SGA Dynamic Parameters: The Core Components of Automated Database Tuning

Hitesh KUMAR SHARMA, Aditya SHASTRI, Ranjit BISWAS, Sanjeev KUMAR SINGH
[1]University of Petroleum & Energy Studies
[2]Banasthali University
[3]Hamdard University
[4]Galgotia University Noida
hkshitesh@gmail.com, adityashastri@yahoo.com, ranjitbiswas@yahoo.com, sksingh8@gmail.com

The efficient use of primary memory is one of the major key for the good performance achieved from the any kind of server. The management of the various components stored in the main memory is the key challenge to get the desired throughput from an application running on the server. As we know the DBMS mostly works on client-server architecture. So the memory management for the DBMS's components stored in the main memory of the server is the critical task for a DBA. The DBA should have the knowledge of the components those are stored in main memory during runtime. This paper helps the DBA to get the detailed description of these core components of DBMS.
Keywords: *SGA, DBMS, DBA, Dynamic Parameters.*

1 Introduction

The Complexity of the database is increasing day by day. Due to the Exponential growth of the amount of data and its complexity, the responsibilities of DBA has been increased in the same ratio.
One of the major responsibilities of DBA is to make the database available 24*7 for the user and the response time should be minimum. There are many components in a DBMS those are responsible for poor response time. These may be categories as software component (database design, SQL query parsing and optimize etc.) and hardware component (disk, main memory, network component etc.). Out of these many software and hardware components the major part played in performance tuning is the memory management component.
In this paper we will explain in detail the role of memory management in performance enhancement of a DBMS. The System Global Area (SGA) management impacts a lot on the performance of a DBMS. It is a big challenge in itself to predict and allot a right amount of memory for different components of SGA. The memory captured by an SGA is called as Instance of the database server. The management of this instance is done by several dynamic SGA Parameters or instance parameters. This paper will give the detail overview of SGA and its dynamic parameters.

2. System Global Area (SGA)

Inside the Instance of a database server, data is stored in two places: in memory and on disk. Memory has the best performance but also has the highest cost. Disk, on the other hand, can store vast amounts of data and cost effective but has very slow performance relative to memory. Due to the better performance, it is desirable to use memory to access data whenever possible. But because of the vast amounts of data usually accessed and the number of users who need this data, there is a lot of contention on this resource. To make most effective use of memory, you must achieve a balance between the memory used for DBMS caching and the memory needed by the users. The **System Global Area** (SGA) is the commonly shared main memory space on database server that is globally

shared by the clients connected to the server. If multiple users are connected to the database servers at a time than the pool of memory allotted to the server is called SGA and the SGA with background processes is called an Instance. An SGA and Oracle processes constitute an Oracle instance. Oracle automatically allocates memory for an SGA when you start an instance, and the operating system reclaims the memory when you shut down the instance. Each instance has its own SGA. The SGA is read/write. All users connected to a multiple-process database instance can read information contained within the instance's SGA, and several processes write to the SGA during execution of Oracle.

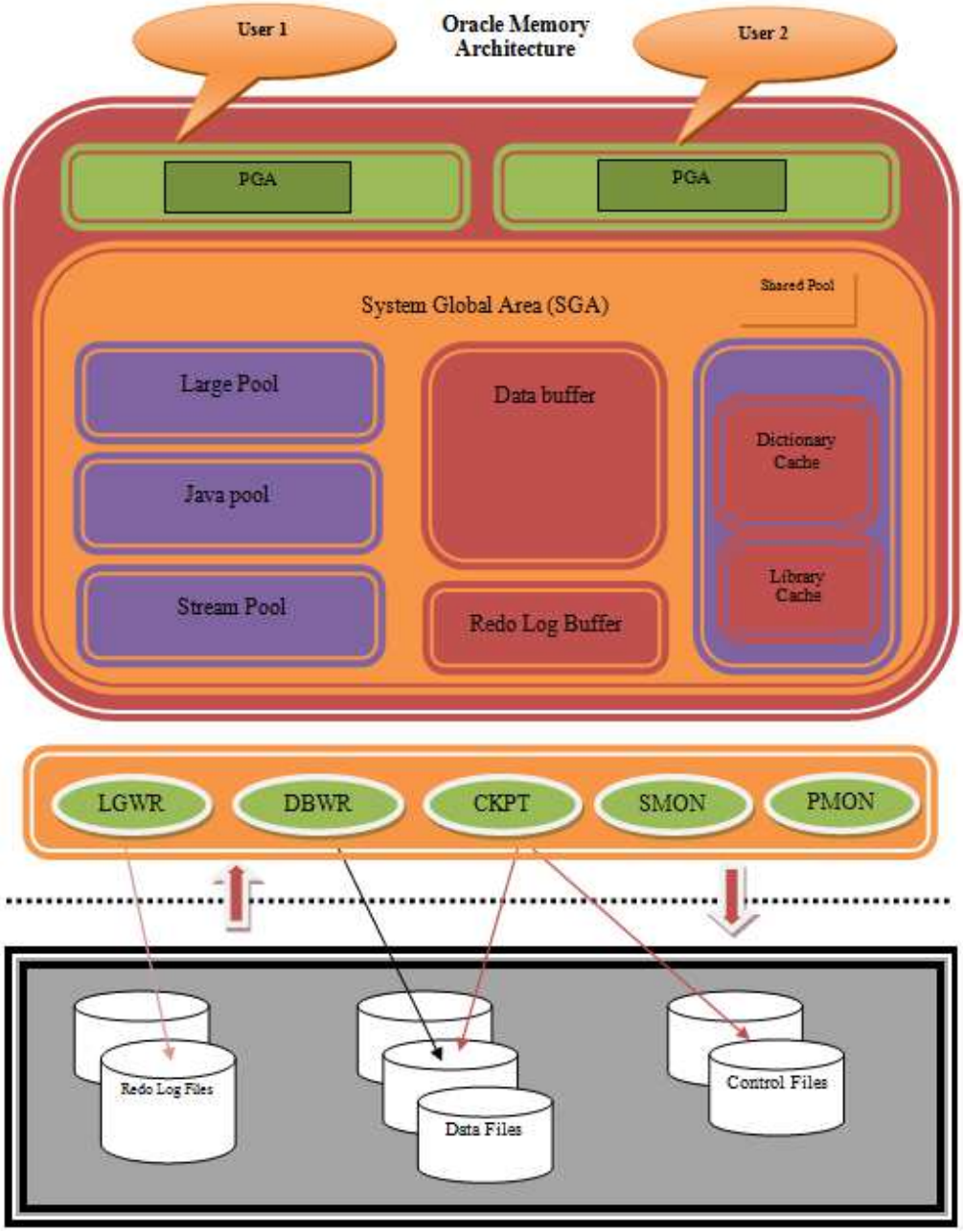

Fig.1 Oracle Memory Architecture

The SGA contains the following data structures:

- Database buffer cache;
- Redo log buffer;

- Shared pool;
- Java pool;
- Large pool (optional);
- Streams pool;
- Other miscellaneous information.

The Database Server Instance (DSI) also reserved some fixed space that is used to store instance information and the information of background processes running behind. User data is not stored here. The SGA includes information communicated between processes, such as locking information. If the system uses shared server architecture, then the request and response queues and some contents of the PGA are in the SGA.

The detailed description of the various components of SGA is given below.

2.1. Database Buffer Cache

This component of SGA holds the data blocks fetched from the disk to satisfy user's query. If the DML operations are performed on data blocks then The DML operations on data objects are first performed on these blocks and transferred to the data files by the DB Writer (i.e. DBWR) processes.

Free buffer, pinned buffer and dirty buffer are the part of LRU list. Free buffer will take place towards the LRU end of the list and dirty buffer will take place towards the MRU end. The Diagram (Figure 2) shows the detailed view of Data Buffer Cache. This diagram clearly shows the transferring the data from data files present in the disk to the various parts of data buffer cache. It also shows the role of LRU list and Write list. The transferring of the data processed by background process.

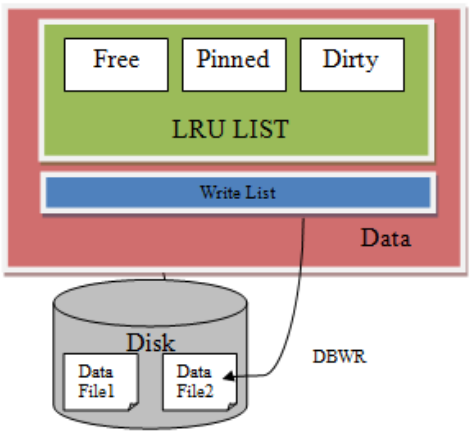

Fig.2 Data Buffer Cache (Components)

When a process requires data, it starts looking for it in the data buffer. If it finds that data, it is a cache hit otherwise it is cache miss. In the event of a cache miss, the process has to copy the data from data file into the LRU List. Before copying, the process will first start looking for free space in the LRU list starting from the LRU end. As the process starts to scans for free space, if it hits a dirty data, it copies that over to the write list. It continues until it has found enough free space or if it hits a threshold to search. If it hits the threshold, it asks DBWR process to write some of the data blocks from write list to data files and free up space. If it gets the required free space, it reads the data into the MRU end of LRU List. Whenever an Oracle process accesses any of the data in the LRU list (cache hit), the data is moved to the MRU end of that buffer. Over the time, the older data (except for full table scans) moves to the LRU end of the buffer.

2.2. Redo Log Buffer

Redo log buffer is used for recovery management. Whenever an SQL operation is performed on the database, a corresponding entry is made in redo log buffer. Due to the limited size of buffer the entries will be transferred to the online redo file present in disk using the LGWR

process. If some failure occurs due to abnormal termination then the data can be recovered using this buffer or online redo file.

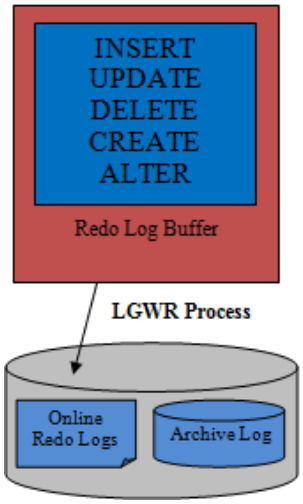

Fig.3 Data Buffer Cache (Block Movement)

The initialization parameter LOG_BUFFER determines the size (in bytes) of the redo log buffer. In general, larger values reduce log file I/O, particularly if transactions are long or numerous. The default setting is either 512 kilobytes (KB) or 128 KB times the setting of the CPU_COUNT parameter, whichever is greater.

2.3. Shared Pool

The shared pool component of SGA contains the parsed SQL statements and the metadata about the database objects. It divided into two major parts.

- Library Cache;
- Data Dictionary Cache.

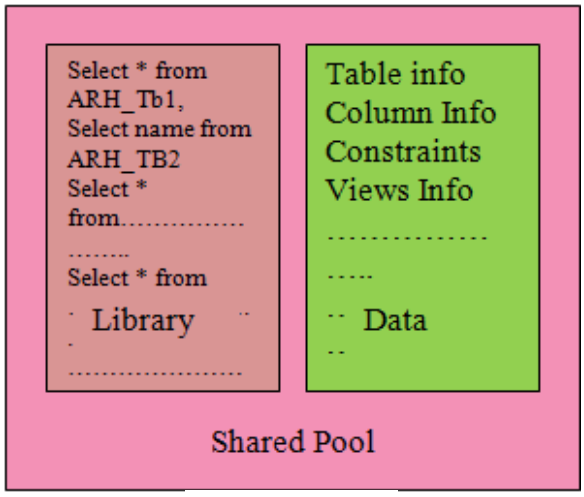

Fig.4 Shared Pool

The total size of the shared pool is determined by the initialization parameter SHARED_POOL_SIZE. The default value of this parameter is 8MB on 32-bit platforms and 64MB on 64-bit platforms. We can increase or decrease the value of this parameter as per the performance need.

2.3.1. Library Cache

The library cache stores the parsed SQL statements. If the SQL statements parsed succesfully then it gets entry in library

cache to reduce the parsing time in future for same SQL query. It is the reserved space in shared pool component of SGA. The diagram (Figure 4) shows the information stored in library cache. Some SQL queries has been shown in the diagram. The queries are stored in parsed form. The use of bind variable in SQL query make this space more useful Otherwise the same query will store multiple time just because of the difference in data used in where condition. Shared SQL areas are accessible to all users, so the library cache is contained in the shared pool within the SGA.

2.3.2. Dictionary Cache

The data dictionary is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle.

The data dictionary is accessed so often by Oracle that two special locations in memory are designated to hold dictionary data. One area is called the **data dictionary cache**, also known as the **row cache** because it holds data as rows instead of buffers (which hold entire blocks of data). The other area in memory to hold dictionary data is the library cache. All Oracle user processes share these two caches for access to data dictionary information.

2.4. Large Pool

The database administrator can configure an optional memory area called the large pool to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with more than one database);
- I/O server processes;
- Oracle backup and restore operations.

By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle can use the shared pool primarily for caching shared SQL and avoid the performance overhead caused by shrinking the shared SQL cache. In addition, the memory for Oracle backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool. The large pool does not have an LRU list. It is different from reserved space in the shared pool, which uses the same LRU list as other memory allocated from the shared pool.

2.5. Java Pool

Java pool memory is used in server memory for all session-specific Java code and data within the JVM. Java pool memory is used in different ways, depending on what mode the Oracle server is running in. The Java pool advisor statistics provide information about library cache memory used for Java and predict how changes in the size of the Java pool can affect the parse rate. The Java pool advisor is internally turned on when statistics_level is set to TYPICAL or higher. These statistics reset when the advisor is turned off.

2.6. Streams Pool

In a single database, you can specify that Streams memory be allocated from a new pool in the SGA called the Streams pool. To configure the Streams pool, specify the size of the pool in bytes using the STREAMS_POOL_SIZE initialization parameter. If the size of the Streams pool is greater than zero, then any SGA memory used by Streams is allocated from the Streams pool. If the size of the Streams pool is zero, then the memory used by Streams is allocated from the shared pool and may use up to 10% of the shared pool.

3. SGA Parameters

As we have discussed above that the SGA is the collection of various components.

The components have been divided on the basis of their requirements to achieve better functionality. The components have different memory requirement to store different kind of information. The accessibility mode, time and background process is also different for different component. By altering the memory assignment for them will have positive or negative impact on DBMS performance. Most of the DBMS provides the list of parameters associated with each component of SGA. By altering the value of these components the DBA can control the memory assignment for each component separately.

The following table lists the name of parameters with its associated SGA component.

Table 1. SGA Components and Corresponding SGA Parameters

SGA Component	Initialization Parameter
The buffer cache	DB_CACHE_SIZE
The buffer cache	DB_BLOCK_BUFFER
The buffer cache	DB_BLOCK_SIZE
The buffer cache	DB_KEEP_CACHE_SIZE
The buffer cache	DB_RECYCLE_CACHE_SIZE
The shared pool	SHARED_POOL_SIZE
The Redo Log Buffer	LOG_BUFFER
The large pool	LARGE_POOL_SIZE
The Java pool	JAVA_POOL_SIZE
The Streams pool	STREAMS_POOL_SIZE

The value of these parameters can be changed using **alter system** command. By altering the value of these parameters the DBA can increase or decrease the value of corresponding part of SGA.

4. Dynamic SGA Parameters

The parameters listed above can be changed by alter system command but all of them cannot give their changed effect on running instance. Some of them give their effect on restarting the instance. But the rest parameters can affect the instance in running state. These parameters who gave their effect on running instance are called Dynamic parameters. Since these parameters are used to alter the SGA components hence these are called Dynamic SGA parameters.

Table 2. SGA Components and Corresponding Dynamic SGA Parameters

SGA Component	Initialization Parameter
The buffer cache	DB_CACHE_SIZE
The shared pool	SHARED_POOL_SIZE
The Redo Log Buffer	LOG_BUFFER

SGA Component	Initialization Parameter
The large pool	LARGE_POOL_SIZE
The Java pool	JAVA_POOL_SIZE

These parameters are also called Auto Tuned parameters because these can be changed at runtime by using any auto executed application.

4.1. DB_CACHE_SIZE

This parameter is used for allocating the size of database buffer. By changing the value of this parameter the DBA can change the size of data buffer. The I/O overhead can be reduced by allocation sufficient memory to data buffer using this parameter. The value is calculated by multiplying No. of Blocks to Block size.

```
DB_CACHE_SIZE = 4M * Number of
CPU * Granule
```

The value larger than this is rounded up to the nearest granule size.

4.2. SHARED_POOL_SIZE

This parameter is used to assign the memory space to shared pool. The shared pool contains parsed SQL queries, PL/SQL procedures, Triggers, cursors, and other structures. If you set PARALLEL_AUTOMATIC_TUNING to false, then Oracle also allocates parallel execution message buffers from the shared pool. Larger values improve performance in multi-user systems. Smaller values use less memory.

4.3. LOG_BUFFER

LOG_BUFFER specifies the number of bytes allocated to the redo log buffer. Larger values reduce I/O to the redo log by writing fewer blocks of a larger size. Particularly in a heavily used system, this may help performance. In general, larger values for LOG_BUFFER reduce redo log file I/O, particularly if transactions are long or numerous. In a busy system, a value 65536 or higher is reasonable.

4.4. LARGE_POOL_SIZE

This parameter is used to assign the memory for backup and recovery of the database. Parallel execution allocates buffers out of the large pool only when PARALLEL_AUTOMATIC_TUNING is set to true. You can specify the value of this parameter using a number, optionally followed by K or M to specify kilobytes or megabytes, respectively. If you do not specify K or M, then the number is taken as bytes.

4.5. JAVA_POOL_SIZE

This parameter specifies the size of the Java pool, from which the Java memory manager allocates most Java state during runtime execution. This memory includes the shared in-memory representation of Java method and class definitions, as well as the Java objects that are migrated to the Java session space at end-of-call.

5. Conclusion

In this paper we have explained the memory architecture of one of the leading DBMS (i.e. Oracle 10g). It describes the instance configuration of oracle database server. It explains the detailed view of various parts of SGA (System Global Area) and the parameters used to fix and manipulate the memory for the various parts of SGA.

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 ApplicationTuning", IEEE Data Engineering, *Bulletin 22(2)*, June 1999.

[4] Bernstein, P. *et al*., "The Asilomar Report on Database Research", ACMSIGMOD 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/VSsystems", IBM Syst. J. 19, 1 (1980), 68-80.

[7] Foster, D. V., Mcgehearty, P. F., Sauer, C. H., and Waggoner, C. N., "A languagefor 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., "Queuingnetwork models: Methods of solution andtheir program implementation", *CurrentTrends in Programming Methodology*. Vol.3, Software Modeling and Its Impact onPerformance, 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 Blue Prints™ 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, CA90025-7118 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.hpeloquence.com/support/misc/dbtuning.html

[17] Sharma H., Shastri A., Biswas R. **"**Architecture of Automated Database Tuning Using SGA Parameters", *Database Systems Journal*, Romania, 2012

[18] Sharma H., Shastri A., Biswas R. "A Framework for Automated Database Tuning Using Dynamic SGA Parameters and Basic Operating System Utilities", *Database Systems 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/olaponline-analytical-processing-tools.html

Hitesh KUMAR SHARMA, The author is an Assistant Professor in University of Petroleum& Energy Studies, Dehradun. He has published 10+ research papers in National Journals and 20+ research papers in International Journal. Currently He is pursuing his Ph.D. in the area of database tuning**.**

Aditya SHASTRI, Ph.D. MIT, Published about 230+ research papers in international journals on Graph Theory with applications in Communication, Computer Graphics and Parallel Processing, Vice Chancellor, Director, Banasthali University, Banasthali, INDIA

Ranjit BISWAS, Head and Professor Jamia Hamdard (Hamdard University) Published about150+ research papers in International journals/bulletins.

Sanjeev KUMARSINGH: The author is an Associate Professor in Galgotias University, Noida. He has published 35+ research paper in International Journal and 15+ research papers in National Journals. He is Ph.D. in Mathematics.