SLOB

The Simple Database I/O Testing Toolkit for Oracle Database
Release 2.4.2

Table of Contents

Disclaimer 4

Background 4

SLOB Overview and Architecture 6

Understanding The SLOB Method 6

SLOB Data Block 6

Understanding Compression and SLOB 8

SLOB Block Read and Write Operations 8

SLOB Random Single Block Read (SELECT) Operations 8

SLOB Short Table Scan (SELECT) Operations 8

SLOB Block Write (Modify DML) Operations 9

The SLOB Schema Models 10

Understanding Single and Multiple Schema SLOB 10

Understanding SLOB Hot Spots and Hot Schema 13

SLOB Hot Spot 13

SLOB Hot Schema 15

SLOB Think Time 16

SLOB Key Performance Indicators / Throughput Metrics 17

SLOBops / SQL Executions / LIOPS (Logical I/O Per Second) 17

Using SLOB 19

Tablespace Requirement 19

Sys V IPC Semaphores 19

Database Creation Kit 21

Data Loading 21

The setup.sh Script 21

Example of Loading Multiple Schema Model 21

Example of Loading Single Schema Model 22

Performance Testing 24

The runit.sh Script 24

The runit.sh Script - Single Schema Model 24

The runit.sh Script – Multiple Schema Model 25

The runit.sh Script – Overriding slob.conf Settings 26

OS-Level Performance Data 27

SLOB Tunable Parameters 28

UPDATE_PCT 28

RUN_TIME 28

WORK_LOOP 28

SCALE 28

SCAN_PCT 28

SCAN_TABLE_SZ 29

WORK_UNIT 29

REDO_STRESS 30

LOAD_PARALLEL_DEGREE 30

THREADS_PER_SCHEMA 30

DO_HOTSPOT 30

HOTSPOT_MB 30

HOTSPOT_OFFSET_MB **31**
HOTSPOT_FREQUENCY **31**
HOT_SCHEMA_FREQUENCY **31**
THINK_TM_FREQUENCY **31**
THINK_TM_MIN **31**
THINK_TM_MAX **31**
DATABASE_STATISTICS_TYPE **32**
EXTERNAL_SCRIPT **32**
Example EXTERNAL_SCRIPT Use Cases 32
DBA_PRIV_USER **32**
Connecting to an Amazon RDS for Oracle Instance – An Example 33
SQL*Net Related Parameters **34**
ADMIN_SQLNET_SERVICE 34
SQLNET_SERVICE_BASE 34
SQLNET_SERVICE_MAX 35
SYSDBA_PASSWD 35
The awr_info.sh Script **36**
The awr_info.sh Script Columns – A Legend **37**
FILE 37
SESSIONS 37
ELAPSED 37
DB CPU 37
DB Tm 37
EXECUTES 37
LIO 37
PREADS 37
READ_MBS 37
PWRITES 37
WRITE_MBS 37
REDO_MBS 37
DFSR_LAT 37
DPR_LAT 38
DFPR_LAT 38
DFPW_LAT 38
LFPW_LAT 38
TOP WAIT 38
Advanced Topics **38**
Where To Get More Information **39**

Disclaimer

THIS DOCUMENTATION IS PROVIDED FOR REFERENCE PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS DOCUMENTATION, IT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY WHATSOEVER AND TO THE MAXIMUM EXTENT PERMITTED, PEAK PERFORMANCE AND KEVIN CLOSSON d.b.a PEAK PERFORMANCE SYSTEMS DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SAME. PEAK PERFORMANCE SYSTEMS SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, DIRECT, INDIRECT, CONSEQUENTIAL OR INCIDENTAL DAMAGES, ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS DOCUMENTATION OR ANY OTHER DOCUMENTATION. NOTWITHSTANDING ANYTHING TO THE CONTRARY, NOTHING CONTAINED IN THIS DOCUMENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM PEAK PERFORMANCE SYSTEMS AND KEVIN CLOSSON d.b.a PEAK PERFORMANCE SYSTEMS (OR ITS SUPPLIERS OR LICENSORS).

Background

SLOB is not a database benchmark. SLOB is an Oracle I/O workload generation tool kit. However, by force of habit, many SLOB users refer to SLOB with such terms as “benchmark”. SLOB aims to fill the gap between Orion and CALIBRATE_IO (neither generate legitimate database I/O as explained partly here) and full-function transactional **benchmarks** (such as Swingbench). Transactional workloads are intended to test the transactional capability of a database. SLOB users presume Oracle Database is a robust, ACID-complaint, transactional relational database and thus there is no need to “benchmark” it.

SLOB is not a benchmark kit.

SLOB embodies a *method* for testing hardware platforms to determine suitability for Oracle Database deployments requiring high performance I/O. SLOB is simple, yet powerful. Indeed, simple can be powerful! Consider steam (water vapor), for example. There is nothing complex about steam but when harnessed properly steam is a force to be reckoned with.

Powerful things can be simple and that is a good thing. This is why SLOB is a helpful tookit.

SLOB is simple to understand—and use.

SLOB drives Oracle Database to perform massive-scale *SQL execution* with minimal host CPU utilization. This characteristic is true whether testing for physical I/O capability or when the entire data set is cached in the SGA buffer pool.

In short, SLOB is a necessary tool for studying the underlying platform limits that often impact Oracle Database performance.

SLOB Overview and Architecture

Understanding The SLOB Method

At the heart of SLOB is the "SLOB method." The SLOB Method aims to test platforms without *application contention*. One cannot drive maximum hardware performance using application code that is, for example, bound by application locking or even sharing Oracle Database blocks. That's right—there is overhead when sharing data in data blocks! But SLOB—in its default deployment—is immune to such contention.

While the default SLOB deployment is free of application contention, SLOB *can* be configured to test with varying degrees of application contention if so desired.

SLOB Data Block

The basis of the default, contention-free behavior of SLOB is the sparse data block. Figure 1 depicts how SLOB places exactly a single row of data per data block.

As depicted in Figure 1, the default SLOB data consists of a row with a key column and a series of VARCHAR2(128) columns to take up space within the block.

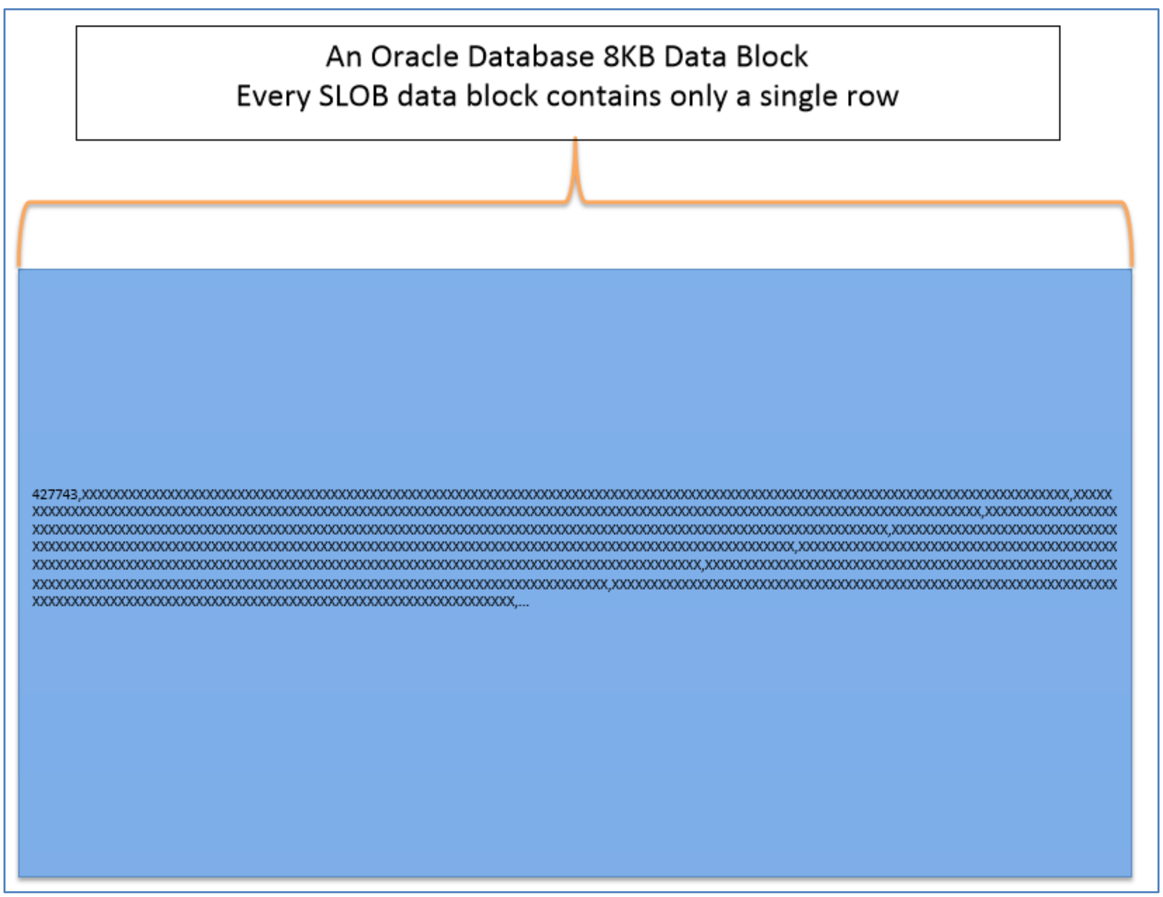

Figure 1: The Default Oracle Block Layout with SLOB

A default SLOB row is roughly 2KB so SLOB data blocks remain mostly in their initialized state (header and zeroes).

Understanding Compression and SLOB

Due to the *SLOB Method*, one should not use the default SLOB schema for testing compression technology. Simply put, default SLOB data compresses too deeply to be of any use in assessing compression technology.

SLOB Block Read and Write Operations

SLOB operations fall into both read and write categories.

SLOB Random Single Block Read (SELECT) Operations

During read operations, SLOB does not access the VARCHAR2 columns. Doing so would burden host CPU unnecessarily. Instead, SLOB simply locates the block (via an index on the key column) and performs the COUNT() aggregate on the second ordinal column. Performing COUNT(c2) on this schema forces a lightweight block access but not an expensive "row walk." SLOB only accesses the block header of the blocks containing the rows scoped by the SQL predicates. In pseudo code, SLOB performs read operations as follows:

```
SELECT COUNT(c2) FROM <slob table> WHERE <key> BETWEEN <random key> and <random key>.
```

As the pseudo code suggests there is a set of rows—and therefore blocks—being accessed on every execution of this statement. The SLOB test administrator can control how many blocks are accessed in each operation. While it's true that the blocks accessed in a single SLOB operation are logically adjacent in the SLOB *active data set*, they are guaranteed to be physically scattered across the tablespace containing the SLOB schema(s).

SLOB Short Table Scan (SELECT) Operations

Introduced in SLOB 2.4 is the ability to direct SLOB to perform short table scans. The method for controlling the percentage of all SELECT operations that perform a short table scan is centered on the slob.conf variable called SCAN_PCT.

The short table scan feature establishes that a percentage of all SELECTs will perform a full table scan on a short table. The size of the table is governed by the SCAN_TABLE_SZ parameter in slob.conf. An example of a SLOB test, that incorporates short table scans, would be to set SHORT_TABLE_SZ to the value 1M (1 megabyte). When loading SLOB, the data loader (setup.sh) will therefore load a 1 megabyte short table for every schema being loaded. Then, during the performance test, one might set UPDATE_PCT to 20 leaving 80% of the SLOB operations to be SELECT statements. Further, one could set SCAN_PCT to 50. With these settings, SLOB will perform 20% UPDATEs and 80% SELECTs. Because SCAN_PCT is set to 50, half of the SELECT statements will be random access (*db file sequential read*) and the other half will be short scans (*direct path read* or *db file scattered read* depending on environmental factors like SGA buffer pool capacity, etc).

For more information please see the specific sections below covering these slob.conf parameters.

SLOB Block Write (Modify DML) Operations

For write operations SLOB uses an UPDATE statement with the same method of selectivity as the SELECT statement above. The UPDATE statement manipulates only the non-indexed columns so as to avoid index maintenance overhead. As described later in this document the SLOB test administrator can chose varying degrees of data modification intensity for each SQL execution (a.k.a., SLOBop) by directing SLOB to manipulate more, or fewer, columns in each operation. Changing the VARCHAR columns allows for the generation of redo logging while not suffering the overhead of maintaining indexes.

The SLOB Schema Models

SLOB allows the test administrator to choose between two schema models:

- **SLOB Multiple Schema.** One can think of SLOB Multiple Schema as a form of multitenant architecture. As described above, this manner of SLOB testing ensures there is no application data sharing. With SLOB Multiple Schema the test administrator dictates how many SLOB threads (Oracle Database sessions) will connect to the instance and perform SLOB operations against their respective schema. The default is a single SLOB thread per schema. The SLOB test administrator can choose to load large numbers of schemas but then test only subsets of schemas or, indeed, the entire set of schemas.
- **SLOB Single Schema.** As a slight variation to the SLOB Method, the test administrator can choose to deploy a single SLOB schema. In this model multiple SLOB threads (Oracle Database sessions) perform SLOB operations on the same schema. This manner of SLOB testing introduces shared data contention at the block level. There is nothing that would prevent two SLOB threads (Oracle Database sessions) from selecting the same blocks of data in two concurrent SLOB operations. Of course the larger the active data set the lower the frequency of such shared data contention.

Understanding Single and Multiple Schema SLOB

Single Schema Model

SLOB threads (Oracle Database sessions) operating in the Single Schema model run the risk of visiting the same data. When testing the SLOB Single Schema model one might start to see wait events such as *read by other session*. When events like *read by other session* are raised it indicates CPU cycles spent that did not result in a physical I/O—as would be the case in the strict SLOB Method. This impacts the physical I/O per DB CPU achievable by the platform being tested.

When sharing data in the Single Schema model a session may spend processor cycles only to find out that another session is already reading the block of data and thus the session goes to sleep on a *read by other session* wait event to be notified when the block has been installed in the SGA buffer pool. This style of testing might be desirable and, indeed, the odds of choosing the same data between SLOB threads is a factor of how many threads are running and how large the SLOB active data set is.

Figure 2 shows a depiction of Single Schema SLOB residing in a tablespace that is larger than the SLOB active data set. As the graphic shows, the active data set in this case is 500GB as per the slob.conf setting of the SCALE parameter. More information will be given later on slob.conf parameters.

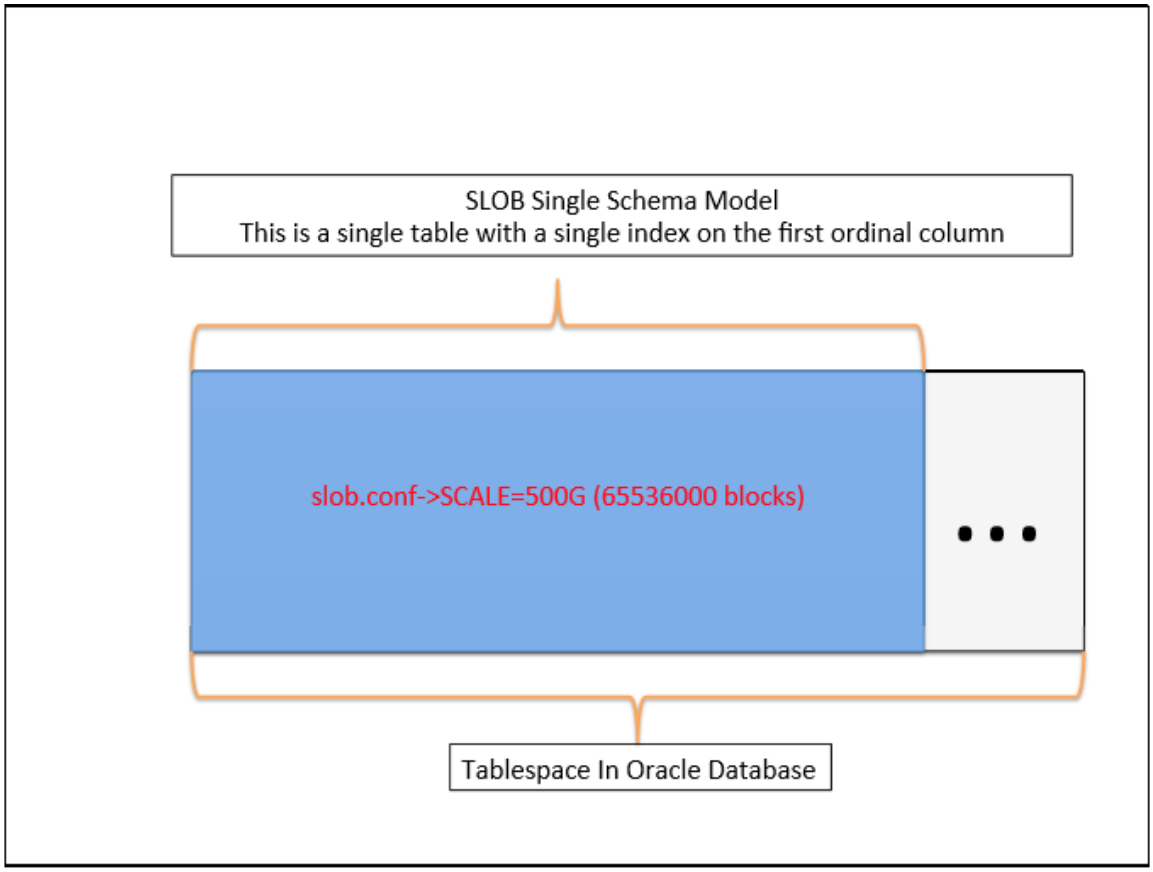

Figure 2: SLOB Single Schema Model

Figure 3 depicts random physical locations of SLOB blocks. In the example, SLOB block 4201 would have the row with the key value 4201 in it. This diagram helps one understand how it is that SLOB drives such a dramatically random block access pattern. As explained above, SLOB chooses a range of blocks upon which to perform a SLOB operation. Consider an example where the low-bound key is, say, 4201 and the high-bound key is 4457 (a slob.conf->WORK_UNIT of 64) the blocks would be scattered across a vast logical and physical area within the tablespace.

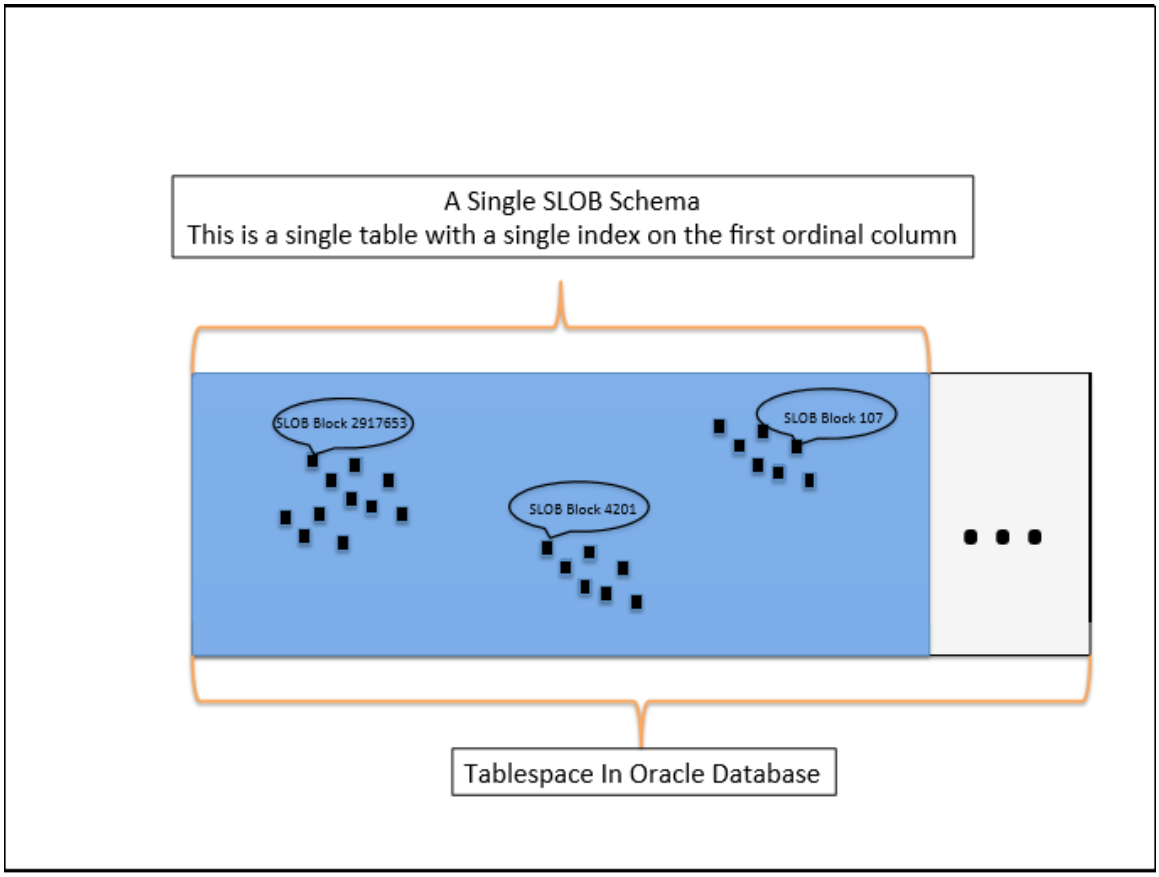

Figure 3: SLOB Data Random Block Locations

Multiple Schema Model

Figure 4 shows a depiction of the SLOB Multiple Schema model. Each of these schemas (1 through 12 shown) is exactly the same, architecturally speaking, as a Single Schema as shown in Figure 2—just smaller.

SLOB allows the test administrator to load up to 4096 schemas in the Multiple Schema model. The test administrator also decides how many SLOB threads (Oracle Database sessions) to connect to each schema during a SLOB test. It is not necessary to test all schemas. For example, the test administrator can choose to load, say, 128 schemas and then, if so desired, test with *64* schemas.

As Figure 4 shows this manner of SLOB testing is a form of multitenant testing. Each schema has its own sessions attached but all share Oracle Database instance background processing such as Database Writer and Log Writer.

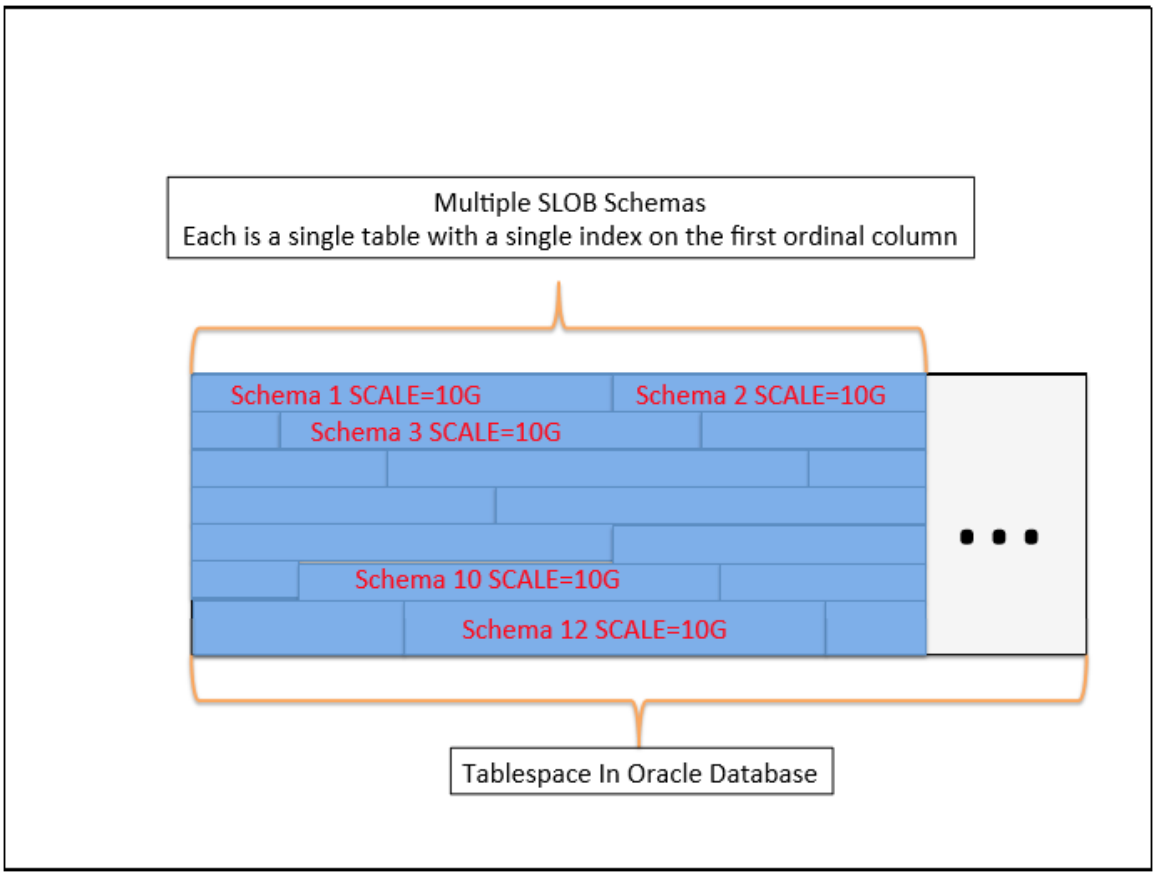

Figure 4: SLOB Multiple Schema Model

Understanding SLOB Hot Spots and Hot Schema

SLOB Hot Spot

As explained above, SLOB default block accesses are completely random. However, under some testing scenarios it is desirable to focus the SLOB read and write activity to a subset of the data—a Hot Spot. Simply put, a SLOB Hot Spot is a subset of the blocks in a SLOB Schema. Generally, a SLOB Host Spot is a small fraction of the total size of a SLOB Schema.

SLOB Hot Spot functionality is supported with both SLOB Single Schema and SLOB Multiple Schema mode.

A SLOB Hot Spot is configured by the SLOB test administrator as a subset range of blocks (expressed in megabytes) per SLOB schema. For example, the administrator might choose to load 1 terabyte into the SLOB Single Schema model and further configure a SLOB Hot Spot of only, say, 1 gigabyte. Finally, the administrator then would choose the Hot Spot *frequency*. SLOB Hot Spot frequency is configured as every *Nth* SLOB operation. If, for example, the administrator configures a Hot Spot frequency of 3 then every 3rd SLOB operation (read or write) would fall into the

range of blocks in the SLOB Hot Spot. SLOB Hot Spot functionality is the same whether SLOB Multiple Schema or SLOB Single Schema is being tested. SLOB Hot Spot related tunable parameters are covered in-depth in the SLOB Tunable Parameter section of this document.

Figures 5 and 6 depict SLOB Hot Spot in Single Schema mode and Multiple Schema mode respectively.

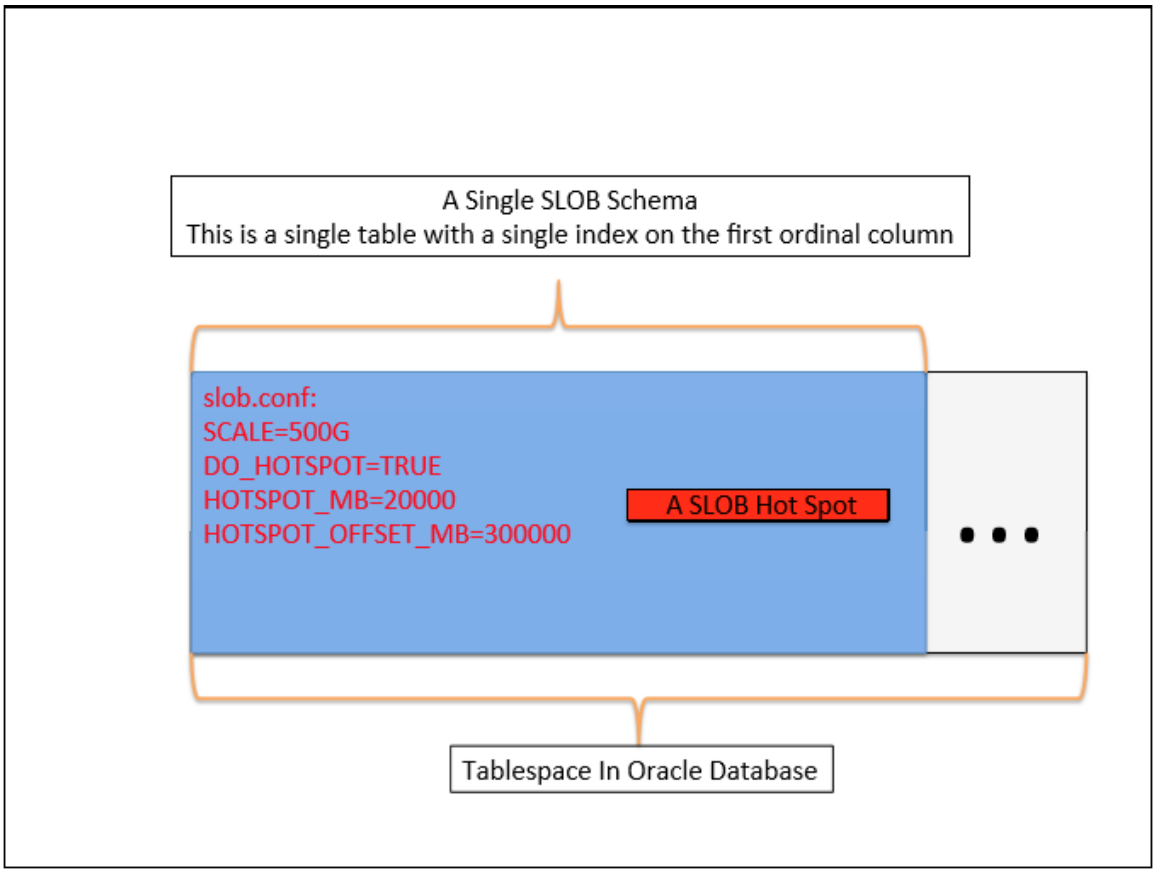

Figure 5: SLOB Single Schema Model with Hot Spot

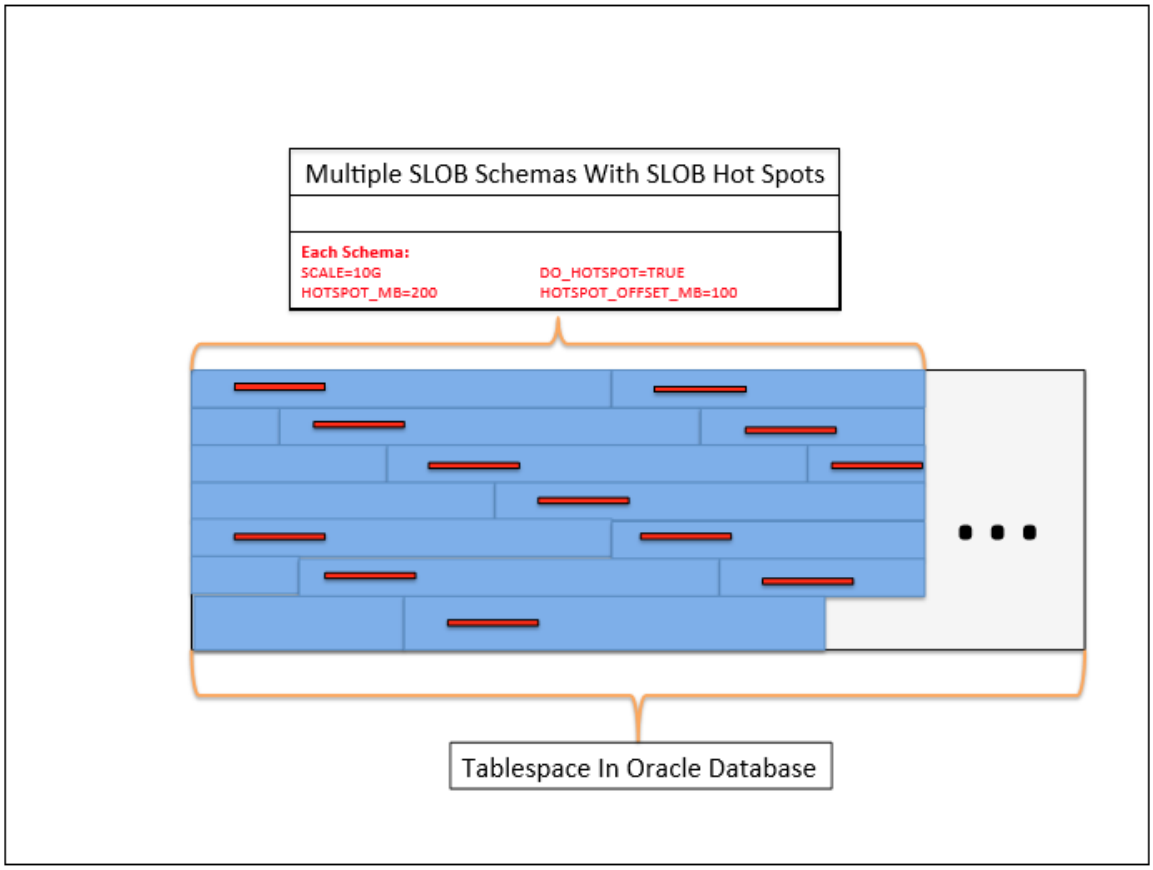

Figure 6: SLOB Multiple Schema Model with Hot Spot

SLOB Hot Schema

By default, SLOB is free of application contention. However, if so desired the administrator can choose to introduce application contention to a SLOB test through the use of the SLOB Hot Schema functionality. SLOB Hot Schema only makes sense in SLOB Multiple Schema mode.

When SLOB Hot Schema is enabled, every *Nth* SLOB operation (read/write) occurs on the SLOB schema owned by the first SLOB schema created—user1 (created in the database with the GRANT statement). This style of SLOB testing introduces a significant amount of application-level contention. Moreover, in Real Application Clusters (RAC) testing situations, SLOB Hot Schema places a significant load on the RAC interconnect.

SLOB Hot Schema and SLOB Hot Spot can be used in combination to further vary the level of contention being tested.

The SLOB tunable parameters related to Hot Schema are covered in the SLOB Tunable Parameter section of this document.

Figure 7 depicts SLOB in Single Schema mode with both Hot Schema and Hot Spot functionality in use.

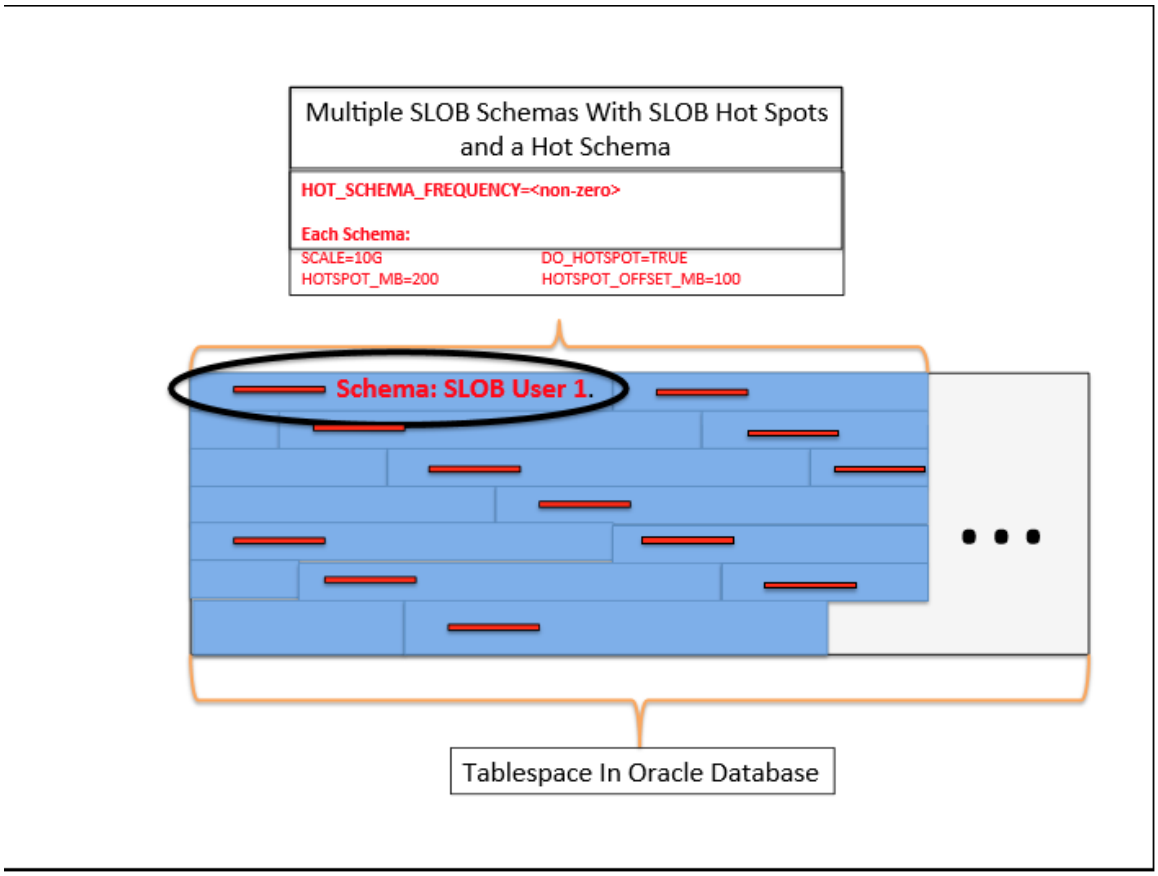

Figure 7: SLOB Multiple Scheme Model with Hot Spot and Hot Schema

SLOB Think Time

By default SLOB threads perform SLOB operations in a loop without any pauses. Pauses simulate human user think-time. The value of testing SLOB with think time is it allows for very large numbers of sessions connected to the database instance without saturating the system. Large numbers of sessions place interesting demands on the platform by way of scheduling and memory management.

To test with SLOB think time the SLOB test administrator chooses the minimum and maximum think time as well as the frequency at which think times will be injected into every SLOB thread's work loop.

An example of SLOB with think time is, for instance, 1024 SLOB threads (database sessions) executing SLOB operations with think times averaging between .05 and .5 seconds at a frequency interval of 3. As such every third SLOB operation will sleep

for a random value between .05 and .5 seconds. The slob.conf parameters for tuning SLOB think time are covered in the SLOB Tunable Parameter section.

The default is no SLOB think-time.

SLOB Key Performance Indicators / Throughput Metrics

The key performance indicator (KPI) for SLOB is not transactions per time-unit such as Transactions per Second (TPS) or Transactions per Minute (TPM). These KPI are reserved for transactional workloads. Oracle is obviously a very capable transaction engine. If one wishes to test Oracle transactional performance-as an indicator of platform suitability for one's application—then the only accurate approach is to test one's application.

Testing arbitrary transactional code is not an indicator of what a platform will sustain for any workload other than the synthetic one being tested. On the other hand, a non-transactional workload such as SLOB will offer clear indication of what the platform can sustain in such key areas as transaction logging throughput and random block I/O while showing how much processor bandwidth remains to accommodate one's own application code. By testing cached SLOB, one can get a very clear idea how well the platform handles critical scalability underpinnings such as logical I/O—after all, a platform that can't scale cache-hits cannot scale cache misses and thus physical I/O.

SLOB performance is generally expressed in the following terms:

- IOPS
- SLOBops
- SQL Executions
- Logical I/O Per Second

SLOBops / SQL Executions / LIOPS (Logical I/O Per Second)

SLOB workload metrics are:

- **Physical I/O Testing**
  - SQL Executions.
  - Physical IOPS (a.k.a., PIOPS)
- **Logical I/O Testing**
  - Database Logical I/O per second (a.k.a., LIOPS). A Logical I/O is a cache hit in the SGA buffer pool of the Oracle instance.

Physical I/O Per Second or SQL Executions

The metric of throughput is based on the type of testing being conducted by the SLOB test administrator. Consider, for example, how to express SLOB results in the case of testing cached SLOB.

The Load Profile in Figure 8 was taken from a test of cached SLOB on a two-socket (2s20c40t) Xeon based server. One could express this test result as either 47,200 sqlexec/s or 12,199,141 Logical I/O per second using the term *LIOPS*.

Load Profile	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	19.6	60.1	0.00	5.11
DB CPU(s):	19.5	59.9	0.00	5.10
Background CPU(s):	0.1	0.3	0.00	0.00
Redo size (bytes):	36,969.2	113,338.3		
Logical read (blocks):	12,199,141.1	37,399,516.7		
Block changes:	107.0	327.9		
Physical read (blocks):	0.9	2.8		
Physical write (blocks):	0.2	0.6		
Read IO requests:	0.9	2.8		
Write IO requests:	0.1	0.3		
Read IO (MB):	0.0	0.0		
Write IO (MB):	0.0	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:				
RAC GC blocks received:	4.2	12.8		
RAC GC blocks served:	5.2	16.0		
User calls:	3.8	11.8		
Parses (SQL):	9.0	27.6		
Hard parses (SQL):	0.1	0.2		
SQL Work Area (MB):	0.5	1.4		
Logons:	0.4	1.2		
Executes (SQL):	47,200.7	144,705.7		
Rollbacks:	0.0	0.0		
Transactions:	0.3			

Figure 8: Oracle AWR Report Example. Cached SLOB.

Figure 9 shows the Load Profile of the same server performing physical I/O after decreasing the size of the SGA buffer pool. The SLOB test administrator in this case could express these results as either the sum of physical read and write I/O requests per second (116,943 + 24,556 = 141,499 PIOPS) or 583.5 sqlexec/s.

Load Profile	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	61.9	0.6	0.11	5.17
DB CPU(s):	14.2	0.1	0.02	1.18
Background CPU(s):	2.1	0.0	0.00	0.00
Redo size (bytes):	19,242,267.9	172,970.5		
Logical read (blocks):	142,071.0	1,277.1		
Block changes:	59,509.9	534.9		
Physical read (blocks):	117,161.7	1,053.2		
Physical write (blocks):	28,267.2	254.1		
Read IO requests:	116,943.4	1,051.2		
Write IO requests:	24,556.6	220.7		
Read IO (MB):	915.3	8.2		
Write IO (MB):	220.8	2.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:				
RAC GC blocks received:	339.5	3.1		
RAC GC blocks served:	5.4	0.1		
User calls:	12.0	0.1		
Parses (SQL):	25.7	0.2		
Hard parses (SQL):	5.2	0.1		
SQL Work Area (MB):	1.5	0.0		
Logons:	1.3	0.0		
Executes (SQL):	583.5	5.3		
Rollbacks:	0.0	0.0		
Transactions:	111.3			

Figure 9: Oracle AWR Report Load Profile. SLOB Physical I/O.

In general, folks in the SLOB user community tend to express results in either LIOPS or PIOPS terms.

Using SLOB

Tablespace Requirement

The SLOB setup script (setup.sh) requires a tablespace with sufficient space to contain the SLOB schemas. The SLOB test administrator can chose to create a tablespace in a file system or ASM.

The default space requirement is roughly 15 gigabytes for setup.sh plus any ancillary overhead in TEMP segments for such purposes as sort spill. Naturally, UNDO segment space will be required.

There is more information on setup.sh later in this document.

Sys V IPC Semaphores

The SLOB "trigger kit" requires a single SYS V IPC semaphore set that contains a single semaphore.

Database Creation Kit

The SLOB kit provides a simple database creation kit under the SLOB/misc/create_database_kit directory. The easiest way to get started with SLOB is to either a) provision storage for a file system and use the Database Creation Kit, or b) use an existing **test database** with a special-purpose tablespace allocated for SLOB.

For more information on the Database Creation Kit please see the README file in the SLOB/misc/create_database_kit directory.

Data Loading

The setup.sh Script

The setup.sh script is used to load SLOB data. The script takes two mandatory options:

- Option 1. Tablespace into which SLOB will create and load the test schemas
- Option 2. The number of schemas to create and load

The setup.sh script takes directives from slob.conf to control the number of database blocks to load in each schema (primary table and short-scan table) and the degree of concurrent loading used during data loading. Please refer to the section entitled SLOB Tunable Parameters for more information on parameters related to data-loading.

Example of Loading Multiple Schema Model

Figure 10 shows an example of loading 2 schemas with 1 gigabyte each into the primary SLOB table and 1 megabyte into each schema's short scan table using concurrent loading with 4 data loaders.

```
[$
[$ sh ./setup.sh IOPS 2
NOTIFY  : 2017.02.09-09:39:38 :
NOTIFY  : 2017.02.09-09:39:38 : Begin SLOB setup.
NOTIFY  : 2017.02.09-09:39:38 : Load parameters from slob.conf:

SCALE: 1G (131072 blocks)
SCAN_TABLE_SZ: 1M (128 blocks)
LOAD_PARALLEL_DEGREE: 4
ADMIN_SQLNET_SERVICE: ""
SQLNET_SERVICE_BASE: ""

Connect strings to be used:
ADMIN_CONNECT_STRING: "/ as sysdba"
NON_ADMIN_CONNECT_STRING: " "

NOTIFY  : 2017.02.09-09:39:38 : Testing connectivity to the instance to validate slob.conf settings
NOTIFY  : 2017.02.09-09:39:38 : Testing Admin connect using "/ as sysdba"
NOTIFY  : 2017.02.09-09:39:38 : Successful test connection: "sqlplus -L / as sysdba"
NOTIFY  : 2017.02.09-09:39:38 : Dropping prior SLOB schemas. This may take a while if there is a large number of old schemas.
NOTIFY  : 2017.02.09-09:39:40 : Deleted 2 SLOB schema(s).
NOTIFY  : 2017.02.09-09:39:40 : Previous SLOB schemas have been removed
NOTIFY  : 2017.02.09-09:39:40 : Preparing to load 2 schema(s) into tablespace: IOPS
NOTIFY  : 2017.02.09-09:39:40 : Loading user1 schema
NOTIFY  : 2017.02.09-09:40:12 : Finished loading, indexing and gathering statistics on user1 schema in 32 seconds
NOTIFY  : 2017.02.09-09:40:12 : Commencing multiple, concurrent schema creation and loading
NOTIFY  : 2017.02.09-09:40:12 : Waiting for background batch 1. Loading up to user2
NOTIFY  : 2017.02.09-09:40:55 : Completed concurrent data loading phase: 43 seconds
NOTIFY  : 2017.02.09-09:40:55 : Creating SLOB procedure
NOTIFY  : 2017.02.09-09:40:55 : SLOB procedure created
NOTIFY  : 2017.02.09-09:40:55 : Row and block counts for SLOB table(s) reported in ./slob_data_load_summary.txt
NOTIFY  : 2017.02.09-09:40:55 : Please examine ./slob_data_load_summary.txt for any possbile errors
NOTIFY  : 2017.02.09-09:40:55 :
NOTIFY  : 2017.02.09-09:40:55 : NOTE: No errors detected but if ./slob_data_load_summary.txt shows errors then
NOTIFY  : 2017.02.09-09:40:55 : examine /home/oracle/2.4/LATEST/PACK/SLOB/cr_tab_and_load.out

NOTIFY  : 2017.02.09-09:40:55 : SLOB setup complete. Total setup time:  (77 seconds)
NOTIFY  : 2017.02.09-09:40:55 : Please do not forget to compile the wait kit.
NOTIFY  : 2017.02.09-09:40:55 : Please change directories to ./wait_kit and execute make(1).
NOTIFY  : 2017.02.09-09:40:55 : Example:
NOTIFY  : 2017.02.09-09:40:55 :  $ cd ./wait_kit
NOTIFY  : 2017.02.09-09:40:55 :  $ make
[$
[$ egrep 'SCALE|SZ|DEGREE' slob.conf
SCALE=1G
SCAN_TABLE_SZ=1M
LOAD_PARALLEL_DEGREE=4
$
```

Figure 10: SLOB Multiple Schema Model – Data Loading

Example of Loading Single Schema Model

Figure 11 shows an example of loading 2 gigabytes into SLOB Single Schema Model using parallelism of 4. Additionally, 1 megabyte will be loaded into the short scan table since SCAN_TABLE_SZ is set to the default size of 1 megabyte.

```
[$
[$ egrep 'SCALE|SZ|DEGREE' slob.conf
SCALE=2G
SCAN_TABLE_SZ=1M
LOAD_PARALLEL_DEGREE=4
[$
[$ sh ./setup.sh IOPS 1
NOTIFY  : 2017.02.09-09:55:18 :
NOTIFY  : 2017.02.09-09:55:18 : Begin SLOB setup.
NOTIFY  : 2017.02.09-09:55:18 : Load parameters from slob.conf:

SCALE: 2G (262144 blocks)
SCAN_TABLE_SZ: 1M (128 blocks)
LOAD_PARALLEL_DEGREE: 4
ADMIN_SQLNET_SERVICE: ""
SQLNET_SERVICE_BASE: ""

Connect strings to be used:
ADMIN_CONNECT_STRING: "/ as sysdba"
NON_ADMIN_CONNECT_STRING: " "

NOTIFY  : 2017.02.09-09:55:18 : Testing connectivity to the instance to validate slob.conf settings
NOTIFY  : 2017.02.09-09:55:18 : Testing Admin connect using "/ as sysdba"
NOTIFY  : 2017.02.09-09:55:18 : Successful test connection: "sqlplus -L / as sysdba"
NOTIFY  : 2017.02.09-09:55:18 : Dropping prior SLOB schemas. This may take a while if there is a large number of old schemas.
NOTIFY  : 2017.02.09-09:55:19 : Deleted 0 SLOB schema(s).
NOTIFY  : 2017.02.09-09:55:19 : Previous SLOB schemas have been removed
NOTIFY  : 2017.02.09-09:55:19 : Preparing to load 1 schema(s) into tablespace: IOPS
NOTIFY  : 2017.02.09-09:55:19 : Loading user1 schema
NOTIFY  : 2017.02.09-09:56:24 : Finished loading, indexing and gathering statistics on user1 schema in 65 seconds
NOTIFY  : 2017.02.09-09:56:24 : Creating SLOB procedure
NOTIFY  : 2017.02.09-09:56:24 : SLOB procedure created
NOTIFY  : 2017.02.09-09:56:24 : Row and block counts for SLOB table(s) reported in ./slob_data_load_summary.txt
NOTIFY  : 2017.02.09-09:56:24 : Please examine ./slob_data_load_summary.txt for any possbile errors
NOTIFY  : 2017.02.09-09:56:24 :
NOTIFY  : 2017.02.09-09:56:24 : NOTE: No errors detected but if ./slob_data_load_summary.txt shows errors then
NOTIFY  : 2017.02.09-09:56:24 : examine /home/oracle/2.4/LATEST/PACK/SLOB/cr_tab_and_load.out

NOTIFY  : 2017.02.09-09:56:24 : SLOB setup complete. Total setup time:  (66 seconds)
NOTIFY  : 2017.02.09-09:56:24 : Please do not forget to compile the wait kit.
NOTIFY  : 2017.02.09-09:56:24 : Please change directories to ./wait_kit and execute make(1).
NOTIFY  : 2017.02.09-09:56:24 : Example:
NOTIFY  : 2017.02.09-09:56:24 :  $ cd ./wait_kit
NOTIFY  : 2017.02.09-09:56:24 :  $ make
$
```

Figure 11: SLOB Single Schema Model – Data Loading

Performance Testing

The runit.sh Script

The runit.sh script is the performance test driver. By default, it uses parameter settings in slob.conf. However, one can override certain slob.conf settings with command line options if so desired. Please refer to the section entitled SLOB Tunable Parameters for more information on slob.conf parameters.

Figure 12 shows the help text for runit.sh.

```
$
$ sh ./runit.sh nonsense-options-here
NOTIFY  : 2017.02.09-09:59:01 : For security purposes all file and directory creation and deletions
NOTIFY  : 2017.02.09-09:59:01 : performed by ./runit.sh are logged in file: /home/oracle/2.4/LATEST/PACK/SLOB/.file_operations_audit_trail.out.
NOTIFY  : 2017.02.09-09:59:01 : SLOB TEMPDIR is /tmp/.SLOB.2017.02.09.095901. SLOB will automatically delete this directory after this invocation of ./runit.sh.
FATAL   : 2017.02.09-09:59:01 : Single Option Invocation requires one option: A non-zero integer.
FATAL   : 2017.02.09-09:59:01 : Invalid command line. Abort.

./runit.sh supports the following command usage:

1. Single Option Invocation.
       $ sh ./runit.sh <number-of-SLOB-schemas-to-test>

2. Multiple Option Invocation
   2.1 This invocation style requires *exactly* four options.
       $ sh ./runit.sh -s <number-of-slob-schemas-to-test> -t <SLOB-threads-per-schema>

NOTE: With Single Option Invocation slob.conf->THREADS_PER_SCHEMA is used. If you
      want more than a single SLOB thread per schema set THREADS_PER_SCHEMA in slob.conf.
      The default setting for slob.conf->THREADS_PER_SCHEMA is 1.

      With Multiple Option Invocation slob.conf->THREADS_PER_SCHEMA is overridden.
      The number of SLOB threads per schema is taken from the argument passed
      in with the -t option.

EXAMPLES:

      Example 1. 256 SLOB schemas each with slob.conf->THREADS_PER_SCHEMA number
      of SLOB threads per schema:
        $ sh ./runit.sh 256

      Example 2. 16 SLOB schemas each with 32 SLOB threads:
        $ sh ./runit.sh -s 16 -t 32

NOTE: Example 2 produces 512 (16*32) Oracle Database sessions.

ADDITIONAL INFORMATION: The SLOB documentation at kevinclosson.net/slob or SLOB/doc


FATAL   : 2017.02.09-09:59:01 : Aborting execution. Cleaning up SLOB temporary directory (/tmp/.SLOB.2017.02.09.095901).
$
```

Figure 12: SLOB runit.sh Help Output.

The runit.sh Script - Single Schema Model

Figure 13 shows a screen shot of testing Single Schema SLOB. As the image shows the number of SLOB threads is determined by the slob.conf setting of THREADS_PER_SCHEMA which was set to 8 in the example.

```
$
$ grep THREADS slob.conf
THREADS_PER_SCHEMA=8
$
$ sh ./runit.sh 1
NOTIFY  : 2017.02.09-10:22:51 : For security purposes all file and directory creation and deletions
NOTIFY  : 2017.02.09-10:22:51 : performed by ./runit.sh are logged in file: /home/oracle/2.4/LATEST/PACK/SLOB/.file_operations_audit_trail.out.
NOTIFY  : 2017.02.09-10:22:51 : SLOB TEMPDIR is /tmp/.SLOB.2017.02.09.102251. SLOB will automatically delete this directory after this invocation of ./runit.sh.
NOTIFY  : 2017.02.09-10:22:51 : Sourcing in slob.conf
NOTIFY  : 2017.02.09-10:22:51 : Clearing files from previous SLOB testing.
NOTIFY  : 2017.02.09-10:22:51 :
NOTIFY  : 2017.02.09-10:22:51 : Conducting SLOB pre-test checks...

UPDATE_PCT: 25
SCAN_PCT: 0
RUN_TIME: 300
WORK_LOOP: 0
SCALE: 2G (262144 blocks)
WORK_UNIT: 64
REDO_STRESS: LITE
HOT_SCHEMA_FREQUENCY: 0
DO_HOTSPOT: FALSE
HOTSPOT_MB: 8
HOTSPOT_OFFSET_MB: 16
HOTSPOT_FREQUENCY: 3
THINK_TM_FREQUENCY: 0
THINK_TM_MIN: .1
THINK_TM_MAX: .5
DATABASE_STATISTICS_TYPE: statspack
THREADS_PER_SCHEMA: 8

ADMIN_SQLNET_SERVICE: ""
SQLNET_SERVICE_BASE: ""
SQLNET_SERVICE_MAX: ""

EXTERNAL_SCRIPT: ""

Connect strings to be used:
admin_connect_string: "/ as sysdba"
non_admin_connect_string: ""
admin_conn: "sqlplus -L / as sysdba"

NOTIFY  : 2017.02.09-10:22:51 : Testing SYSDBA connectivity to the instance to validate slob.conf settings.
NOTIFY  : 2017.02.09-10:22:51 : Testing connectivity. Command: "sqlplus -L / as sysdba".
NOTIFY  : 2017.02.09-10:22:51 : Creating (or replacing) SLOB BFILE (SLOBDIR) directory under /tmp.
NOTIFY  : 2017.02.09-10:22:51 : Testing connectivity. Command: "sqlplus -L user1/user1".
NOTIFY  : 2017.02.09-10:22:52 : Performing redo log switch.
NOTIFY  : 2017.02.09-10:22:52 : Redo log switch complete.
NOTIFY  : 2017.02.09-10:22:52 : Setting up trigger mechanism.
NOTIFY  : 2017.02.09-10:23:02 : Running iostat, vmstat and mpstat on current host--in background.
NOTIFY  : 2017.02.09-10:23:02 : Connecting 8 (THREADS_PER_SCHEMA) session(s) to 1 schema(s) ...
NOTIFY  : 2017.02.09-10:23:02 :
NOTIFY  : 2017.02.09-10:23:02 : Executing statspack "before snap" procedure. Connect string is "sqlplus -S -L / as sysdba".
NOTIFY  : 2017.02.09-10:23:04 : Before statspack snap ID is 21
NOTIFY  : 2017.02.09-10:23:04 :
NOTIFY  : 2017.02.09-10:23:04 : Test has been triggered. Processes are executing.
NOTIFY  : 2017.02.09-10:23:04 : List of monitored sqlplus PIDs written to /tmp/.SLOB.2017.02.09.102251/21973.f_wait_pids.out.
NOTIFY  : 2017.02.09-10:23:14 : Waiting for 287 seconds before monitoring running processes (for exit).
NOTIFY  : 2017.02.09-10:28:01 : Entering process monitoring loop.
NOTIFY  : 2017.02.09-10:28:05 : Run time 301 seconds.
NOTIFY  : 2017.02.09-10:28:05 : Executing statspack "after snap" procedure. Connect string is "sqlplus -S -L / as sysdba".
NOTIFY  : 2017.02.09-10:28:06 : After statspack snap ID is 22
NOTIFY  : 2017.02.09-10:28:07 : Terminating background data collectors.
./runit.sh: line 1488:  6082 Killed                  ( iostat -xm 3 > iostat.out 2>&1 )
NOTIFY  : 2017.02.09-10:28:07 :
NOTIFY  : 2017.02.09-10:28:07 : SLOB test is complete.
NOTIFY  : 2017.02.09-10:28:07 : Cleaning up SLOB temporary directory (/tmp/.SLOB.2017.02.09.102251).
$
```

Figure 13: SLOB Single Schema Testing With slob.conf Thread Count Control.

The runit.sh Script – Multiple Schema Model

Figure 14 shows an example of Multiple Schema SLOB. Here, again, the number of SLOB threads is being controlled by the slob.conf value assigned to THREADS_PER_SCHEMA which was 3 for the sake of example.

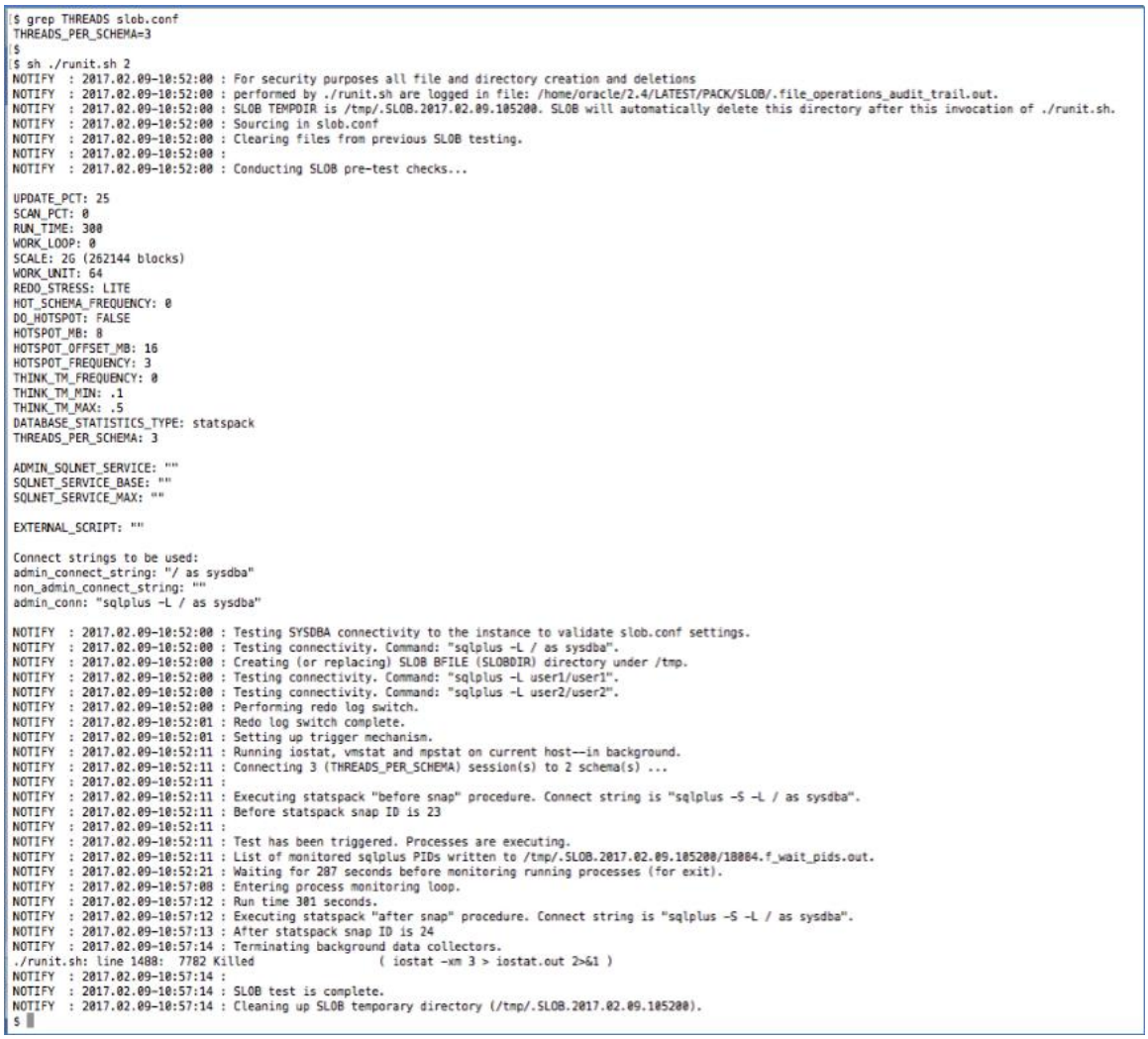

Figure 14: SLOB Multiple Schema Testing With slob.conf Thread Count Control.

The runit.sh Script – Overriding slob.conf Settings

Figure 15 shows an example of overriding slob.conf settings. As the image shows the slob.conf setting for THREADS_PER_SCHEMA was 3. However, with the use of runit.sh Multiple Option Invocation the slob.conf setting for SLOB threads was overridden. As the image shows runit.sh connected 5 SLOB threads (Oracle Database sessions) each to 2 SLOB schemas.

```
$ grep THREADS slob.conf
THREADS_PER_SCHEMA=3
$ sh ./runit.sh -s 2 -t 5
NOTIFY  : 2017.02.09-11:05:01 : For security purposes all file and directory creation and deletions
NOTIFY  : 2017.02.09-11:05:01 : performed by ./runit.sh are logged in file: /home/oracle/2.4/LATEST/PACK/SLOB/.file_operations_audit_trail.out.
NOTIFY  : 2017.02.09-11:05:01 : SLOB TEMPDIR is /tmp/.SLOB.2017.02.09.110501. SLOB will automatically delete this directory after this invocation of ./runit.sh.
NOTIFY  : 2017.02.09-11:05:01 : Sourcing in slob.conf
NOTIFY  : 2017.02.09-11:05:01 : Clearing files from previous SLOB testing.
NOTIFY  : 2017.02.09-11:05:01 :
NOTIFY  : 2017.02.09-11:05:01 : Conducting SLOB pre-test checks...

UPDATE_PCT: 25
SCAN_PCT: 0
RUN_TIME: 300
WORK_LOOP: 0
SCALE: 2G (262144 blocks)
WORK_UNIT: 64
REDO_STRESS: LITE
HOT_SCHEMA_FREQUENCY: 0
DO_HOTSPOT: FALSE
HOTSPOT_MB: 8
HOTSPOT_OFFSET_MB: 16
HOTSPOT_FREQUENCY: 3
THINK_TM_FREQUENCY: 0
THINK_TM_MIN: .1
THINK_TM_MAX: .5
DATABASE_STATISTICS_TYPE: statspack
THREADS_PER_SCHEMA: 5 (-t option)

ADMIN_SQLNET_SERVICE: ""
SQLNET_SERVICE_BASE: ""
SQLNET_SERVICE_MAX: ""

EXTERNAL_SCRIPT: ""

Connect strings to be used:
admin_connect_string: "/ as sysdba"
non_admin_connect_string: ""
admin_conn: "sqlplus -L / as sysdba"

NOTIFY  : 2017.02.09-11:05:01 : Testing SYSDBA connectivity to the instance to validate slob.conf settings.
NOTIFY  : 2017.02.09-11:05:01 : Testing connectivity. Command: "sqlplus -L / as sysdba".
NOTIFY  : 2017.02.09-11:05:02 : Creating (or replacing) SLOB BFILE (SLOBDIR) directory under /tmp.
NOTIFY  : 2017.02.09-11:05:02 : Testing connectivity. Command: "sqlplus -L user1/user1".
NOTIFY  : 2017.02.09-11:05:02 : Testing connectivity. Command: "sqlplus -L user2/user2".
NOTIFY  : 2017.02.09-11:05:02 : Performing redo log switch.
NOTIFY  : 2017.02.09-11:05:02 : Redo log switch complete.
NOTIFY  : 2017.02.09-11:05:02 : Setting up trigger mechanism.
NOTIFY  : 2017.02.09-11:05:12 : Running iostat, vmstat and mpstat on current host--in background.
NOTIFY  : 2017.02.09-11:05:12 : Connecting 5 (THREADS_PER_SCHEMA) session(s) to 2 schema(s) ...
NOTIFY  : 2017.02.09-11:05:12 :
NOTIFY  : 2017.02.09-11:05:12 : Executing statspack "before snap" procedure. Connect string is "sqlplus -S -L / as sysdba".
NOTIFY  : 2017.02.09-11:05:13 : Before statspack snap ID is 25
NOTIFY  : 2017.02.09-11:05:13 :
NOTIFY  : 2017.02.09-11:05:13 : Test has been triggered. Processes are executing.
NOTIFY  : 2017.02.09-11:05:13 : List of monitored sqlplus PIDs written to /tmp/.SLOB.2017.02.09.110501/24418.f_wait_pids.out.
NOTIFY  : 2017.02.09-11:05:23 : Waiting for 287 seconds before monitoring running processes (for exit).
NOTIFY  : 2017.02.09-11:10:10 : Entering process monitoring loop.
NOTIFY  : 2017.02.09-11:10:14 : Run time 301 seconds.
NOTIFY  : 2017.02.09-11:10:14 : Executing statspack "after snap" procedure. Connect string is "sqlplus -S -L / as sysdba".
NOTIFY  : 2017.02.09-11:10:15 : After statspack snap ID is 26
NOTIFY  : 2017.02.09-11:10:16 : Terminating background data collectors.
./runit.sh: line 1488:  9011 Killed                  ( iostat -xm 3 > iostat.out 2>&1 )
./runit.sh: line 1488:  9012 Killed                  ( vmstat 3 > vmstat.out 2>&1 )
NOTIFY  : 2017.02.09-11:10:16 :
NOTIFY  : 2017.02.09-11:10:16 : SLOB test is complete.
NOTIFY  : 2017.02.09-11:10:16 : Cleaning up SLOB temporary directory (/tmp/.SLOB.2017.02.09.110501).
$
```

Figure 15: SLOB Multiple Schema Testing. Overriding slob.conf with Command Line Options

OS-Level Performance Data

The runit.sh produces iostat.out, vmstat.out and mpstat.out along with AWR reports. These commands are only executed on the host where runit.sh is executed so they are of little value when testing a remote host via SQL*Net.

The SLOB test administrator can disable the collection of OS performance data by setting and exporting NO_OS_PERF_DATA=TRUE before invoking the runit.sh script.

SLOB Tunable Parameters

UPDATE_PCT

The UPDATE_PCT parameter controls what percentage of SLOB operations that will modify blocks of data (modify DML.)

Values between 51 and 99 are non-deterministic.

Setting UPDATE_PCT to zero results in a 100% SQL SELECT workload. Please note, when testing SLOB with a small SGA buffer pool there will be physical reads from the tablespace even with UPDATE_PCT set to 100. Oracle Database cannot modify a block of data until it is first read into the SGA buffer pool.

RUN_TIME

The RUN_TIME parameter controls the wall-clock duration of a SLOB test in seconds. Set RUN_TIME to an integer value and the SLOB runit.sh script will terminate the execution of the test accordingly. Note, RUN_TIME can be overridden with WORK_LOOP.

If a value is assigned to RUN_TIME then, correspondingly, WORK_LOOP should be set to zero.

WORK_LOOP

The WORK_LOOP parameter is used to control SLOB test duration based on iterations of the SLOB work loop. The SLOB work loop consists of selecting a random set of blocks upon which SLOB performs a SLOB operation (SELECT or UPDATE SQL operation). When controlling a SLOB test with the WORK_LOOP parameter, testing is not complete until all SLOB threads (Database sessions) have performed WORK_LOOP number of SLOB operations.

SCALE

The SCALE parameter controls both data loading and test execution behavior. In other words, both setup.sh and runit.sh use this parameter.

SCALE can be assigned simple integer values or integer values modified by M/G/T (megabytes, gigabytes, terabytes) nomenclature. When assigned a simple integer value SCALE is the number of database blocks that will be loaded by setup.sh. For example, setting SCALE to 10000 will cause roughly 80 megabytes of data to be loaded into each SLOB schema (10,000 SLOB rows and therefore 10,000 SLOB blocks + slight index overhead). Alternatively, SCALE could be set to “80M” to achieve roughly the same schema fill levels.

SCAN_PCT

The scan table feature is new in SLOB 2.4.

The SCAN_PCT parameter establishes the percentage of SELECT operations that will perform a short scan against a short scan table. For example, if UPDATE_PCT is set to 50, SLOB will perform equal amounts of UPDATE and SELECT DML. Of the remaining SLOB operations all will be SELECT statements. If SCAN_PCT were set to 50, then half of the SELECT statements will be short scans against a short scan table.

Remember that with Multi-Schema Model there is one short scan table for every schema. On the other hand, with Single Schema Model, there is only a single traditional SLOB table and a single short scan table. One should expect a reasonable amount of waits for the *read by other session* wait event when SCAN_PCT is set high enough to have multiple, concurrent sessions attacking a scan table. This is true whether there are multiple threads in the Single Schema Model or multiple shared schemas in the threaded Multiple Schema Model.

The only way to prevent *read by other session* wait events, when testing with short table scans, is to use Multi-Schema Model and only a single thread per schema.

The default value for SCAN_PCT is zero which disables short table scans during a performance test (runit.sh).

SCAN_TABLE_SZ

The SCAN_TABLE_SZ is new in SLOB 2.4. The SCAN_TABLE_SZ parameter establishes the size of the short scan table(s). When set to a non-zero value, the SLOB data loader (setup.sh) will create, and load, a short scan table in every schema (whether Single Schema Model or Multi-Schema Model).

The SCAN_TABLE_SZ parameter can be assigned simple integer values or integer values modified by M/G/T (megabytes, gigabytes, terabytes) nomenclature.

The default value is 1 megabyte (128 8KB database blocks and thus 128 rows of data as is the norm with the SLOB Method).

Setting this value to zero will result in no short scan table creation when executing the SLOB data loader (setup.sh).

WORK_UNIT

During SLOB testing, WORK_UNIT controls the scope of blocks being manipulated by each SLOB operation. For example, if WORK_UNIT is set to 32 then each SELECT and UPDATE will scope and manipulate 32 random blocks of data in the tablespace. When testing with high levels of UPDATE_PCT (e.g., 50%) one generally sets WORK_UNIT small (e.g., 16) so as to not burden the UNDO functionality of Oracle Database. On the contrary if stressing UNDO functionality is of interest then one would set UPDATE_PCT and WORK_UNIT to large values such as 50 and 1024 respectively.

REDO_STRESS

Set this parameter to either HEAVY or any other non-null value. When set to HEAVY SLOB will generate significant amounts of redo logging (e.g., hundreds of megabytes per second on high performance platforms). Generally speaking, setting REDO_STRESS to a value other than HEAVY will generate reasonable amounts of redo. The default is LITE.

LOAD_PARALLEL_DEGREE

The LOAD_PARALLEL_DEGREE parameter serves two purposes depending on whether the test administrator is loading SLOB Single or Multiple Schema. In both Single and Multiple Schema models this parameter controls the number of Oracle Database sessions concurrently inserting data into the "base schema." The "base schema" is in the SLOB *user1* schema.

With Single Schema model there is only the "base schema" hence LOAD_PARALLEL_DEGREE affects the total data loading time. On the contrary, Multiple Schema model has potentially many schemas in addition to the "base schema." In this model LOAD_PARALLEL_DEGREE has a bit of a *batching* effect by controlling how many schemas are being loaded in parallel—after the base table is loaded in parallel. For example, if LOAD_PARALLEL_DEGREE is 16 then the base schema will be loaded with 16 processes running in parallel. If the load operation happens to be Multiple Schema model then, once the base schema is loaded, there will be sets of 16 processes loading into 16 different schemas in parallel. If there are, for example, 128 schemas to be loaded in Multiple Schema model then there will be 16 processes loading the base schema followed by the remaining 127 schemas loaded with 7 sets of 16 and a final set of 15 (1 + (7*16) + 15 = 128).

THREADS_PER_SCHEMA

The THREADS_PER_SCHEMA parameter controls how many SLOB threads (Oracle Database sessions) will be performing SLOB operations against each schema during a performance test. The THREADS_PER_SCHEMA parameter can be overridden by the –t option to the SLOB runit.sh script.

DO_HOTSPOT

The test administrator can enable SLOB Hot Spot testing by setting DO_HOTSPOT to TRUE.

Setting DO_HOTSPOT to FALSE disables SLOB Hot Spot functionality.

HOTSPOT_MB

The HOTSPOT_MB parameter controls the size of each Hot Spot in megabytes whether testing Single Schema or Multiple Schema model.

HOTSPOT_OFFSET_MB

The HOTSPOT_OFFSET_MB parameter controls the location (within the SLOB active data set) of the SLOB Hot Spot in each schema being tested. The Hot Spot will be located at the same offset in every schema being tested whether SLOB Single or Multiple Schema model

HOTSPOT_FREQUENCY

Each SLOB thread (Oracle Database session) performs SLOB operations in a loop. The HOTSPOT_FREQUENCY parameter controls the frequency at which each session will target the SLOB Hot Spot with a SLOB operation. For example, setting HOTSPOT_FREQUENCY to 7 means every 7th SLOB operation targets the SLOB Hot Spot of each schema being tested during a performance test.

HOT_SCHEMA_FREQUENCY

The HOT_SCHEMA_FREQUENCY parameter controls the frequency at which each session will target the SLOB Hot Schema (the Oracle Database *user1* schema) with a SLOB operation. For example, setting HOT_SCHEMA_FREQUENCY to 3 means 33% of all SLOB operations will target the Hot Schema.

Setting HOT_SCHEMA_FREQUENCY to zero disables SLOB Hot Schema functionality.

THINK_TM_FREQUENCY

The THINK_TIME_FREQUENCY parameter controls the frequency at which each SLOB thread pauses between SLOB operations so as to simulate think time of a human database user. SLOB think-time is implemented with the DBMS_LOCK.SLEEP package. For example, if THINK_TIME_FREQUENCY is 2 then each SLOB thread will pause after every other SLOB operation. The length of pause is a random value in seconds between THINK_TM_MIN and THINK_TM_MAX.

Setting THINK_TM_FREQUENCY to zero disables think time functionality.

THINK_TM_MIN

When testing SLOB with think time (see THINK_TM_FREQUENCY) the THINK_TM_MIN parameter establishes the low bound for the random sleep value chosen by each SLOB thread, at each frequency interval, as per the value assigned to THINK_TM_FREQUENCY.

The unit of time for THINK_TM_MIN is seconds and values to hundredths-precision are supported.

THINK_TM_MAX

When testing SLOB with think time (see THINK_TM_FREQUENCY) the THINK_TM_MAX parameter establishes the high bound for the random sleep value

chosen by each SLOB thread, at each frequency interval, as per the value assigned to THINK_TM_FREQUENCY.

The unit of time for THINK_TM_MAX is seconds and values to hundredths-precision are supported.

DATABASE_STATISTICS_TYPE

New in SLOB 2.4 is the ability to generate statspack reports. The legal values for DATABASE_STATISTICS_TYPE are "awr" or "statspack."

The default value is statspack. To generate Automatic Workload Repository (AWR) reports, set this parameter to "awr".

EXTERNAL_SCRIPT

The EXTERNAL_SCRIPT parameter enables the SLOB administrator to perform a custom script before and after the measured test portion of runit.sh processing. The script must take only one of three legal arguments:

- "pre"
- "post"
- "end"

The script must exit successfully (exit 0) in order for runit.sh to function properly.

Example EXTERNAL_SCRIPT Use Cases

An example use case for EXTERNAL_SCRIPT functionality is to execute operating system commands to communicate with a storage array for the purpose of starting or stopping array performance metrics. Another example would be to start OProfile profiling (i.e., opcontrol –start) in the "pre" section of the script and then conclude the profiling in the "post" section of the script (i.e., opreport –l).

The "end" function of an external script can be helpful for storing the results files produced by a SLOB test. For example, the "end" section might create a directory and copy files such as slob.conf and AWR reports (or STATSPACK) into the directory. In the case of interacting with a storage array, one might use the "end" section to copy storage array performance files into a subdirectory.

There is a skeleton external script called external_script.sh in the SLOB/misc directory.

DBA_PRIV_USER

By default, both the SLOB data loader (setup.sh) and the test driver (runit.sh) will connect to the instance with the "system" database account with the typical "manager" password. Note: older versions of SLOB used the "sys" account for administrative connections. See also SYSDBA_PASSWD.

If you wish to connect to the instance as an elevated privilege account other than SYSTEM, then you must set DBA_PRIV_USER. The most common usage for this parameter is to connect to cloud DBaaS such as Amazon RDS for Oracle.

For example, if testing Amazon Web Services (i.e., Amazon RDS for Oracle), it is not possible to connect as SYSTEM, nor SYS. The *master account*, for an RDS instance, is the elevated priority user with sufficient permissions to administrative SLOB tasks typically performed with a "/ as sysdba" connection. To that end, the SLOB test administrator will need to set DBA_PRIV_USER=<*master account*> when testing Amazon RDS for Oracle. When DBA_PRIV_USER is set then SYSDBA_PASSWD must also be set.

Figure 16 shows a screen shot of a portion of the Amazon RDS for Oracle Configuration Details page in the AWS Console. Highlighted in yellow is the *master account* name for the instance. To use this RDS instance with SLOB, the SLOB administrator would set the DBA_PRIV_USER parameter to "oracle" and the SYSDBA_PASSWD to the password chosen when launching the RDS for Oracle instance. A more detailed example follows.

NOTE: When setting DBA_PRIV_USER, it is **mandatory** to also set SYSDBA_PASSWD. For more information about the SYSDBA_PASSWD parameter please see the section below under the SYSDBA_PASSWD heading.

Figure 16: Connect Details for Oracle on Amazon RDS

Connecting to an Amazon RDS for Oracle Instance – An Example

The following example is provided to further illustrate the relationships between slob.conf parameters for environments where connecting with the "system" (SYSTEM) database account is not possible—as is the case with Amazon RDS for Oracle.

Figure 17 shows the tnsnames.ora entry configured for connecting to the RDS for Oracle instance with details shown in Figure 16. A successful tnsping displays the tnsnames.ora entry.

Figure 17 also shows the sqlplus connect string, grants and relevant slob.conf entries used to connect to the RDS for Oracle instance as the master user. As stated above, the *master user* for this instance is "oracle" (highlighted in yellow in both Figures 16 and 17).

```
$ tnsping slob8

TNS Ping Utility for Linux: Version 12.1.0.2.0 - Production on 04-MAY-2017 20:56:06

Copyright (c) 1997, 2014, Oracle.  All rights reserved.

Used parameter files:


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION= (ADDRESS= (PROTOCOL=TCP) (HOST=slob8.[redacted].us-west-2.rds.amazonaws.com) (PORT=1521)) (
CONNECT_DATA= (SERVER=dedicated) (SERVICE_NAME=ORCL)))
OK (60 msec)
$
$ sqlplus oracle@slob8/oraclepassword

SQL*Plus: Release 12.1.0.2.0 Production on Thu May 4 20:56:12 2017

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Thu May 04 2017 20:55:58 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> @grants
SQL> col grantee format a20
SQL> select * from dba_role_privs where granted_role='DBA';

GRANTEE              GRA ADM DEL DEF COM
-------------------- --- --- --- --- ---
RDSADMIN             DBA YES NO  YES NO
ORACLE               DBA YES NO  YES NO
SYSTEM               DBA NO  NO  YES YES
SYS                  DBA YES NO  YES YES

SQL>
SQL> !egrep 'SYSDBA_PASSWD|DBA_PRIV_USER' slob.conf
SYSDBA_PASSWD=oraclepassword
DBA_PRIV_USER=oracle

SQL>
```

SQL*Net Related Parameters

ADMIN_SQLNET_SERVICE

The ADMIN_SQLNET_SERVICE parameter is used by both setup.sh and runit.sh. If you want all system user (or slob.conf->DBA_PRIV_USER) connections to go through a specific TNS names service then set this parameter accordingly. For example, you might care to have a SQL*Net service called SLOBDBA. As such you would set:

ADMIN_SQLNET_SERVICE=SLOBDBA.

SQLNET_SERVICE_BASE

This parameter is only used by runit.sh and serves multiple purposes. It is used for either

- Auto-load balanced SQL*Net connections, or
- Round-robin style SLOB connections
    - More detail is provided below below under the heading Understanding SLOB Round Robin SQL*Net Connections

Simple Real Application Clusters Auto Load Balanced Connections

If SQLNET_SERVICE_BASE is set, but SQLNET_SERVICE_MAX is NULL, then SQLNET_SERVICE_BASE will be used during execution of runit.sh to direct all SLOB user sessions to this service. For example, consider a testing environment where SLOB will be used to test a Real Application Clusters configuration where SQL*Net is configured to load balance connections through a single TNS entry called "PROD". While there may be instances PROD1, PROD2 and so on, the DBA has chosen to not expose TNS services to connect to the individual instances but, instead, to have all connections come through the "PROD" TNS service. In this situation, SQLNET_SERVICE_BASE would be set to "PROD" and SQLNET_SERVICE_MAX would be set to NULL.

*Understanding SLOB Round Robin SQL*Net Connections*

If SQLNET_SERVICE_BASE is non-null and SQLNET_SERVICE_MAX is set to a non-zero integer then runit.sh will treat the value assigned to SQLNET_SERVICE_MAX as the highest integer value to append to SQLNET_SERVICE_BASE during a round-robin connection test. In other words, if both SQLNET_SERVICE_BASE and SQLNET_SERVICE_MAX are set no non-NULL then SQLNET_SERVICE_BASE becomes a SQL*Net service *base name* and integer values 1 through SQLNET_SERVICE_MAX will be appended in a round-robin fashion. See SQLNET_SERVICE_MAX for more information.

SQLNET_SERVICE_MAX

This parameter is only used by runit.sh. When set to a non-zero integer, SQLNET_SERVICE_MAX is the highest integer value appended to SQLNET_SERVICE_BASE in a Real Application Clusters testing scenario. For example, if SQLNET_SERVICE_MAX is 3 and SQLNET_SERVICE_BASE is set to "SLOB" then runit.sh will round-robin the connections from the SLOB1 service to SLOB2 then SLOB3 and back to SLOB1. This, of course, presumes you have configured functioning SQL*Net services called SLOB1, SLOB2 and SLOB3 in tnsnames.ora.

It's best to set this parameter to an equal divisor of the number of SLOB threads you will test with.

SYSDBA_PASSWD

This parameter is used by both runit.sh and setup.sh and accompanies DBA_PRIV_USER.

The default for this setting is "manager" since the default for DBA_PRIV_USER is "system" (SYSTEM database account). If the default is not suitable then you must set both ADMIN_SQLNET_SERVICE and SYSDBA_PASSWD to the credentials that will allow runit.sh (or setup.sh) to connect as an elevated (DBA) account. This might be the SYSTEM database account, another account entirely—as would be the case with cloud DBaaS such as Amazon RDS for Oracle.

The awr_info.sh Script

The awr_info.sh script (located in the SLOB/misc directory) post-processes the non-RAC report called awr.txt. The awr_info.sh script produces pipe-delimited performance data suitable for cut/paste into a spreadsheet.

To make the most of awr_info.sh it is best to rename the awr.txt file generated by the runit.sh script to indicate the number of SLOB threads. For example, in Figure 18 example usage of awr_info.sh is shown. In the example case there were two executions of runit.sh—one with the 64 SLOB threads and the other with 128 SLOB threads. The awr.txt file was named with the session count appended to each.

```
$
$ sh ./misc/awr_info.sh awr.txt.64 awr.txt.128
FILE|SESSIONS|ELAPSED|DB CPU|DB Tm|EXECUTES|LIO|PREADS|READ_MBS|PWRITES|WRITE_MBS|REDO_MBS|DFSR_LAT|DPR_LAT|DFPR_LAT|DFPW_LAT|LFPW_LAT|TOP WAIT|
awr.txt.64|64|122|4.6|62.5|2793|190086|90166|  705.3|26232|   325|34.6|      633|0|0|      59|    2935|db file sequential read         11095432     7018.6       0.63   91.3 User I/O|
awr.txt.128|128|122|11.8|125.8|3450|235104|115520|  903.7|40406|   412|43.2|      976|0|0|     271|    6136|db file sequential read         14115025      13.8K       0.98   89.7 User I/O|
$
```

Figure 18: SLOB awr_info.sh Script Output Example.

The awr_info.sh Script Columns – A Legend

FILE
The name of the file processed by awr_info.sh.

SESSIONS
The number of SLOB threads (Oracle Database sessions) used in the test that produced the awr.txt file. The SLOB test administrator would rename awr.txt to awr.txt.N where N is the number of SLOB threads.

This value (*N*) is the product of multiplying the arguments supplied to runit.sh (–s and –t options). For example, if testing Single Schema Model, with 64 threads, the arithmetic would be 1 * 64 = 64 (runit.sh –s 1 –t 64). In the case of Multiple Schema Model with, say, 4 schemas and 16 threads per schema the arithmetic would be 4 * 16 = 64 (runit.sh –s 4 –t 16).

ELAPSED
Run time in seconds.

DB CPU
DB CPU per second

DB Tm
DB Time per second

EXECUTES
SQL executions per second.

LIO
Logical reads (buffer cache hits) per second

PREADS
Physical reads per second.

READ_MBS
Physical read throughput in megabytes per second.

PWRITES
Physical writes per second.

WRITE_MBS
Physical write throughput in megabytes per second.

REDO_MBS
Redo write throughput in megabytes per second.

DFSR_LAT
Latencies (service times) in microseconds for db file sequential read wait events.

DPR_LAT
Latencies (service times) in microseconds for direct path read wait events.

DFPR_LAT
Latencies (service times) in microseconds for db file parallel read wait events.

DFPW_LAT
Latencies (service times) in microseconds for db file parallel write background wait events.

LFPW_LAT
Latencies (service times) in microseconds for log file parallel write background wait events.

TOP WAIT
The top wait event in suffered during the test execution.

Advanced Topics
In the main SLOB directory you can find a subdirectory called advanced_topics such as shown in Figure 18:

```
$
$ pwd
/home/oracle/build/SLOB
$ ls
advanced_topics  awr  misc  README.AIX  runit.sh  setup.sh  simple.ora  slob.conf  slob.sql  wait_kit
$ cd advanced_topics
$ ls -l
total 720
-rw-r--r--. 1 oracle dba    601 May  2 13:46 advanced.ora
-rw-r--r--. 1 oracle dba  45230 May  2 13:58 awr.html.gz
-rw-r--r--. 1 oracle dba  33266 May  2 13:59 awr_rac.html.gz
-rw-r--r--. 1 oracle dba 165982 May  2 13:58 awr.txt.128
-rw-r--r--. 1 oracle dba   2588 May  2 13:58 db_stats.out
-rw-r--r--. 1 oracle dba  98530 May  2 13:58 iostat.out
-rw-r--r--. 1 oracle dba 345167 May  2 13:58 mpstat.out
-rw-r--r--. 1 oracle dba    487 May  2 13:48 slob.conf
-rw-r--r--. 1 oracle dba   4573 May  2 14:02 typescript
-rw-r--r--. 1 oracle dba  11399 May  2 13:58 vmstat.out
$
```

Figure 17: SLOB Advanced Topics Directory Listing.

This directory has the output produced by a bona fide SLOB test using the slob.conf also in this directory. Note: this content was produced prior to SLOB 2.3 so the runit.sh output and slob.conf parameters differ from SLOB 2.3 accordingly. The directory also directory the init.ora file and—most importantly—the screen capture from the test execution in the file called typescript. I recommend examining the contents of this directory once you've worked out the basics.

Where To Get More Information

SLOB has gained a great deal of popularity. In addition to kevinclosson.net/slob you can simply enter the search term “oracle slob” in your favorite search engine to learn what others in the user community are sharing about their use and knowledge of SLOB.