ARH_Db_Tuner: The GUI tool to Monitor and Diagnose the SGA Parameters Automatically

Hitesh KUMAR SHARMA[1], Aditya SHASTRI[2], Ranjit BISWAS[3]
[1]Assistant Professor, University of Petroleum & Energy Studies
[2]Vice-Chancellor, Banasthali University, Rajasthan-304022, India
[3]Head and Professor,CSE Dept, Jamia Hamdard (Hamdard University)
hkshitesh@gmail.com, adityashastri@yahoo.com, ranjitbiswas@yahoo.com

Database administrators should be aware of resource usages to maintain database system performance. As database applications become more complex and diverse, managing database systems becomes too costly and prone to error. Autonomic database tuning becomes more important than ever. One of the major issues to address in regards to ORACLE database performance is the size of the database. The bulk of the information consists of a large number of records, contained in many tables, each ranging from thousands of rows. There are many factors that can have direct effects on the performance of the database. CPU, Memory, Network, Disk I/O are among other factors. In order to make a database up and run efficiently, each factor must be addresses carefully, and the best tuning strategy must be applied for optimum performance. ORACLE performance issues are complex, and for a DBA, there is a large number of values to monitor and examine in order to decide on best tuning strategy.

The aim of the work behind this paper was to design and implement a Database Performance Tuning Measurement Toolkit for ORACLE Database Servers working on MS-Windows platforms. This system is called ARH_Db_Tuner. This system is the advance version of ORACLE Performance Monitoring Toolkit [19] (OPMT for short). Some of the future aspects of OPMT have been implemented in ARH_Db_Tuner. It is used for testing, analysis and reporting of the database performance.*

*** ARH stands for Aditya-Ranjit-Hitesh (The first names of Authors)**

Keywords: *SGA, SGA Dynamic Parameters, Database Tuning, DBA, Automated Tuning.*

1 Introduction

Database vendors are becoming aware that the human cost of operating large database systems is growing dramatically. As the scope of relational database functions has expanded in recent years, the complexity of database systems has also grown. The added complexity and the increase in data size (now frequently into tens of terabytes) have increased the burden on database administrators. The combination of increased data volumes, larger systems, and increased function, has motivated the need for autonomic capability within database management systems in order to reduce cost of ownership and to enable databases to operate in environments with limited access to skilled administration personnel [1].

A new and interesting approach to this management problem is an *autonomic DBMS* that is able to automatically manage its resources to maintain acceptable performance in the face of changing conditions [2]. Other terms used in the literature for an autonomic DBMS are *self-tuning DBMSs* [3] and *no knobs operation* [4]. An autonomic DBMS must be able to perform the configuration tasks currently carried out by DBAs to initially set up a system. Typical configuration tasks include determining appropriate allocations for main memory areas such as the buffer pools and the sort heap, mapping database objects (user defined tables and indexes) to disk storage and mapping database objects to

buffer pools. Performing configuration tasks requires knowledge of the available system resources and of the characteristics of the workload.

Many system performance evaluation models have been proposed, varying in degree of accuracy, cost to build and run, and the reliance of the method on configuration-specific parameters. Many of these models are concerned with evaluating the appropriateness of a particular hardware configuration on the mix of programs that will run on the system. These methods, generally used for capacity planning, include queuing models, simulation, and monitors of contrived or real workloads. Detailed simulation models, such as IBM's CSS, FIVE [5], and ANCICSVS [6], QSIM [7], and APLOMB [8], are complicated, requiring large initial investments of time and money. These investments are not warranted by a single use; thus, these simulators are best suited for development by specific manufacturers and may not be publicly available. General packages, such as SCERT and CASE, are more useful as flexible evaluation tools [9].

On the other hand, and as businesses across all industries are getting more dependent on IT infrastructure, the problem of availability of IT systems becomes an increasingly strategic business concern. This is because the interdependencies of modern business are often linked through software. On one hand, employees, customers, and partners communicate and conduct commerce through networked systems, at a more intense level today than ever before. On the other hand, when one system in the network fails, it can break dependencies and impact business processes across the enterprise and beyond to customers and partners.

To avoid the ramifications of database downtime, many corporations have taken a renewed interest in database availability. For some, the goal is continuous availability, where a database server never fails, which would result in so-called unplanned downtime. Most companies do not need such a stringent level of database availability; they are satisfied with high availability, which allows for a small room of planned downtime allocated for database maintenance. Most strategies for continuous availability and high availability assume that a slight repair to hardware and/or software is all that is required for recovery.

High availability is often associated with fault-tolerant systems. The term *fault-tolerant* means a system can operate in the presence of hardware component failures A single component failure in a fault-tolerant system will not cause a system interruption because the alternate component will take over the task transparently. As the cost of components continues to drop, and the demand for system availability increases, many *non-fault-tolerant* systems have redundancy built-in at the subsystem level. As a result, many non-fault-tolerant systems can tolerate hardware faults— consequently, the line between a fault-tolerant system and a non-fault-tolerant system becomes increasingly blurred [10, 11, 12, 13]

Another key point in bringing database systems users' satisfaction is the system performance. Performance can be defined as the ability of a system to deliver the results based on the request of the users, while keeping them satisfied. The key point to remember is satisfaction. If current database key indicators reveal that the database is running optimally, but users are not satisfied with the response times, then there will be a need for tuning [14].

One of the biggest responsibilities of a DBA is to ensure that the database is tuned properly. The ORACLE RDBMS is usually tunable and allows the database to be monitored and adjusted to increase its performance. One should do performance tuning for the following reasons [15]:

- The speed of computing might be wasting valuable users' time (users waiting for response);

- Enable your system to keep-up with the speed business is conducted; and
- Optimize hardware usage to save money (companies are spending millions on hardware).

Tuning the database performance is not a simple task and it depends on the specific requirements, the operating system and the target hardware. There is no "one fits it all" approach. The goal is to avoid obvious slowdowns and balance the available resources (I/O bandwidth, memory and CPU) [16].

2. Database Performance Tuning

Every DBA has experienced a situation in which an application slows down after it has been in production for a while. But why this happens is not always evident. Perhaps the number of transactions issued has increased or maybe the volume of data has increased [17]. The overall performance of a system can be generally measured according to transaction response times, that is, the time it takes to complete a query or task, which is also the time the user must wait for the task to finish and possibly return results. Slow (long) response times translate into bad performance and frustrated users, whereas quick response times mean better performance and happy users. Database system may show different levels of performance (good or bad) at different times of the day, according to heavy or light user activity. If a user query takes a relatively long time to finish, based on previous tuning, it is an indication that system may have a performance problem that may be resolved, and this should be investigated.

Another way to know if there is a performance problem is by simply monitoring the system on a regular basis, and this is the theme of the work behind this paper.

3. The ARH_Db_Tuner System

The structure of the system include buffer cache, database blocks tuning, input/output tuning, operating system tuning, shared pool and others. It is out of the scope of this paper to include the flowcharts of these.

ARH_Db_Tuner consists of three functional parts, each is interfaced through a number of screens, the first part is dedicated for testing the system, and it is interfaced by Five screens that the user can move between by selecting the appropriate tab. Each screen is related to different category.

The second part is for displaying the various options to generate graph just on one click. There are for different charts available for some major SGA Parameter.

The Third functional part is the Auto/Manual Refreshing of Charts and Table data.

The user interface screens are available to perform system tests, these screens are used for:

Shared pool: The first screen "Shared Pool" is divided into three database blocks. These are: Library cache—in which three records were retrieved, these are: Gethitratio, Pins and Reloads. When the button “Calculate reloads–to–Pins Ratio” is pressed the system will display the value of “sum(reloads)/sum(pins)”. This value is analyzed and the result is displayed in a separate window as either "good" or "bad” alongside some recommendation that will appears if the analysis is bad. This is to inform the user about the necessary actions that must be taken to improve the system. In addition to the recommendation, some brief comments that are related to the monitored database system status will be displayed also as shown in figure 2 below. Shared pool reserved area—in which two records are retrieved; these are: "Request Misses" and "Request Failures". These values are displayed when the V$SHARED_POOL_RESERVED” button is pressed. Data dictionary cache—in which three values are retrieved; these are: "Parameter", "Gets" and "Getmisses". When the user clicks on “sum of GETMISSES to

sum of GETS" button, OPMT calculates the percentage of the sum of "GETMISSES to the sum of GETS" and checks if this value is less than 15 per cent, then it will show some analysis regarding this calculated value.

Redo Log buffer: The Third screen "Redo log buffer" is used for two system testing functionalities; these are:Session Wait. This is used to check the length of system waiting period in seconds. Pressing on "Ckeck Seconds in Wait" button, as shown in the upper part of figure 4 below, four pieces of information will be retrieved and displayed. These are SID "Session ID", the Event, the waiting period in Seconds, and State of the system. If the waiting period is relatively too long, the system will display "Bad" in the "Analysis" window in the left bottom corner of the screen. It also gives the necessary recommendations to deal with this problem and some comments too. System Status "Sysstat". This to check for the value of redo buffer allocation retries. Again, if it is relatively large the system will show the results of the analysis, recommendation and comments too.

Input/Output: The third user interface is the "Input/Output" screen and it is about scan statistics. This screen is mainly used for retrieving and displaying two values, namely: the name and the value of the scanned table. Again, if the value is undesirably high, the system will display "Bad" in the Analysis window and recommendations and comments will be given too. This screen is shown in figure 5 below.

Undo Segments. This is the forth screen which is used for checking the number of deleted values. The user can get the number of deleted values by clicking on the "Check if number of waits > 1% number of requests" button. The desirable value must be less than 1% of the total number of requests. If this is the case, the system will display "Good", otherwise, the user will get "Bad" result as described above. This is shown in figure 6 below.

The graphical representation of the statistics has been show in figure 7, 8, 9 and 10. All graphical representation windows are having a Auto Refresh buttons. On clicking on these buttons the graphs will refresh automatically after 5 seconds and display the real time statistics of the database.

Fig. 1. Login Window

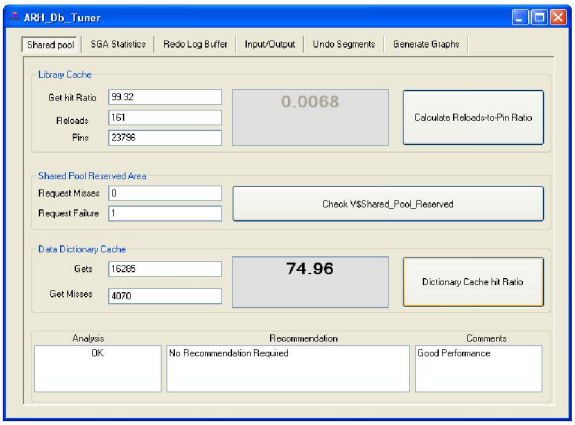

Fig. 2. Shared Pool Statistics Window

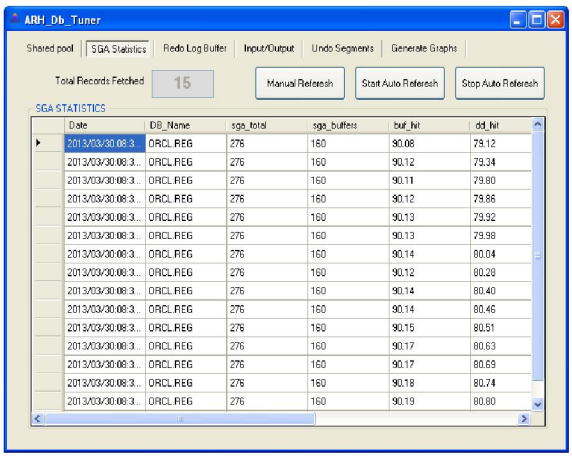

Fig. 3. SGA Statistics Window

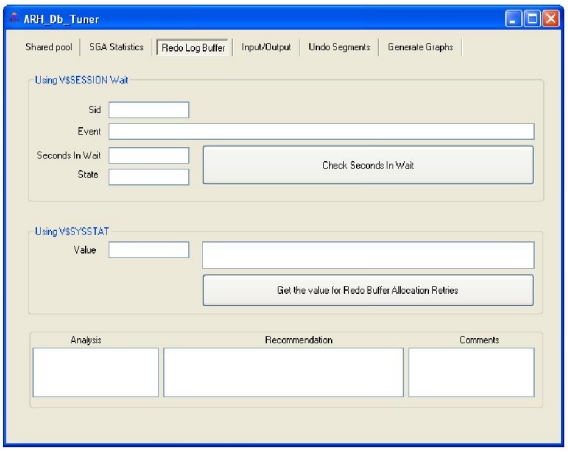

Fig. 4. Redo Log Buffer Statistics Window

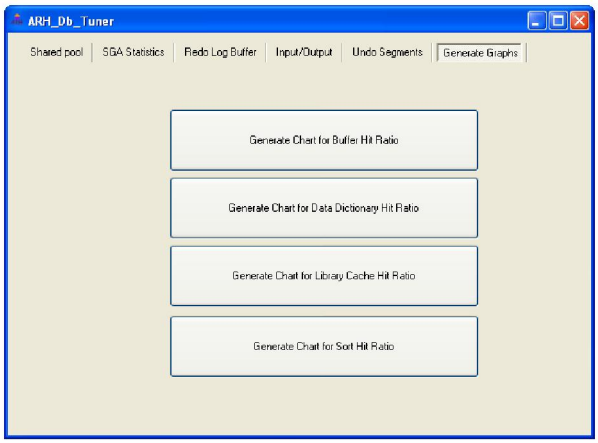

Fig. 7. Graph Generation Window

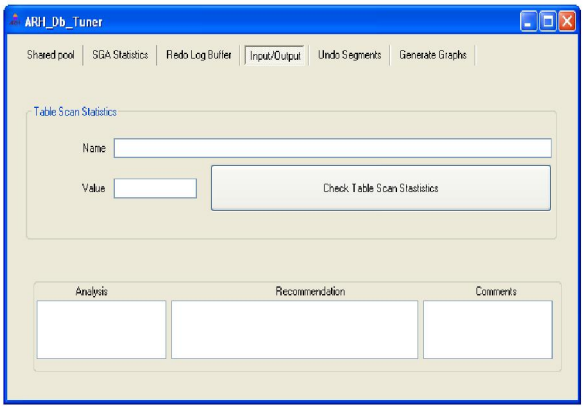

Fig. 5. I/O Statistics Window

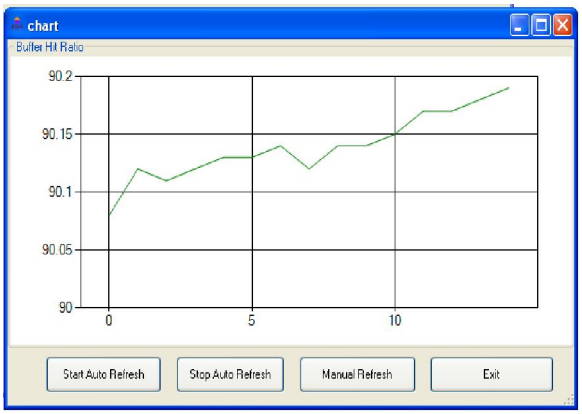

Fig. 8. Buffer Hit Ratio Graph Window

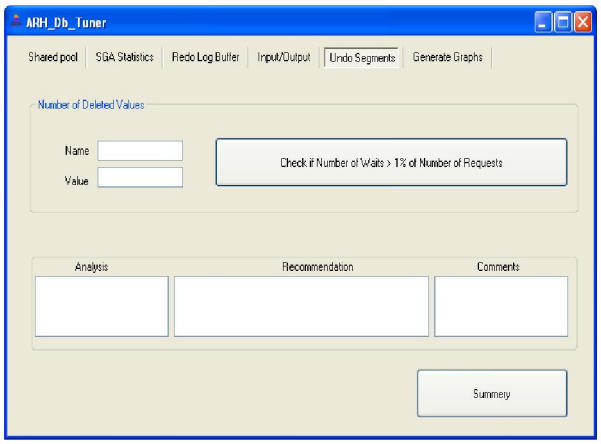

Fig. 6. Undo Segment Statistics Window

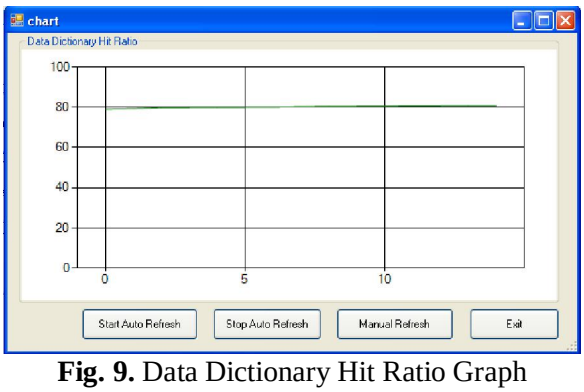

Fig. 9. Data Dictionary Hit Ratio Graph Window

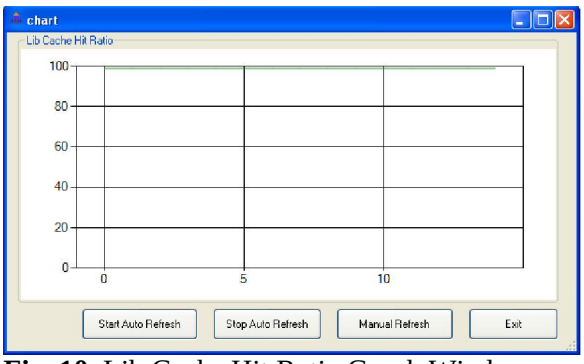

Fig. 10. Lib Cache Hit Ratio Graph Window

4. Conclusion and Future Work

Tuning the database can become quite complex, but modern databases offers the administrator an unparalleled ability to control the PGA and SGA. Until old databases evolve into a completely self-tuning architecture, the DBA was responsible for adjusting the dynamic configuration of the system RAM. Automated SGA adjustment scripts can be used to allow the DBA to grow and shrink the SGA regions. Manual tuning cost more for an organization but it is one of the major needs for an organization to attract the customer. So we have proposed a solution to fulfill the need of an organization in the shape of this Automation Framework. This framework will not take any cost and it will give faster result compare to manual tuning.

ARH_Db_Tuner is a database performance monitoring and tuning system that was developed to be used with ORACLE Database system run on Microsoft Windows operating systems. The main objective of speeding up database systems performance tuning processes and simplifying the DBA duties in a reliable and flexible manner was achieved. The system has been tested and proved efficient and reliable. ARH_Db_Tuner provides a flexible structure that can be further developed with minor changes. It provides friendly interfaces that can be easily used by the DBA and developers to monitor database performance. All test information and produced history files for each performance test are stored and can displayed and examined at any time.

The system design is flexible and can be easily expanded. Future expansions and changes may include:

- Make the system multi-lingual.
- Expand it to work on other database management systems like MS-SQL Server, DB2.
- Make it to run on other operating systems, like UNIX and Linux.

References

[1] Lightstone, S. *et al.*, "Toward Autonomic Computing with DB2 Universal Database", *SIGMOD Record*, Vol. 31, No. 3, September 2002.

[2] Xu, X., Martin, P. and Powley, W., "Configuring Buffer Pools in DB2 UDB", IBM Canada Ltd., the National Science and Engineering Research Council (NSERC) and Communication and Information Technology Ontario (CITO), 2002.

[3] Chaudhuri, S. (ed). Special Issue on, "Self-tuning Databases and Application Tuning", IEEE Data Engineering, *Bulletin 22(2)*, June 1999.

[4] Bernstein, P. *et al.*, "The Asilomar Report on Database Research", ACM SIGMOD Record 27(4), December 1998, pp. 74 - 80.

[5] Nguyen, H. C., Ockene, A., Revell, R., and Skwish, W. J., "The role of detailed simulation in capacity planning". IBM Syst. J. 19, 1 (1980), 81-101.

[6] Seaman, P. H., "Modeling considerations for predicting performance of CICS/VS systems", IBM Syst. J. 19, 1 (1980), 68-80.

[7] Foster, D. V., Mcgehearty, P. F., Sauer, C. H., and Waggoner, C. N., "A language for analysis of queuing models", *Proceedings of the 5th Annual Pittsburgh Modeling and Simulation Conference* (Univ. of Pittsburgh, Pittsburgh, Pa., Apr. 24-26). 1974, pp. 381-386.

[8] Reiser, M., and Sauer, C. H., "Queuing network models: Methods of solution and their program implementation", *Current Trends in Programming Methodology*. Vol. 3, Software Modeling and Its Impact on Performance, K. M. Chandy and R. T. Yeb, Eds. Prentice-Hall, Englewood Cliffs, N. J., 1978, pp. 115-167.

[9] Borovits, I., and Neumann, S., "Computer Systems Performance Evaluation", D.C. Heath and Co., Lexington, Mass., 1979.

[10] Enrique Vargas, "High Availability Fundamentals", Sun BluePrints™ OnLine, November 2000,
http://www.sun.com/blueprints

[11] Harry Singh, "Distributed Fault-Tolerant/High-Availability Systems", Trillium Digital Systems, a division of Intel Corporation, 12100 Wilshire Boulevard, Suite 1800 Los Angeles, CA 90025-7118 U.S.A. Document Number 8761019.12.
[12] David McKinley, "High availability system. High availability system platforms", Dedicated Systems Magazine - 2000 Q4 (http://www.dedicated-systems.com)
[13] Sasidhar Pendyala, "Oracle's Technologies for High Availability", Oracle Software India Ltd., India Development Centre.
[14] James Koopmann, "Database Performance and some Christmas Cheer", an article in the Database Journal, January 2, 2003.
[15] Frank Naudé, "Oracle Monitoring and Performance Tuning", http://www.orafaq.com/faqdbapf.htm
[16] Michael Marxmeier, "Database Performance Tuning", http://www.hp-eloquence.com/support/misc/dbtuning.html
[17] Sharma H., Shastri A., Biswas R. **"**Architecture of Automated Database Tuning Using SGA Parameters", *Database System Journal*, Romania, 2012
[18] Sharma H., Shastri A., Biswas R. "A Framework for Automated Database TuningUsing Dynamic SGA Parameters and Basic Operating System Utilities", *Database System Journal*, Romania, 2013
[19] Mihyar Hesson, "Database performance Issues"
[20] PROGRESS SOFTWARE, Progress Software Professional Services, http://www.progress.com/za/services/index.ssp
[21] Ralph Kimball,http://www.informatik.uni-trier.de/~ley/db/indices/atree/k/Kimball:Ralph.html
[22] Information Builders, "OLAP–Online Analytical Processing–Tools", http://www.informationbuilders.com/olap-online-analytical-processing-tools.html

About Authors:

Hitesh KUMAR SHARMA, The author is an Assistant Professor in University of Petroleum & Energy Studies, Dehradun. He has published 8 research papers in National Journals and 5 research paper in International Journal. Currently He is pursuing his Ph.D. in the area of database tuning**.**

Aditya SHASTRI, Ph.D. MIT, Published about 200 research papers in international journals on Graph Theory with applications in Communication, Computer Graphics and Parallel Processing ,Vice Chancellor, Director, BanasthaliUniversity, Banasthali, INDIA

Ranjit BISWAS, Head and Professor Jamia Hamdard (Hamdard University) Published about 100 research papers in International journals/bulletins.