its turn.

The following formula is used to estimate the service time for a request based on the overall utilization of the component that services the request. The expected service time includes the time that is spent both waiting for and using the resource in question. Think of service time as that portion of the response time accounted for by a single component within your computer, as the following formula shows: \[ S = \frac{P}{1-U} \]

\( S \) is the expected service time.

\( P \) is the processing time that the operation requires once it obtains the resource.

\( U \) is the utilization for the resource (expressed as a decimal).
This resource-utilization formula is used to estimate the response time for a heavily loaded CPU. However, high utilization for the CPU does not always indicate a performance problem. The CPU performs all calculations that are needed to process transactions. The more transaction-related calculations that it performs within a given period, the higher the throughput will be for that period. As long as transaction throughput is high and seems to remain proportional to CPU utilization, a high CPU utilization indicates that the computer is being used to the fullest advantage.

On the other hand, when CPU utilization is high but transaction throughput does not keep pace, the CPU is either processing transactions inefficiency or it is engaged in activity not directly related to transaction processing. CPU cycles are being diverted to internal housekeeping tasks such as memory management. The following activities can be easily eliminated:

- Large queries that might be better scheduled at an off-peak time
- Unrelated application programs that might be better performed on another computer

If the response time for transactions increases to such an extent that delay becomes unacceptable, the processor might be swamped; the transaction load might be too high for the computer to manage. Slow response time can also indicate that the CPU is processing transactions inefficiently or that CPU cycles are being diverted. When CPU utilization is high, a detailed analysis of the activities that the database server performs
can reveal any source of inefficiency that might be present due to improper configuration.

Remote workstations, whose memories are available for application execution workstations, are found dynamically during the execution. They are called as "memory present workstations". The mechanism is to decide the availability of remote workstations. On memory available workstations, a process is running to monitor the amount of available memory periodically. "netstat-k" command provided by Solaris operating system is used to get memory information from the kernel statistics structure. Each time the process gets the information, the process broadcasts it to all application execution workstations. On application execution workstations, a client process is running and waiting for the information sent from the memory monitoring processes running on memory available workstations. The client process has a memory area which can be shared with application processes, and the received information about the amount of memory at each workstation is written on the shared memory. The application processes can read this information at anytime, to decide the policy of swapping operations. For example, when a memory available node does not have enough memory space, the shortage of memory is detected by application processes, so that another node is chosen as a swapping destination afterward. On the other hand, if some other processes begin their execution on a memory available node which already accepted swapping operations, the swapped out data must be migrated to other memory available nodes to make space on its memory for the new processes.
2.4 Role of Intelligent Agents in Distributed Data Mining

Agents are defined as software or hardware entities that perform some set of tasks on behalf of users with some degree of autonomy [RN, 95]. In order to work for somebody as an assistant, an agent has to include a certain amount of intelligence, which is the ability to choose among various courses of action, plan, communicate, adapt to changes in the environment, and learn from experience. In general, an intelligent agent can be described as consisting of a sensing element that can receive events, a recognizer or classifier that determines which event occurred, a set of logic ranging from hard-coded programs to rule-based inferencing, and a mechanism for taking action [DAA, 95].

Data mining agents seek data and information based on the profile of the user and the instructions she gives. A group of flexible data-mining agents can co-operate to discover knowledge from distributed sources. They are responsible for accessing data and extracting higher-level useful information from the data. A data mining agent specializes in performing some activity in the domain of interest. Agents can work in parallel and share the information they have gathered so far.

Ayse Yasemin SEYDIM's paper on Agents, special types of software applications, has become a very popular paradigm in computing in recent years [Ays, 99]. Some of the reasons for this popularity is their flexibility, modularity and general applicability to a wide range of problems. Recent increase in agent-based applications is also because of the technological developments in distributed computing, robotics and the emergence of
object-oriented programming paradigms. Advances in distributed computing technologies have given rise to use of agents that can model distributed problem solving.

Pericles A. Mitkas et al’s work on Software agent technology has matured enough to produce intelligent agents, which can be used for controlling a large number of concurrent engineering tasks [PAD*, 01]. Multi-agent systems are communities of agents that exchange information and data in the form of messages. The agents’ intelligence can range from rudimentary sensor monitoring and data reporting, to more advanced forms of decision making and autonomous behavior. The behavior and intelligence of each agent in the community can be obtained by performing data mining on available application data and the respected knowledge domain. An Agent Academy a software platform is designed for the creation, and deployment of multiagent systems, which combines the power of knowledge discovery algorithms with the versatility of agents. Using this platform, agents are equipped with a data-driven inference engine, can be dynamically and continuously trained. Three prototype multi-agent systems are developed with Agent Academy.

Agent-based systems belong to the most vibrant and important areas of research and development to have emerged in information technology. Because of the lively extensive spreading of directions in research no publicly accepted solid definitions of agent-based systems and their elements – agents is provided. Hence, in context of this paper some general definitions are used: Software agent is software that acts as an agent for
another as in a relationship of agency. When several agents act they may form a multi-agent system. Intelligent Agent (IA) refers to a software agent that exhibits some form of artificial intelligence. According to Wooldridge intelligent agents are defined as agents, capable of flexible autonomous action to meet their design objectives. They must involve:

- Reactivity: to perceive and respond in a timely fashion to changes occurring in their environment in order to satisfy their design objectives. The agent's goals and/or assumptions that form the basis for a procedure that is currently executed may be affected by a changed environment and a different set of actions may have to be performed.

- Pro-activeness: ability to exhibit goal-directed behavior by taking the initiative, responding to changes in their environment in order to satisfy their design objectives.

- Sociability: capability of interacting with other agents (software and humans) through negotiation and/or cooperation to satisfy their design objectives.

In this research work intelligent agents are used to automate various tasks. During the mining process the mobile intelligent agents keeps monitoring the different workstations in the geographically distributed areas.

2.5 Parallel Algorithms for Distributed Data Mining

Mohammed J. Zaki et al has discussed various parallel algorithms for building decision-tree classifiers on shared memory multi-processor (SMP)
systems are dealt with [MCR, 01]. The proposed algorithms span the gamut of data and task parallelism. The data parallelism is based on attribute scheduling among processors. This basic scheme is extended with task pipelining and dynamic load balancing to yield faster implementations. The task parallel approach uses dynamic subtree partitioning among processors. The performance of these algorithms on two machine configurations: one in which data is too large to fit in memory and must be paged from a local disk as needed and the other in which memory is sufficiently large to cache the whole data. This performance evaluation shows that the construction of a decision-tree classifier can be effectively parallelized on an SMP machine with good speedup.

Mohammed J. Zaki's work in parallel ARM algorithms is categorized as data-parallelism or task-parallelism algorithms [Moh, 00]. In the former, the algorithms partition the data sets among different nodes; in the latter, each site performs the task independently but must access the entire data set. The Count Distribution (CD) algorithm is a simple data-parallelism algorithm. It uses the sequential Apriori algorithm in a parallel environment and assumes data sets are horizontally partitioned among different sites. The CD algorithm’s main advantage is that it doesn’t exchange data tuples between processors it only exchanges the counts. In the first scan, each processor generates its local candidate itemset depending on the items present in its local partition. The algorithm obtains global counts by exchanging local counts with all other processors. The algorithm’s
communication overhead is $O(|C| \cdot n)$ at each phase, where $|C|$ and $n$ are the size of candidate itemsets and number of data sets, respectively.

Writing parallel data mining algorithms are a non-trivial task. The main challenges associated with parallel data mining include

Minimizing I/O

- minimizing synchronization and communication
- effective load balancing
- effective data layout
- deciding on the best search procedure to use
- good data decomposition
- minimizing/avoiding duplication of work

Data Distribution is a task-parallelism-based algorithm that partitions the candidate itemsets among the processors. Each processor is responsible for computing the counts of its locally stored subset of the candidate itemsets for all the transactions in the database. Each processor must scan the portions of the transactions assigned to other processors as well as its locally stored portion of the transactions. It thus suffers from high communication overhead and performs poorly when compared with CD. Candidate Distribution partitions the candidates during iterations, so that each processor can generate disjoint candidates independently. At the same time, it selectively replicates the database so that a processor can generate global counts independently. Candidate Distribution performs worse than CD. The Common Candidate Partitioned Database uses a data-parallel
approach in shared-memory architecture. The algorithm partitions the
database logically into equal-sized chunks. Each processor generates a
disjoint candidate subset, leading to good computational division.

The parallel design space spans three main components: the hardware
platform, the type of parallelism, and the load-balancing strategy. In
distributed-memory architecture, each processor has its own local memory,
which only that processor can access directly. For a processor to access data
in the local memory of another processor, message passing must send a
copy of the desired data elements from one processor to the other. Although
shared memory architecture offers programming simplicity, a common bus’s
finite bandwidth can limit scalability. A distributed memory, message-
passing architecture cures the scalability problem by eliminating the bus,
but at the expense of programming simplicity.

Two dominant approaches for using multiple processors have
emerged: distributed memory (where each processor has a private memory)
and shared memory (where all processors access common memory). Shared-
memory (SMP) architecture has many desirable properties. Each processor
has direct and equal access to all the system’s memory. Parallel programs
are easy to implement on such a system. A different approach to
multiprocessing is to build a system from many units, each containing a
processor and memory. In distributed-memory (DMM) architecture, each
processor has its own local memory, which only that processor can access
directly. For a processor to access data in the local memory of another
processor, message passing must send a copy of the desired data elements
from one processor to the other. Although shared memory architecture offers programming simplicity, a common bus's finite bandwidth can limit scalability. A distributed memory, message-passing architecture cures the scalability problem by eliminating the bus, but at the expense of programming simplicity. A third, very popular, paradigm combines the best of the distributed- and shared-memory approaches. Included in this paradigm are hardware- or software distributed shared-memory systems. These systems distribute the physical memory among the workstations but provide a shared global address space on each processor. The hardware or software ensures cache coherence; so locally cached data always reflects any processor's latest modification [Moh, 00].

![Diagram showing the taxonomy of parallel association-rule-mining (ARM) algorithms.]

Figure 2b: Parallel Association-rule-mining (ARM) algorithms.

The taxonomy shows that each method fits in the design space, organized by load balancing strategy, architecture, and parallelism. For example, Count the word Distribution uses static balancing, distributed memory, and data parallelism.
The general approach is to implement the Apriori algorithm in the most efficient manner possible, utilizing a minimum of hardware and a minimum of time, as well as insuring that utilization of the hardware comparators is near 100% [ZV, 02]. For some parts of the implementation, namely the support calculation, this is an easy task as checking for set equivalence is a simple operation. However, the candidate generation and pruning operations are significantly more complicated as they introduce new data in the system at unpredictable intervals. It is insured that the memories are of minimum size and yet can also be sufficient to store all data necessary. This architecture allows for roughly 560 units on a single device, assuming perfect routing and placement (this is based on place and route of the full systolic array design. Hardware usage is roughly 70 slices per unit with resources for up to 16 2-byte item candidate sets). These units are all connected end-to-end in the form of a linear array, and contain memory locations to temporarily store the candidates whose support is being calculated and to allow for stalling.

Rakesh Agrawal et al in their paper presents three parallel algorithms for mining association rules, an important data mining problem [RJ, 00]. These algorithms have been designed to investigate and understand the performance implications of a spectrum of trade-offs between computation, communication, memory usage, synchronization, and the use of problem-specific information in parallel data mining. Specifically,
1. The focus of the Count Distribution algorithm is on minimizing communication. It does so even at the expense of carrying out redundant duplicate computations in parallel.

2. The Data Distribution algorithm attempts to utilize the aggregate main memory of the system more effectively. It is a communication-happy algorithm that requires workstations to broadcast their local data to all other workstations.

3. The Candidate Distribution algorithm exploits the semantics of the particular problem on hand to reduce synchronization between the processors and has load balancing built into it.

These algorithms have been implemented on an IBM POWER parallel System SP2, a shared-nothing machine. These results, besides being of interest in themselves, have larger applicability. The performance evaluation can provide guidance to the designers of parallel algorithms for other data mining tasks.

As parallelization techniques increase the speed, in this research work the parallel data mining algorithms like Count distribution and Hash Partitioned Apriori are used.

2.6 Parallel Data Mining Vs Distributed Data Mining

- Parallel computing usually considers dedicated homogeneous HPC systems to solve parallel problems.
- Distributed computing extends the parallel approach to heterogeneous general-purpose systems.
Both look at the parallel formulation of a problem.

But usually reliability, security, heterogeneity are not considered in parallel computing. But they are considered in Grid computing.

"A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable." (Leslie Lamport)

Figure 2c: Parallel and Distributed Data Mining Application Suite

2.7 Architectures

DMS implements a data-parallel record-based parallelization of data mining primitives. In Figure 2d an overview of DMS is shown; basically, the system consists of a manager and a number of servers which each process a subset of the data. Requests are received by the manager from the client data mining tool. The request may be handled entirely by the manager if it is a simple query; for example, a request for a list of data sets currently available to the client. If the request requires processing of the data, it is farmed out to the servers. The manager then consolidates responses from the servers and returns the required results to the client. The manager also
maintains a catalog detailing the data which DMS currently has access to [FA, 01].

![Diagram](image)

**Figure 2d: An Overview of DDM's Architecture**

The server components of DMS consist of four modules:

- The factory inputs data into DMS from a variety of sources including text files and Tandem NSSQL and Oracle databases. The factory converts the data into COREs (Column Represented Data Objects), DMS's internal data representation.

- The engine undertakes all operations on DMS data, such as building cross-tables and fetching rows of data to be passed through to the front end client.

- The cache stores and manages COREs and other internal DMS objects.

- The fileserver manages the permanent storage of and access to COREs on disk.

The high data processing speeds attained by DMS are due to several factors; primarily

- Effective parallelization of the data.
- Efficient encoding of the data into COREs; these structures are such that they are compact in memory and fast to access. The encoding also makes the COREs data-type independent, which simplifies the code and further compacts the data.

- Simple and optimized algorithms are used to manipulate the COREs. Storing data by column rather than by row, reducing disk access times and memory requirements for column based operations.

- Zoom-in functions enabling fast processing of subsets of the data, removing the need to scan the entire data.

A scalable solution for distributed applications calls for distributed processing of data controlled by the application resources and human factors. A distributed architecture for data mining is likely to reduce the communication load.

![Diagram](image_url)

Figure 2c: Architecture of Distributed Data Mining

A DDM system is a very complex entity that is comprised of many components like distributed algorithms, communication subsystem, resource management, task scheduling, user interfaces etc. It should
provide efficient access to both distributed data and computing resources to monitor the entire mining process and present results to the user in the appropriate format. The architecture that is used for this research work is the client server model along with the agents.

Shonali Krishnaswamy et al’s paper on, “A Hybrid Model for Improving Response Time in Distributed Data Mining”, presents a hybrid distributed data mining (DDM) model for optimization of response time [SSA, 03]. The model combines a mobile agent approach with client server strategies to reduce the overall response time. The hybrid model proposes and develops accurate \textit{apriori} estimates of the computation and communication components of response time as the costing strategy to support optimization. Experimental evaluation of the hybrid model is presented. The agent based model is a popular approach to constructing distributed data mining systems and is characterized by a variety of agent’s co-ordinating and communicating with each other to perform the various tasks of the data mining process. The motivation for using agent technology in distributed data mining stems from two reasons. Firstly there is the underlying basis of distributed data mining being a technology that has characteristics that are intuitively suited for an agent based approach [BGS, 98]. These characteristics are modular and well defined sub tasks, the need to encapsulate different mining algorithms to present a common interface, the requirement for interaction and cooperation between different parts of the system and the ability to deal with distribution. Secondly agent technology is seen as addressing specific concerns of increasing scalability and
enhancing performance by reducing the communication overhead associated
with the transfer of large volumes of data.

Masato Oguchi, Masaru Kitsuregawa’s paper on “Implementation and
Evaluation of Parallel Data Mining on PC Cluster and Optimization of its
Execution Environments”, presents a PC cluster which has a SAN-connection
as well as a LAN-connection, and examined its performance features
[MM, 01]. Basic characteristics of data transfer on the cluster are evaluated.
Performance of parallel data mining application on the SAN cluster is
examined. In the case of SAN cluster, each workstation can access all
shared disks directly. However, if a lot of workstations access the same
shared disk simultaneously, performance of application must degrade due
to I/O-bottleneck. A dynamic data declustering method, in which data is
declustered to several other disks through a SAN during the execution of
application, is proposed and evaluated. In order to improve the quality of the
rule, very large amounts of transaction data are analysed, which requires
considerable computation time. Previous researchers have given several
parallel algorithms for mining association rules [SK, 96], based on Apriori.
One of these algorithms, called Hash Partitioned Apriori (HPA), is
implemented and evaluated on the PC cluster. HPA partitions the candidate
itemsets among processors using a hash function, like the hash join in
relational databases.

Remote workstations, whose memories are available for application
execution workstations, are found dynamically during the execution. On
memory available workstations, a process is running to monitor the amount
of available memory periodically. "netstat -k" command provided by Solaris operating system is used to get memory information from the kernel statistics structure. Each time the process gets the information, the process broadcasts it to all application execution workstations. On application execution workstations, a client process is running and waiting for the information sent from the memory Monitoring processes running on memory available workstations.

David Skillcorn's, paper on "Strategies for Parallel Data Mining", presents a set of cost measures that can be applied to parallel algorithms to predict their computation, data access, and communication performance [Dav, 99]. These measures make it possible to compare different parallel implementation strategies for data mining techniques without benchmarking each one. For two reasons, data mining could be the killer application that parallel computing has been seeking. First, analyzing variation appears to be algorithmically complex and hence might require levels of computing power that only parallel computers can provide in a timely way. Second, the data sets involved are large and rapidly growing larger and parallel computers are organized to handle such large volumes effectively, although some data-mining problems are already taxing their limits. Data-mining algorithms can potentially be parallelized in many different ways. Because designing and implementing parallel programs is expensive, it is impractical to test all of these by building implementations and comparing them. Fortunately, practical complexity measures for parallel programming are rapidly maturing. This article shows how to use such
measures to assess different parallelization strategies for data-mining algorithms. The structure of some algorithms results in a double speedup phenomenon is also shown. A high-level cost analysis greatly simplifies the search for an effective algorithm.

Josenildo C. Da Silve, Chris Giannella, Ruchita Bhargava’s, paper on *Distributed Data Mining and Agents*, presents a Multi-Agent Systems (MAS) architecture for distributed problem solving [JCR, 05]. Distributed Data Mining (DDM) algorithms focus on one class of such distributed problem solving tasks—analysis and modeling of distributed data. This paper offers a perspective on DDM algorithms in the context of multiagents systems. It discusses broadly the connection between DDM and MAS. It provides a high-level survey of DDM, then focuses on distributed clustering algorithms and some potential applications in multi-agent-based problem solving scenarios. It reviews algorithms for distributed clustering, including privacy preserving ones. It describes challenges for clustering in sensor-network environments, potential shortcomings of the current algorithms, and future work accordingly. It also discusses confidentiality (privacy preservation) and presents a new algorithm for privacy-preserving density-based clustering.

Agents in MAS need to be pro-active and autonomous. Agents perceive their environment, dynamically reason out actions based on conditions, and interact with each other. In some applications the knowledge of the agents that guide reasoning and action depend on the existing domain theory. However, in many complex domains this knowledge is a result of the outcome of empirical data analysis in addition to pre-existing domain
knowledge. Scalable analysis of data may require advanced data mining for detecting hidden patterns, constructing predictive models, and identifying outliers, among others. In a multi-agent system this knowledge is usually collective. This collective "intelligence" of a multi-agent system must be developed by distributed domain knowledge and analysis of distributed data observed by different agents. Such distributed data analysis may be a non-trivial problem when the underlying task is not completely decomposable and computing resources are constrained by several factors such as limited power supply, poor bandwidth connection, and privacy sensitive multi-party data, among others.

Some features of a distributed scenario where DDM is applicable are as follows.

i. The system consists of multiple independent sites of data and computation which communicate only through message passing.

ii. Communication between the sites is expensive.

iii. Sites have resource constraints e.g. battery power.

iv. Sites have privacy concerns.

Typically communication is a bottleneck. Since communication is assumed to be carried out exclusively by message passing, a primary goal of many DDM methods in the literature is to minimize the number of messages sent. Some methods also attempt to load-balance across sites to prevent performance from being dominated by the time and space usage of any individual site. As pointed out in [Pro, 00], "Building a monolithic database,
in order to perform non-distributed data mining, may be infeasible or simply impossible" in many applications. The cost of transferring large blocks of data may be prohibitive and result in very inefficient implementations.

Klusch, M. Lodi, S. Gianluca, M.'s paper on "Role of Agents in Distributed Data Mining: Issues and Benefits" talks about performing data mining in a distributed and parallel computing environment [KLG, 03]. Data clustering is the method of grouping together all similar items to a cluster. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. (Hierarchical clustering). Similarity of two data streams can be measured using the Euclidean distance. According to the KDEC scheme an agent takes the role of the helper and it engages the other site agents in a negotiation to agree,

- what kernel function to use for computing the local density estimate samples and
- the use of the DE-cluster algorithm for local clustering based on the global estimate.

In density estimation (DE) based clustering the search for densely populated regions is accomplished by estimating a so-called probability density function from which the given data set is assumed to have arisen

Mahesh V.Joshi, Eui-Hong Han, George Karypis, Vipin Kumar, paper on "Efficient Parallel Algorithm for Mining Associations" states that in the large amounts of data has seen widespread applications in many practical domains [MEG+, 01]. The combinatorial complexity of the problem has
fascinated many researchers. Many elegant techniques such as Apriori have been developed to solve the problem on single processor machines. However, most available datasets are becoming enormous in size. Also their high dimensionality results in possibly large number of mined associations. This strongly motivates the need for efficient and scalable parallel algorithms.

The design of such algorithms is challenging. The first part of this work talks about the non sequential associations which utilize the relationships between events that happen together. The second part is devoted to the more general and potentially more useful sequential associations which utilize the temporal or sequential relationships between events, it is shown that many existing algorithms actually belong to a few categories which are decided by the broader design strategies. Overall the focus is to serve as a comprehensive account for the challenges and issues involved in effective parallel formulations of algorithms for discovering associations and how various existing algorithms handles them.

Mohammed J. Zaki’s paper on “Parallel and Distributed Association Mining: a Survey”, surveys the state of the art in parallel and distributed association-rule-mining algorithms and uncovers the field’s challenges and open research problems [Moh, 00]. This survey can serve as a reference for both researchers and practitioners. Since its inception, association rule mining has become one of the core data-mining tasks and has attracted tremendous interest among researchers and practitioners.
Association Rule Mining is undirected or unsupervised data mining over variable-length data and it produces clear, understandable results. It has an elegantly simple problem statement: to find the set of all subsets of items or attributes that frequently occur in many database records or transactions, and additionally, to extract rules on how a subset of items influences the presence of another subset.

A survey is made with the existing techniques in parallel and distributed data mining. In this research works novel methods are adopted by designing certain algorithms and models to obtain an optimized response time.