SLOB 2

DISCLAIMER 2

PURPOSE 3

SLOB 2 – FUNCTIONALITY IN COMMON WITH PRIOR VERSIONS 3

TABLESPACE REQUIREMENT 3
SYS V IPC SEMAPHORES 3
DATABASE CREATION KIT 3
THE RUNIT.SH SCRIPT 3
CATCHING UP ON THE PAST 4

SLOB 2 WHAT'S NEW? 4

USER CONFIGURABLE RUN CHARACTERISTICS 4
TIMED RUNIT.SH COMPLETION 4
DATABASE SCALE 4
SHARED DATA CONTENTION 4
REAL APPLICATION CLUSTERS 4
NO MORE READERS WITH WRITERS 5
THE AWR_INFO.SH SCRIPT 5
OS-LEVEL PERFORMANCE DATA 5
ABOUT THE DB_STATS.OUT FILE 5
CONFIGURABLE SELECT UPDATE RATIO 5
DATA LOADING 5

INTRODUCING THE SLOB.CONF FILE 6

LEGEND 6
UPDATE_PCT 6
RUN_TIME 6
WORK_LOOP 6
SCALE 7
WORK_UNIT 7
REDO_STRESS 7
SHARED_DATA_MODULUS 7
LOAD_PARALLEL_DEGREE 7
ABOUT SQL*NET CONNECTIVITY AND REAL APPLICATION CLUSTERS SUPPORT—MORE USES FOR THE SLOB.CONF FILE 7
A WARNING ABOUT SQL*NET WITH SLOB 7
ADMIN_SQLNET_SERVICE 8
SQLNET_SERVICE_BASE 8
SQLNET_SERVICE_MAX 8
SYSDBA_PASSWD 8

INTRODUCING THE AWR_INFO.SH SCRIPT 9

LEGEND FOR AWR_INFO.SH SCRIPT COLUMNS 9

FILE 9
SESSIONS 9
ELAPSED 9
EXECUTES 9
PREADS 9
READ_MBS 9
PWRITES 9
WRITE_MBS 9
REDO_MBS 10
DFSR_LAT 10
DPR_LAT 10
DFPR_LAT 10
DFPW_LAT 10
LFPW_LAT 10

ADVANCED TOPICS 10

WHERE TO GET MORE INFORMATION 11

Disclaimer

Purpose

SLOB is a collection of computer software suitable for educational purposes. SLOB is helpful for studying computer systems performance including server hardware and software, storage system hardware and firmware and storage networking hardware and firmware.

Users should seek advice regarding license to use any third-party software involved during SLOB testing such as Operating System Software, Storage (and Storage Networking Infrastructure) Firmware and Oracle Database.

SLOB 2 – Functionality In Common With Prior Versions

Tablespace Requirement

The SLOB setup script (setup.sh) requires a tablespace with sufficient space to contain the SLOB schema. The default name of the SLOB tablespace is IOPS.

The default space requirement is roughly 12 gigabytes for setup.sh plus any ancillary overhead in TEMP segments for such purposes as sort spill. Naturally, UNDO segment space will be required for before imaging.

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 sempahore set that contains a single semaphore.

Database Creation Kit

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

For more information on the create_database_kit please see the README file in that directory.

The runit.sh Script

The runit.sh script is used in SLOB 2 in the same manner as SLOB 1. Please see the section below that explains the changes to execution arguments for runit.sh.

Catching Up On The Past

There is a wealth of information about prior SLOB versions on the internet. A good place to start is at the following link:

http://kevinclosson.wordpress.com/2012/02/06/introducing-slob-the-silly-little-oracle-benchmark/

SLOB 2 What's New?

User Configurable Run Characteristics

There is a new slob.conf file that controls many options related to both loading the database and running the workload. See Introducing slob.conf below.

Timed runit.sh Completion

The runit.sh script will now (optionally) run for N seconds. It is now possible to have SLOB execute for N iterations of the work loop for every session or to execute for a fixed number of seconds.

Database Scale

The setup.sh script now supports configurable database scale. The traditional "reference scale" is 10,000 rows into blocks that each hold only a single row. Thus, a "reference scale unit" is roughly 82MB. With the default maximum schema users the traditional SLOB IOPS tablespace consumes roughly 10.5GB. Thusly, setting SCALE in slob.conf to 1,000,000 generates roughly 1TB when loading 128 SLOB schema users.

Shared Data Contention

The idea behind SLOB is to test storage subsystems without testing Oracle transactional scalability. Oracle does a great job proving transactional scalability with audited proof points like TPC-C. Moreover, no synthetic workload can properly mimic your production workload so it makes little sense to include irrelevant transactional overhead when the task at hand is to focus on your storage subsystem. For this reason SLOB sessions, by default, never share application data. SLOB sessions operate on their own tables and indexes in private schemas.

Real Application Clusters

Over time SLOB has garnered interest from people that want to test Real Application Clusters (RAC). This version of SLOB does indeed support RAC, however, RAC without any shared data contention is not really RAC. To that end, this version of SLOB introduces configurable shared data contention—a feature that can be used with or without RAC. SLOB offers this functionality via a new "zero" user (user0/user0) with SLOB tables and SLOB indexes.

If you configure the slob.conf file appropriately every Nth operation will access the user0 schema. There is more information on this below in the section covering slob.conf parameters.

No More Readers With Writers
The runit.sh script now only takes a single option which is the number of SLOB sessions to run, so, simply:

```
$ sh ./runit.sh 32
```

The awr_info.sh Script
See the section below under the Introducing the awr_info.sh Script heading.

OS-Level Performance Data
Runit.sh now produces iostat.out, vmstat.out and mpstat.out along with awr.txt. Additionally, AWR reports are now created in HTML form including the consolidated report to ensure proper data capture for RAC testing.

You might care to have your wrapper script make a sub-directory after runit.sh exits and copy all these into the directory named in a way that makes sense. For instance, if you configure a set slob.conf UPDATE_PCT=25 and run with 32 sessions you might make a directory called 32.25 and move these files that directory.

About The db_stats.out File
A new output file called db_stats.out is created during a SLOB run. The db_stats.out file is a pipe-delimited file with one line per session during the execution of the workload. The lines contain columns corresponding to the SLOB user number, the length of run in centiseconds, the CPU time consumed by the SLOB session in centiseconds and a percent figure to show how CPU-intensive the session was.

Configurable SELECT UPDATE Ratio
SLOB now allows you to configure the percentage SQL executions that are UPDATE DML. More information on this topic is offered in the detailed slob.conf section.

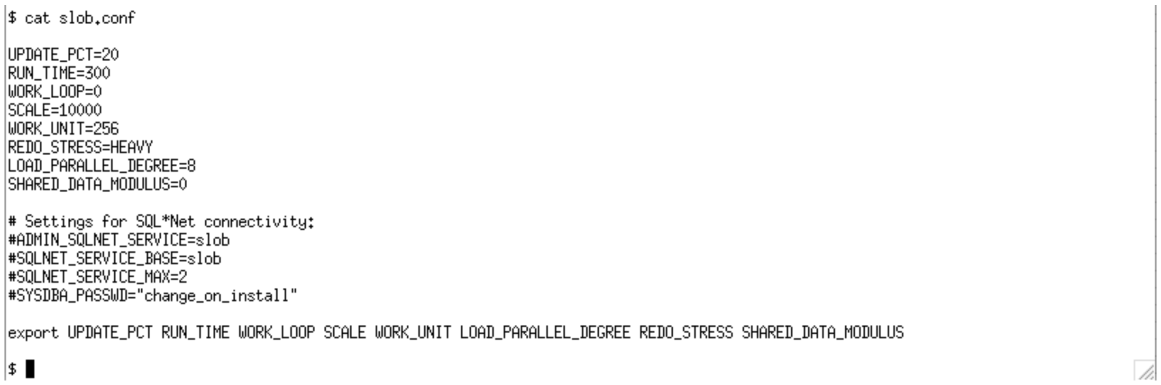
Data Loading
There is a configurable parameter to drive how many SLOB schemas are populated concurrently when executing setup.sh. A good limit to abide by is 1 stream of data loading per processor thread. For example, as per the information in the following tweet about pre-release SLOB, I shared timings for 256GB scale loading results taken from a 2s16c32t E5 Xeon system with internal PCI Flash DAS. The database was ready to test in roughly 26 minutes:

https://twitter.com/kevinclosson/status/330042240187650050

The parallelism for the loading process is controlled by slob.conf LOAD_PARALLEL_DEGREE parameter as discussed in further detail below.

Introducing The slob.conf File

SLOB is now configurable via settings in the slob.conf file. The following is the default slob.conf:



```
$ cat slob.conf

UPDATE_PCT=20
RUN_TIME=300
WORK_LOOP=0
SCALE=10000
WORK_UNIT=256
REDO_STRESS=HEAVY
LOAD_PARALLEL_DEGREE=8
SHARED_DATA_MODULUS=0

# Settings for SQL*Net connectivity:
#ADMIN_SQLNET_SERVICE=slob
#SQLNET_SERVICE_BASE=slob
#SQLNET_SERVICE_MAX=2
#SYSDBA_PASSWD="change_on_install"

export UPDATE_PCT RUN_TIME WORK_LOOP SCALE WORK_UNIT LOAD_PARALLEL_DEGREE REDO_STRESS SHARED_DATA_MODULUS

$ 
```

LEGEND

UPDATE_PCT

Set to N where N is the percentage of all SQL that will be UPDATE DML executions. Values between 51 and 99 are non-deterministic. Setting to 0 or 100 are the functional equivalent of prior generations of SLOB where one might either execute 100% SELECT or 100% UPDATE workload.

RUN_TIME

Set to N where N is the number of seconds you want the test to run (runit.sh). This can be overridden with WORK_LOOP. If you set RUN_TIME you should set WORK_LOOP to 0.

WORK_LOOP

Set to N where N is the fixed number of loop iterations. Testing in this manner is a way to measure job completion (drag race) as opposed to fixed test period with sampling of stats. If you want to do a fixed-iteration test I recommend setting RUN_TIME large enough to make sure the run doesn't end based on RUN_TIME.

SCALE
Set to N where N=10000 is the old SLOB scale. N==250000 is ~256GB with 8KB block and 128 schema users.

WORK_UNIT
SLOB picks a random 256 blocks to work on in each iteration of the work loop. You can make this a smaller "bite" if you wish. This is not a very well tested tunable parameter so expect the unexpected.

REDO_STRESS
Set either to HEAVY or any other non-null value. HEAVY maps to the old writer.sql.heavy. Any value other than HEAVY reduces the redo payload in MB/s by about 80%.

SHARED_DATA_MODULUS
This parameter controls the degree of shared data contention. If set to a non-zero value it is used in modulo arithmetic for every UPDATE (as per UPDATE_PCT) to direct either to the session's private schema or shared data manipulation (the user0 schema).

For example, if UPDATE_PCT is 25 and SHARED_DATA_MODULUS is 2 then 25% of all SQL will be UPDATE and of that half will settle on the shared schema (user0). In other words UPDATE_PCT=25 and SHARED_DATA_MODULUS=2 results in 12.5% of all SQL directed at the shared data schema. Setting this parameter to 1 results in 100% shared data UPDATE contention.

If you enable shared data contention by setting SHARED_DATA_MODULUS expect to see the wait event *enq: TX - row lock contention*.

LOAD_PARALLEL_DEGREE
The setup.sh script uses this to control concurrent data loading. I recommend 1 per core at a minimum and 1 per processor thread at a maximum. Note: if your instance has Parallel Query Option enabled, setting LOAD_PARALLEL_DEGREE will result in concurrent parallel query loading. Concurrent loading that involves Parallel Query Option has been tested to varying degrees but your results may vary and require tuning of Parallel Query initialization parameters.

About SQL*Net Connectivity and Real Application Clusters Support—More Uses For The slob.conf File

A Warning About SQL*Net With SLOB
It's best to ensure your SQL*Net connectivity works—as intended—before you set slob.conf SQL*Net parameters and start executing scripts.

Both the runit.sh and setup.sh scripts validate the slob.conf SQL*Net *connections* at runtime, however, the limit of that validation is whether or not a connection can be made. **If you have slob.conf SQL*Net parameters configured to connect to database you do not intest to test with SLOB, well, SLOB has no knowledge of that.**

ADMIN_SQLNET_SERVICE

If you want all SYSDBA connections to go through a specific tnsnames service then set this parameter accordingly. For example, you might care to have a SQL*Net service called SLOB_DBA. As such you would set:

ADMIN_SQLNET_SERVICE=SLOBDBA.

SQLNET_SERVICE_BASE

This parameter serves multiple purposes.

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 sessions to this service.

On the other hand, if SQLNET_SERVICE_MAX is a non-zero integer then runit.sh will treat this value as the highest integer value to append to SQLNET_SERVICE_BASE during a round-robin connection test. In other words, if both of the parameters SQLNET_SERVICE_BASE and SQLNET_SERVICE_MAX are set then SQLNET_SERVICE_BASE becomes a *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

If 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 4 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 so forth and back to SLOB1 once SQLNET_SERVICE_MAX has been reached.

It's best to set this parameter to an equal divisor of the number of sessions you will test with (specifically the argument passed to runit.sh).

SYSDBA_PASSWD

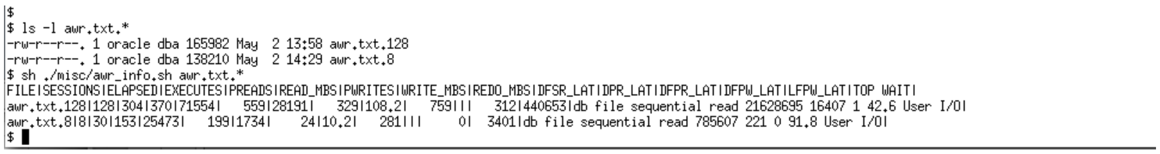
If ADMIN_SQLNET_SERVICE is set then the scripts (setup.sh and runit.sh) will need a password to connect to the database via a SQL*Net service as SYSDBA. For example, if ADMIN_SQLNET_SERVICE is set to "SLOB" and you've configured the Oracle password file (via the orapwd utility) for SYSDBA to "change_on_install" then the scripts will use the following connect string for sqlplus to connect as SYSDBA:

```
      sys/change_on_install@slob as sysdba
```

Introducing The awr_info.sh Script

The awr_info.sh script takes awr.txt files—named as per the requisite naming convention—and produces helpful pipe-delimited output that can be pasted into a spreadsheet.

When a run is complete, and awr.txt is named to awr.txt.N simply use awr_info.sh to get the facts about the run. In the following example there have been two executions of SLOB with different session counts and slob.conf configuration:



```
$
$ ls -l awr.txt.*
-rw-r--r--. 1 oracle dba 165982 May  2 13:58 awr.txt.128
-rw-r--r--. 1 oracle dba 138210 May  2 14:29 awr.txt.8
$ sh ./misc/awr_info.sh awr.txt.*
FILE|SESSIONS|ELAPSED|EXECUTES|PREADS|READ_MBS|PWRITES|WRITE_MBS|REDO_MBS|DFSR_LAT|DPR_LAT|DFPR_LAT|DFPW_LAT|LFPW_LAT|TOP WAIT|
awr.txt.128|128|304|370|71554|   559|28191|   329|108.2|   759|||   312|440653|db file sequential read 21628695 16407 1 42.6 User I/O|
awr.txt.8|8|30|153|25473|   199|1734|    24|10.2|   281|||     0|  3401|db file sequential read 785607 221 0 91.8 User I/O|
$ 
```

Legend For The awr_info.sh Script Columns

FILE
The name of the awr.txt.N file.

SESSIONS
The number of sessions used in the run that produced the awr.txt.N file. This is the option passed to the runit.sh script.

ELAPSED
Run time in seconds.

EXECUTES
SQL executions 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.

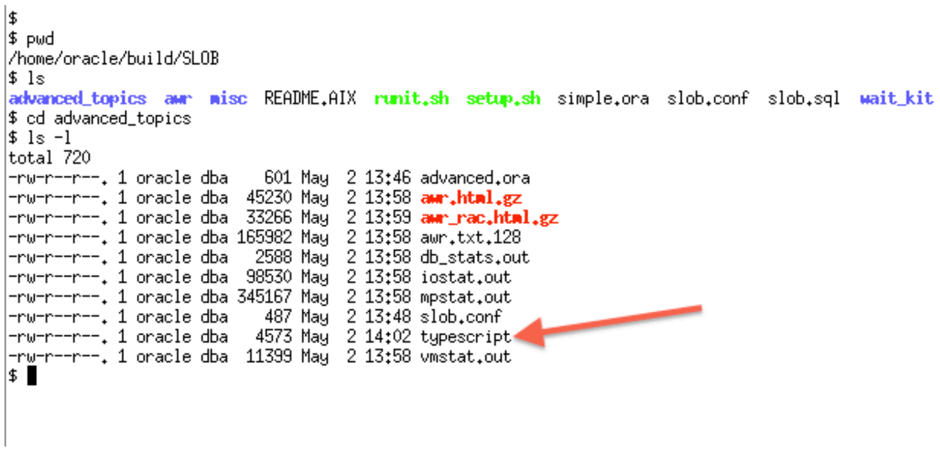
DPR_LAT
Latencies (service times) in microseconds for direct path read wait events.

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.

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



```
$
$ 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
$ █
```

This directory has the output produced by a bona fide run of SLOB using the slob.conf also in this directory. Additionally the directory contains the init.ora file and—most importantly—the screen capture from the run in the file called typescript. I highly 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 the above-cited web pages 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: