Chapter 1

Introduction and Statement of the problem

1.1 Introduction

Database Management Systems (DBMS) provide a convenient way to define storage structures, data storage, retrieval and manipulation of data in a wide spectrum of application domains. These application domains include banking, hospitals, inventory-management, decision support systems, reservation systems, stock brokerage systems, data analysis systems, weather forecasting etc. The business processes in these application-areas are not only complex, but also critical to the success of the entire business enterprise. Any kind of failure in the entire process, or for that matter even an unacceptable delay in the execution of the end user request for the publicized service, may cause huge business losses [1] [2].

According to the survey [3] carried out by the research team at Infosys research labs. in the year 2010, reveals that, performance plays a significant role in the success of Business
critical enterprise applications. It is essential that the software applications must meet the functional requirements accurately during software development process. However, it is equally important that the application must meet the non-functional requirement such as performance for the successful deployment. Further, the application must operate smoothly under high user-loads especially during peak business hours. For instance, an internet banking application becoming slow in its response to customers service requests or downtime of even a few minutes may cause huge business losses, operational overhead and dissatisfied customers. Similarly, a delay of few seconds in a performance-intensive application like air traffic control system may render the application useless. Therefore, it is essential that the database performance must be optimized using appropriate performance improvement techniques.

The performance problems in web applications [4] [5] arise due to the presence of many layers of software components between the end users and the data in the database. A typical web application comprises of 2 or 3 layers of software called ‘tiers’. Figure 1.1 shows a typical online order processing web application wherein, the user interacts with application in the first tier(normally through a browser), the web interface and business logic which implements the business processes and the rules of governance, runs in the middle tier and the data pertaining to the application resides in the data tier and is managed by Database Server. The major performance bottleneck occurs in the data tier because, all the queries submitted by client applications have to be executed by the query processor and execution engine of the DBMS.
Hence, it is essential to fine-tune the DBMS for performance gain for a satisfactory level of service to the end users in the Client tier.

The size of data storage, its transfer over the network and its management has grown tremendously. According to a survey carried out by NASSCOM-Crisil, the global data is growing at a whopping 45% every year. Figure 1.2 shows the expected growth of the global data between the years 2009 and 2020. This size would be 8 Zetabytes and 35 Zetabytes by the year-end of 2015 and 2020 respectively. This is due to the fact that the amount of data the end-user applications generate due to complex business transactions and also the proliferation of new data intensive applications such as data mining. All the database systems begin with a very small user-base and moderate size database providing excellent end user experiences. However, in due course of time, the number of users increase exponentially and the amount of data they generate also rapidly increases. Table -1.1 shows an increase of 60.9% in the number of users of online applications over one year period. This dramatic rise in user base will put tremendous pressure on the performance of Database Systems.


Figure 1.2 Rate of increase in digital data storage : Courtesy: Nasscom-Crisil report

<table>
<thead>
<tr>
<th></th>
<th>2007</th>
<th>2008</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Usage Rate</td>
<td>Number of Netizens (Millions)</td>
<td>Usage Rate</td>
</tr>
<tr>
<td>Online Shopping</td>
<td>22.1%</td>
<td>46</td>
<td>24.8%</td>
</tr>
<tr>
<td>Online Payment</td>
<td>15.8%</td>
<td>33</td>
<td>17.6%</td>
</tr>
</tbody>
</table>

As a result the end users start experiencing service delays and in the worst case complete service outages. A frequent occurrence of delay in service or a complete service outage may result in the end-user’s frustration paving way for a potential business loss as the customers would move to the other service providers. Most business firms will have an IT department to look into these issues and often with little knowledge about the performance bottlenecks in the system. The DBA either suggests the purchase of new hardware or try few up-gradation of either the hardware or software. However, this adhoc approach doesn’t ensure improvement in performance on a continuous basis and also for a prolonged period of time.

With the advent of powerful processors, Giga-byte sized RAMs and high performance storage devices, it is now possible to scale up the system performance to higher levels as the number of users increase and also the ever increasing size of the data generated by the applications. However, every business enterprise will not have the luxury of buying new hardware and also putting all that in place with zero or acceptable minimum downtime. Further, newer operating systems and higher versions of generic software like the DBMS are becoming bulkier, thus nullifying the effect of improved CPU speed and RAM sizes. Estimating the IT infrastructure requirement is a tough job, due to the fact that, by the time the IT requirement estimation process gets completed and equipments are purchased through long tendering process, the data requirements of the applications would have changed drastically. Any estimate that is arrived at looking into business needs of the day and a few years in future will always fall short of the needs that are ever increasing at rapid pace.

In the light of such a complex and critical business scenario, it is desirable to have Database systems that are intelligent enough to sense the degradation in performance and adapt themselves quickly to provide a satisfactory level of performance at all times without any downtime. The autonomic computing[35] is an emerging paradigm that was proposed by IBM with the sole aim of building self-managing, self-optimizing software systems. A generic autonomic framework has been proposed by IBM to provide solutions to some of the challenging problems related to self-management[118], performances using a knowledge component which is part of the framework.

Autonomic computing addresses four important issues. The first one termed as self-healing system [7] [8], wherein the software system can rectify the failure in one of its software components. The system maintains what is termed as symptom catalog for every type of symptom that the system is likely to generate and an appropriate action plan to rectify the problem that eliminates the symptom. The second one is related to addressing the issue of
degradation of system performance issues and application of corrective measures to enhance the system performance to the best possible level. The remaining two issues are related to self-configuring and self-protecting features. This framework provides a very good basis for developing self-tuning architectures for Database systems.

The autonomic computing system has the capability to monitor the performance indicators on a regular basis with a minimum performance overhead and analyze the data collected. It needs to generate a plan that initiates corrective measures to change certain key parameters of the system for enhanced performance. It uses some form of intelligent method of estimating the extent of modification required to some of the key parameters that help scale up the system performance. The complete setup uses Monitor, Analyze, Plan and Execute(MAPE) path in the Feedback-Control-Loop(FCL) architecture [9] described in the second chapter. The autonomic system has a knowledge-base, by using which it takes decisions on its own based on the context provided by the performance indicators called sensors and initiates appropriate corrective measures based on some rules through the tuning parameters called effectors [6].

Traditionally, the database systems were tuned manually by altering the parameters provided in the configuration file and bouncing back the system with these new set of values in place. Initially, the applications were not so demanding in terms of response-time hence, manual tuning was acceptable. However, with the improvement in database technology [10] and addition of several conflicting tuning parameters and also the highly demanding end-user applications in terms of system response-time and zero downtime, it is now possible to have an automatic tuning system in place. In a manually tuned system, the Database Administrator(DBA) must have an in-depth understanding of the tuning parameters; their impact on the system performance and must be highly skilled in anticipating different workloads; the user-load patterns and accordingly alter these parameters on a continuous basis. A mediocre DBA not very well-versed with various tuning parameters and their impact on the response-time, may arbitrarily set these parameters using trial and error method, resulting in either over-tuning or under-tuning leading to an unstable system [11]. Most of the small and medium size enterprises cannot afford to employ expert DBA for the purpose as they have their budgetary limitations. Even if an expert DBA is employed to tune the database, in such scenarios, he/she may either over-tune the system, resulting in wastage of system resources or under-tune the system resulting in unacceptable query response-times. Manual tuning is not only expensive but also tedious and time consuming task with no guarantee of the system-performance at all times. Therefore, it is desirable to devise an intelligent auto-tuning system.
that adapts well to the changing workload conditions, size of the database; the number of users of the system and provides optimum performance.

Most modern DBMSs provide several memory tuning parameters that have a significant impact on the system performance [12] [13]. However, it is essential that the effect of each tuning parameter on the system performance measured in terms of Query- response-time or throughput, must be thoroughly understood under various user-load conditions and also workload types. The Transaction Performance Processing Council (TPC) has established certain benchmark-workloads that represent typical application scenarios such as banking, online stock-brokering system, order-processing system and decision-support-system etc. There are several benchmark workload types [14] that are categorized as TPC-C, TPC-H, TPC-D, TPC-W etc. representing different workload scenarios by the TPC. For instance, TPC-C workload models a typical order-processing system that involves large number of database updates that affect only a small portion of the database. On the other hand, TPC-H represents, a Decision Support System (DSS) where in, only a few database queries are executed that affect a very large portion of the database. The number of users will also be varying randomly having a Gaussian distribution with maximum number of users during the peak hours of the business.

The database management systems that are proprietary, are not only expensive to own and operate, but also the vendors release newer versions after every one or two years with enhanced features and tuning parameters. This is a major issue in small and medium sized enterprises wherein, due to budgetary constraints they can’t afford to upgrade their database systems to newer version frequently. Moreover, the migration from old-version to new-version involves several issues such as data format incompatibility, new hardware requirement and newer expertise in its usage. The open-source database management systems also provide an alternative to keep the IT spending within limits, but they lack the advanced features that the proprietary DBMSs provide. The enterprises when they use database driven applications, the cost of ownership involves not only the cost of the Database System software, but it also includes running costs like salary of the personnel, cost of conducting regular training to the IT staff, Operators and cost of purchasing third party tools for performance analysis[103]. All these factors prompted the database vendors to develop built-in self-tuning features in their products to ease the burden of owning the Database Management software.

In the early stages of self-tuning, most database systems provided tuning advisors [15] in an attempt to help the DBA to perform informed database tuning. After running the tool it would estimate the extent of tuning required in terms of the number of tables to be indexed
including the columns on which the indexes are to be created. Further, the advise also provided the number of materialized views[72] to be created on the base-tables to avoid frequently constructing the views from their definitions. These features were provided by Oracle 9i[116]/10g, DB2 and MSSQL Server 2005.

Oracle 8i/9i/10g/11g, DB2, MSSQL Server are the major proprietary Database Management Systems that have ruled the market and MySQL, PostgreSQL, DERBY are the Open source Database management systems. MySQL is among the most popular open source DBMS that is used both on Windows and also on Linux platforms. In oracle 8i the tuning parameters had to be adjusted manually and DBMS be restarted for the values to take effect. However, in Oracle 9i, dynamic tuning feature was introduced wherein, the tuning parameters could be adjusted during run-time without requiring a system restart. This feature was very much desirable in a business-critical environment such as Banking or Stock broker system to improve system performance with zero downtime. In the newer version 10g, along with dynamic tuning, the Auto or Self-tuning feature was introduced which can be turned on or off. If this feature is ON, then the DBMS manages the System Global Area(SGA) components using tuning algorithms based on workload analysis. Following the autonomic computing trend the other commercial database systems, IBM’s DB2 and Microsoft’s MSSQL Server have also recently provided these features. The trend, in DBMS technology is clearly towards building Database systems that are autonomic, to take care of dynamic workload; database usage scenarios and self-tune the system for improved performance.

Data-mining, Knowledge Management systems, E-commerce, Online-Stores, Data-warehousing are some of the emerging applications that require support from Database management systems. Most modern DBMS provide necessary tools and packages to implement the above applications. The DBMS now provides facilities to import and export data from the older to the newer versions of the system and also data from other database management systems thus paving way for distributed database systems that are heterogeneous [16].

In a distributed environment, it is essential that the DBA should be able to monitor and control from a central place, the database systems that are spread across a large geographical area. In an attempt to stay ahead of their competitors, database vendors have tried to include newer and better features in every new release. This has increased the complexity of the database systems with hundreds of new configurations and dynamically alterable parameters. In this fast pace of system development, and to stay ahead of others, in this race the database vendors have ignored the need to provide management tools to the administrators. However, in
the recent past, the vendors have begun to provide administrative tools that help the DBA to manage the DBMS with considerable ease.

A majority of small and medium enterprises, educational institutes have to manage huge data in their day to day working. They also have to carry out data analysis for taking business decisions that will help them stay competent and better their business prospects in future. To manage the day to day problems arising out of continuous use of the system, enterprises have either a separate IT department or outsource the IT related maintenance work to a third party. Outsourcing may be expensive and also failure of the agency to promptly provide the service at the time of major crisis may result in enormous business loss. If the in-house IT staff is assigned the task of maintaining the Database systems, firstly the company may not be able to recruit an expert DBA due to budgetary limitations. Secondly, it is difficult to find competent DBAs who have the knowledge and ability to tweak the DBMS on a continuous basis for enhanced performance under all types of workload conditions. So, developing a self correcting system that anticipates performance degradation by keeping track of important performance indicators and initiates a corrective measure, this is more desirable. The corrective action would be quite similar to that of an expert DBA taking into account current workload, user load etc. Thus, alleviating the need for an expert DBA for tuning the DBMS and ensure enhanced performance under a variety of load conditions.

There are several factors that provided motivation for undertaking research in this area. Some of the most important motivating factors are:

1. The amount of business loss due to delay in the publicized E-Service and in the worst case a complete outage may cause damage to the extent of $75000 per hour [2]. Above all, the goodwill of the customers will also be lost and they may cause further damage by spreading the news. That is why some of the popular and well-respected E-Services spend lot of money on the IT department by recruiting expert DBAs, upgrading their hardware regularly etc.
2. An auto-tuned system besides avoiding business losses, it can cut-down on the running costs by eliminating the need for expert DBA who’s wages are very heavy.
3. Though most DBMSs have built-in auto-tuning[75] feature, not all enterprises can immediately migrate to these the newer versions. Further, if the DBMS is proprietary, migration will not only be time consuming but it also may be very expensive.
4. Providing auto-tuning feature both in the legacy DBMS and in the newer versions of DBMS that is both inexpensive and easy to implement without any downtime.

5. Setting up database system not only involves initial purchase cost, but also needs running expenses in terms of training the staff, purchase of third party management tools, and hiring system administrator for routine maintenance. All these factors add to the Total Cost of Ownership (TCO) of the software system [1]. One of the primary objectives of this research work is to reduce the TCO by devising a self-tuning system that need very little human intervention and expertise to use.

6. Further, effective and efficient use of system memory resource for enhancing the system-performance in terms end user query response-time was also a motivational factor.

7. Another important factor that acted as a motivational factor to take up research in this area has been the application of machine learning technique through the use of Artificial Neural Network (ANN) as indicated in [15] [17] as a future trend in auto-tuning of DBMS. Machine learning approach has the built in ability to deal with systems that are inherently nonlinear and have complex relationships between the input and output variables.

8. The convenience and effectiveness of providing efficient control action through the use of easy-to-build rule based system based on Fuzzy control theory was also yet another motivating factor. In this research, an attempt is made to combine the benefits of both the machine learning technique and fuzzy inference system for building a robust[86], scalable tuning system that adapts well to a variety of workload conditions.

1.2 Statement of the problem

Performance tuning of Database systems under a variety of dynamic workload conditions has been an important, interesting and challenging field of research that has attracted considerable attention [18] [19] [20] [21] [22] [23]. However, the research work has been focused mainly on static tuning approaches. With the advances in database technology, it is now possible to develop dynamic self-tuning techniques that adapt well to the changing external factors. One of the major goals of a self-tuning system is to optimize the use of dynamically tunable system resources and improve the overall system performance. The commercial database systems like Oracle 9i, DB2 and MSSQL Server do not have the facilities to inform the DBA as to when the tuning advisor has to be run. Further, the problem with
traditional auto-tuning is that the tuning adviser’s recommendations are to be effected by the DBA at regular intervals and in a highly dynamic workload scenario which is not practical. The auto-tuning of system memory provided by some of the commercial database systems involves extensive analysis [24] of statistical data generated by end user applications resulting in considerable computational overhead leading to delays in applying the corrective measures. The auto-tuning feature of modern DBMS that manages the DBMS memory, fails to effectively utilize all the subcomponents of the dynamic tunable subcomponents. Altering the sizes of the memory subcomponents of the DBMS memory blocks is not a trivial task [1] [25] as each subcomponent has a specific purpose and to be altered with the knowledge of the current workload scenario that is highly dynamic in nature.

Keeping these drawbacks in mind, it is desirable to build an auto-tuning system that triggers automatically at appropriate times depending on the changes in the workload and user-load patterns with less computational overhead. In this context, the problem may be stated as:

To develop Adaptive[112] techniques for performance tuning of database systems. This problem may be sub-divided into following sub-problems.

1. To characterize of tuning parameters and workload-types.
2. To explore the possibility of a mathematical model to relate Response time and tuning parameters
3. To develop MAPE(Monitor Analyze Plan and Execute) based tuning architectures using machine learning techniques.
4. To develop a Fuzzy based tuning architecture using fuzzy inference system.
5. To develop a Neuro-Fuzzy based tuning architecture combines the advantages of the above methods.

The research work was carried out in four phases. In the first phase, the impact of each tuning parameter on query response time under various workload types and scaling factors was established through experimentation. Further, the interaction between the tuning parameter was also noted. The experimentation on workload characterization helped in identifying workload based on BHR(Buffer-Hit-Ratio) and Database size. An attempt was made to model the DBMS memory that relates the Query-response-time, Buffer-size and User-load. The model shows good match with the experimental results under certain conditions. However, this model needs to be further refined to take into account other important factors like scaling factor, workload type etc. The experimental outcomes in phase one were used in constructing the training data-
set for the Neural networks and framing fuzzy-rules in fuzzy inference system. In the second phase, a Neural network based self-tuning architecture is proposed and its usefulness in improving the query-response-time was validated under various workload types of different scaling factors. In the third phase, Fuzzy logic based tuning architecture was proposed and its effectiveness in self-tuning was validated against different workload types and scaling factors. In the fifth and the final phase, a Neuro-Fuzzy based self-tuning control architecture was proposed which combines the advantages of both the soft computing techniques and its results were validated under all the workload types and scaling factors. In this phase, the application of neural network based tuning in a Distributed environment has also been explored with promising results.

1.3 Organization of the Thesis

In chapter 1, the general background of the research area, the compelling needs for designing and developing self-tuning database systems are discussed. The chapter also gives the statistical evidence for the rate of growth of data, the business enterprises have to manage and also the support modern database systems offer both in centralized as well as in distributed database scenarios to implement self-tuning paradigm.

In chapter 2, detailed description of the incremental research, methods, techniques and frameworks used by different researchers in the field of database tuning are presented along with their limitations. The chapter also highlights the identified important research gaps.

Chapter 3 describes the experimental setup used in evaluating the impact of each of the dynamic tuning parameters and their interdependency. The chapter establishes the impact of each tuning parameter on the query response time and also the interaction between them under a variety of workloads and user-load scenarios.

In Chapter 4, the machine-learning based tuning using Neural network approach is proposed; results are compared and validated with the established benchmarks.

Chapter 5, explores another self-tuning approach based on Fuzzy Inference system, wherein inputs are fuzzified and fuzzy rules are applied to generate the estimated values in fuzzy terms and finally they are de-fuzzified to crisp values to be used for carrying out the final tuning process. The results are compared and validated against the standard benchmarks.

In Chapter 6, a Neuro-Fuzzy based tuning architecture, taking into advantages of both Neural and Fuzzy systems is presented. The results are extensively validated under all the
benchmark workload types. This chapter also presents the adaptive tuning technique in Distributed database environment.

Finally, in chapter 7 conclusions are drawn, also the future scope for further research in this area have been discussed.