Oracle DB-Tuning Essentials

Agenda

1. The DB server and the tuning environment
2. Objective, Tuning versus Troubleshooting, Cost Based Optimizer
3. Object statistics
4. Access paths I
5. Access paths II
6. Monitoring Performance I
7. Monitoring Performance II
8. Application setup

Monitoring Performance

- Database Time
- AWR, ASH, ADDM
- Statspack
- dbms_xplan
- SQL Monitoring
- Application instrumentation
- SQL Trace, tkprof

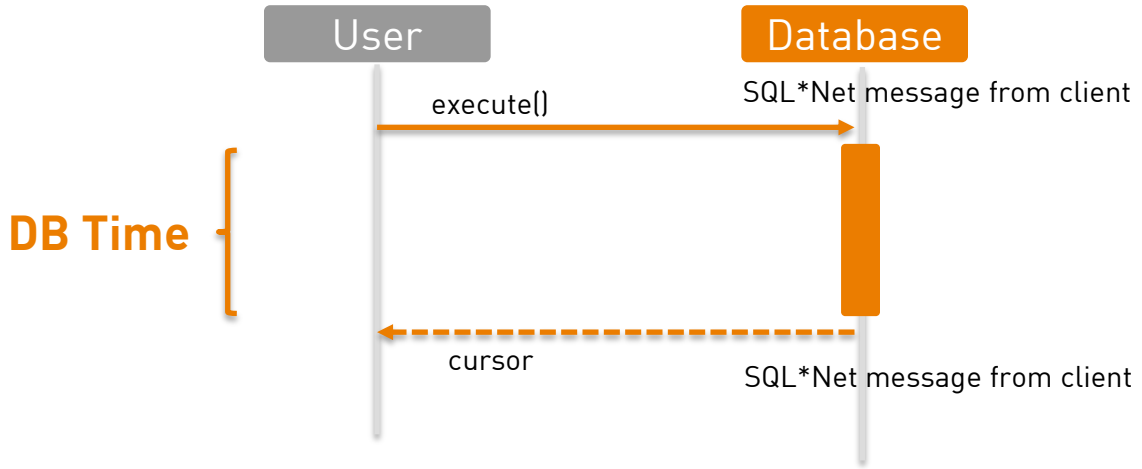
Database Time
User calls

The databases serves user calls

- SQL or PL/SQL parsing
- SQL, PL/SQL, Java,... execution
- SQL fetch,...

The database user is:

- A real user in Client/Server (e.g sqlplus)
- An application server (e.g. one connection from a connection pool)



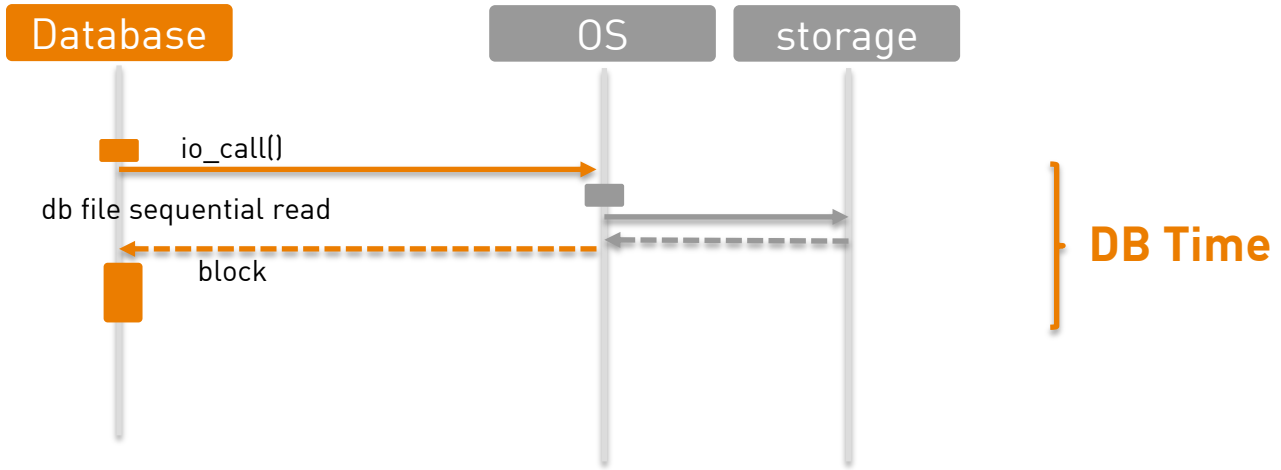
Database Time
Timed foreground event

The databases uses server resources

- CPU (running or waiting in runqueue)
- System calls (network, i/o, timers)

The database measures time

- CPU Time: time running on CPU (waiting for CPU not included)
- Wait Events: system calls (including CPU runqueue when CPU is busy)



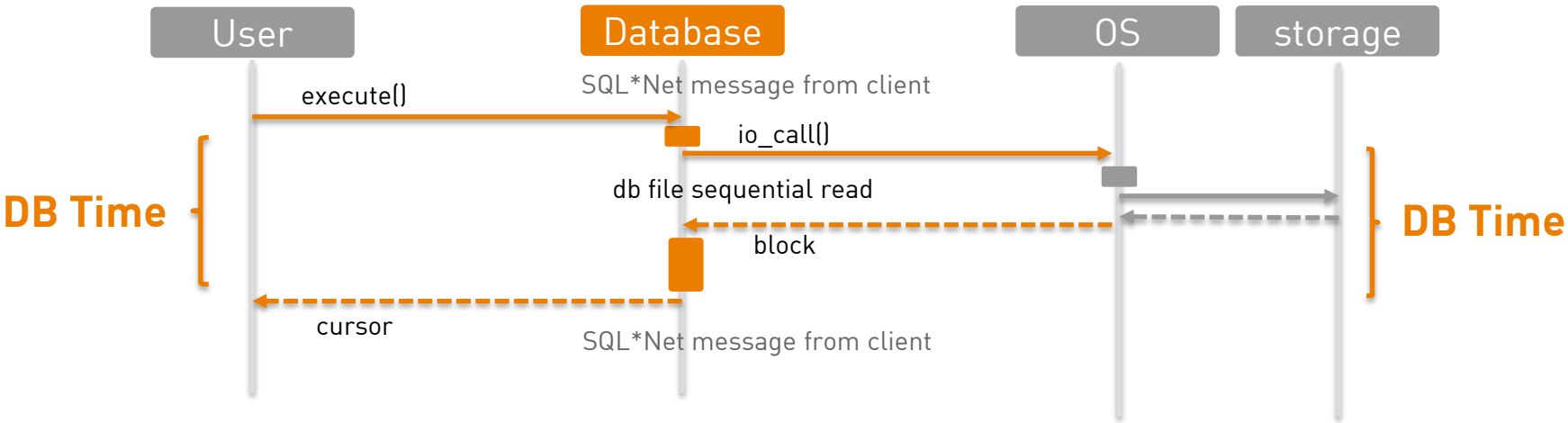
Database Time
DB time

Measures the response time

- Detailed in the Time Model

Measures resource consumption

- DB Time = CPU + Wait Events + (some wait in runqueue)

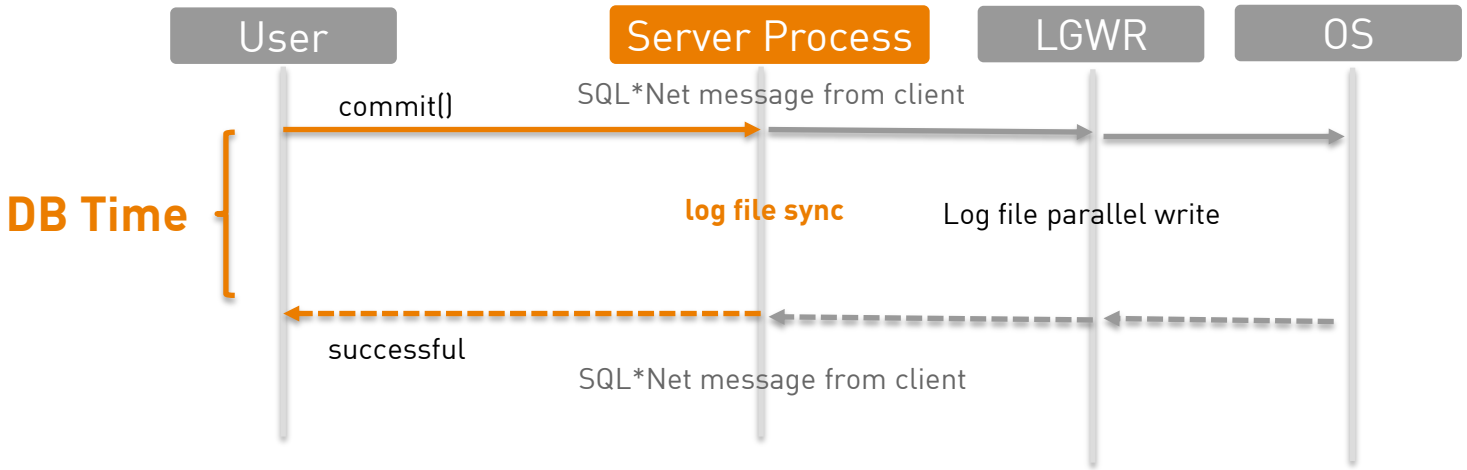


Database Time
Foreground processes

User call is served by the foreground process

- Wait events not participating in the calle are 'idle' events
- Wait events in background processes are 'idle' events

Only non-idle events are counted in DB Time



Here DB Time for the commit call is the 'log file sync' time
Time model shows 'sql execution' for the commit
Timed events shows 'log file sync'

Database Time
Average Active Session

During 1 hour of elapsed time

- One user can use 1 hour of DB Time
- N users can use N hours of DB Time
- The average active session is DB Time / elapsed time

```
Snap Id      Snap Time                      Sessions Curs/Sess
             --------- ------------------- -------- ---------
 Begin Snap:       330 11-Mar-14 15:18:53        40       5.4
   End Snap:       336 11-Mar-14 15:34:32        55       4.7
    Elapsed:               15.64 (mins)
    DB Time:               76.48 (mins)
```

Database load= 76.48 / 15.64 = 4.9 Average Active Session

It's an average and can mean: 5 users 99% busy 

or 10 users half idle 

or ... 

Database Time
Statistics and Wait Events

Oracle gathers statistics about:

- Time spent in user calls (time model)
- Time spent in system calls (wait events)
- Time running on CPU (as reported by the OS)
- Various non-time statistics about the activity

Most statistics are cumulative

- From instance startup
- Exposed in V$ views
- Historized through regular snapshots (AWR or Statspack)
- We analyze the delta between two snapshots

Database Time
DB CPU

DB CPU

- Is the time spend on CPU
- Waiting in runqueue is not accounted here

Facts to know

- Everything needs CPU: you will always have CPU consumption
- One session cannot use more than one CPU (except parallel query)

Main responsibles for CPU consumption

- Parsing: should be avoided (bind variables)
- Sorting: can be tuned (workarea size)
- Block reads: lowering the logical reads will decrease CPU usage

Database Time
User I/O

User I/O

- db file sequential read: read one block to buffer cache
  (typical for index access)
- db file squattered read: read multiple contiguous blocks to buffer cache
  (typical for a full table scan)
- db file direct path read: read contiguous blocks directly to the PGA
  (full table scan on very large tables)

The average time for an IO should be known

- Example: 4ms for 15000 RPM, 0.1ms for SSDs

Database Time
Application

enq: TX

- enq: TX - row lock contention: a row has been locked – waiting for the transaction to complete.
- enq: TX - allocate ITL entry: not enough free space for the ITL
- enq: TX – index contention: index entry locked (unique constraint)

enq: TM

- Table locks (RS, RX, S, SRX, X)
- p1 gives the mode, p2 gives the object_id

Database Time
Concurrency

log file sync

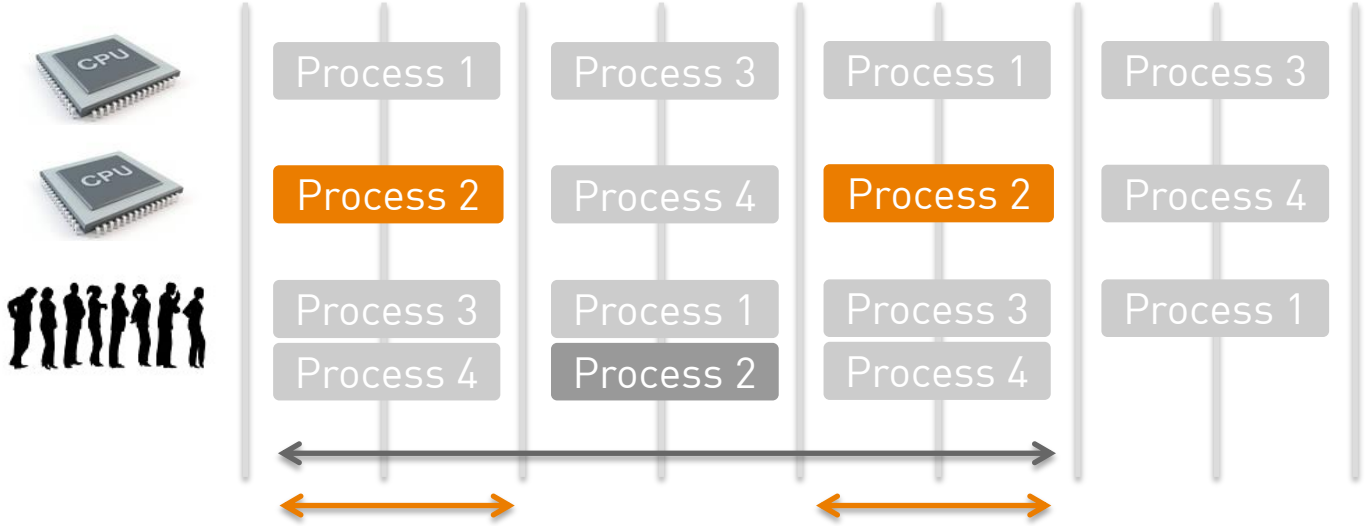
- A commit is waiting to have the redo written
- Too many redo generated, or LGWR too slow

buffer busy

- Concurrent access to same blocks
- Hot blocks (index last leaf block)
- Has a higher impact on RAC

Database Time
CPU runqueue

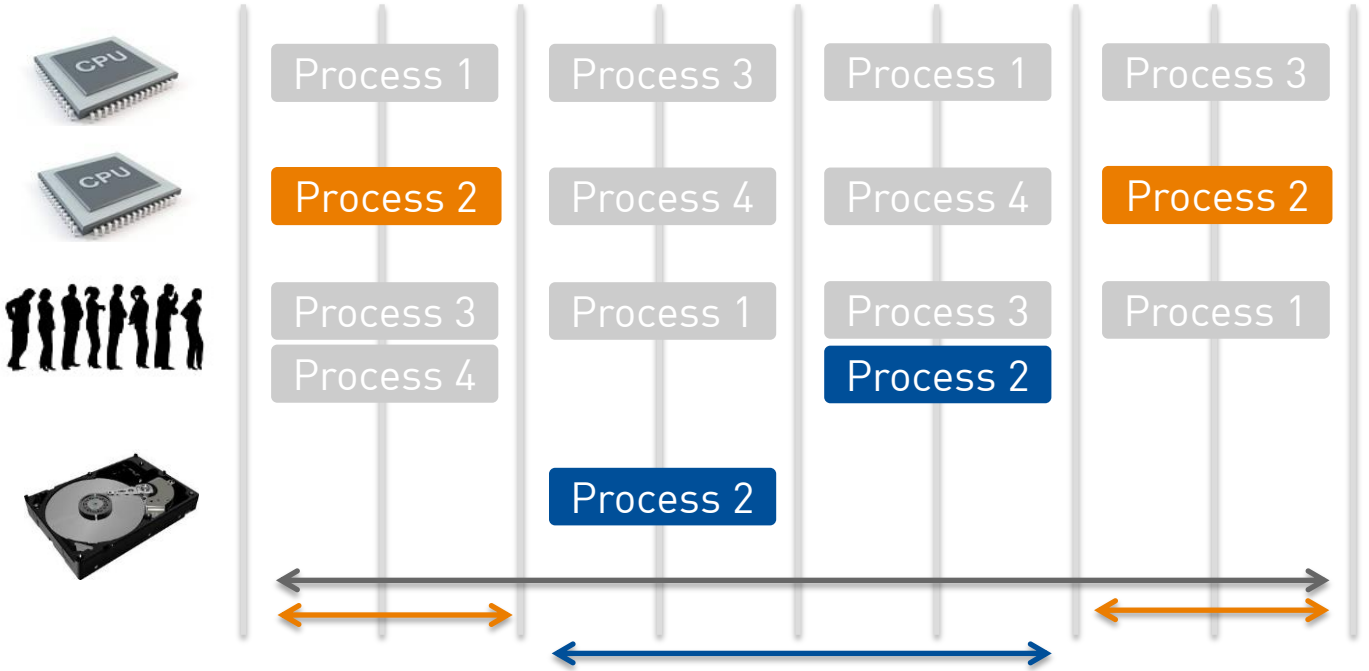
When there are more processes running than available CPU,
the system schedules CPU resource sharing:



- During the 6 time units
  - Process 2 = 4 Units (runqueue not accounted)

Database Time
CPU runqueue

When in runqueue after a system call (wait event)



- During the 8 time units
  - Process 2 = 4 Units
  - Process 2 = 4 Units (wait event inflated by runqueue)

Database Time
Host CPU

CPU utilization %

- Is calculated on threads
- You cannot run at 100%

Host CPU (CPUs: 20 Cores: 5 Sockets:)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
9.82	9.26	18.7	12.0	4.4	69.3

An example of a 75% idle system:

- IBM Power 7 with 4-way SMT
- 5 cores - 20 'CPUs'

Snap Time	Load	%busy	%user	%sys	%idle	%iowait
20-Dec 15:00:07	9.82					
20-Dec 16:00:15	12.50	26.67	14.09	12.58	73.33	6.21
20-Dec 17:00:26	15.53	41.47	24.44	17.03	58.53	5.84
20-Dec 18:00:34	6.44	36.91	25.20	11.70	63.09	4.55
20-Dec 19:00:40	6.52	24.39	15.00	9.40	75.61	3.35
20-Dec 20:00:49	9.26	24.34	14.99	9.35	75.66	1.88

- With load average of 10 all cores are overloaded despite the '75%' idle

- https://blog.dbi-services.com/cpus-cores-versus-threads-on-an-oracle-server/

Database Time
Host CPU

So the message is:

Before tuning DB Time:

- Check that DB Time = DB CPU + sum (wait events)
- Check the host CPU
- Compare load average to number of cores

Monitoring Performance

- Database Time
- AWR, ASH, ADDM
- Statspack
- dbms_xplan
- SQL Monitoring
- Application instrumentation
- SQL Trace, tkprof

AWR, ASH, ADDM
Automatic Workload Repository

AWR – Automatic Workload Repository

- First introduced with Oracle 10g
- Statistics accumulated in memory saved as snapshots in the AWR
- Provides a long term record of database state
- Depends on the STATISTICS_LEVEL instance parameter (≠ BASIC)
- Requires Diagnostics Pack

Set of tables and objects

- Stored in SYSAUX tablespace, in SYS schema

Access through Enterprise Manager

- Database control or Grid control
- But also text and html reports
- And queries on DBA_HIST_ views

AWR, ASH, ADDM
Create and configure AWR snapshots

Snapshots generated by the MMON process

- On DBA demand
- Automatically every 60 minutes. Default retention of 8 days

```
SQL> Select extract( day from snap_interval) *24*60
+ extract( hour from snap_interval) * 60
+ extract( minute from snap_interval ) "Snapshot Interval (Min.)",
       extract( day from retention) *24*60
+ extract( hour from retention) *60
+ extract( minute from retention ) "Retention (Min)"
from dba_hist_wr_control

Snapshot Interval (Min.) Retention (Min)
------------------------ ---------------
                      60           11520
```

Snapshots, metrics and baselines

- Snapshot automatically creates metrics from statistics
- Baselines are sets of snapshots, created manually and stored indefinitely for comparison across time

AWR, ASH, ADDM
Generate an AWR report

Create an AWR report with awrrpt.sql

```
SQL> @?/rdbms/admin/awrrpt.sql
…
Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
AWR reports can be generated in the following formats.  Please enter
the
name of the format at the prompt.  Default value is 'html'.

'html'           HTML format (default)
'text'           Text format
'active-html'    Includes Performance Hub active report
…
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap:
Enter value for end_snap:
…
```

AWR, ASH, ADDM
Read an AWR report

First sections
- Check report duration (must not be too long)
- Check database load (must match the user activity)

Time Model
- Check user calls

Check OS CPU utilization
- Don't count SMT or Hyper Threading as doubling core power

Foreground Timed Events
- Check most relevant ones
- A database should do CPU and I/O
- Check averages, and go to relevant sections

AWR, ASH, ADDM
ASH: Sampling approach

V$SESSION has more and more information

- About user, statement, wait event, user call
- ASH samples V$SESSION every 1 second for active foreground sessions available in V$ACTIVE_SESSION_HISTORY

AWR

- Stores 1 sample every 10 seconds in DBA_HIST_ACTIVE_SESS_HISTORY

Usage

- When Diagnostics Pack is licensed
- When investigating short issues (few seconds to few minutes)
- Easy to make load graphs (one row per sample / active session)

AWR, ASH, ADDM
ADDM: Automatic analysis

ADDM analyses two snapshots and gives

- findings
- recommendations

```
FINDING 1: 31% impact (7798 seconds)
-----------------------------------
SQL statements were not shared due to the usage of literals. This
resulted in additional hard parses which were consuming significant
database time.

RECOMMENDATION 1: Application Analysis, 31% benefit (7798 seconds)
  ACTION: Investigate application logic for possible use of bind
variables
    instead of literals. Alternatively, you may set the parameter
    "cursor_sharing" to "force".
  RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were found
to be
    using literals. Look in V$SQL for examples of such SQL statements.
```

AWR, ASH, ADDM
ADDM: Automatic analysis

Is ADDM sufficient?

```
 DETAILED ADDM REPORT FOR TASK 'TASK_145674' WITH ID 145674
          ----------------------------------------------------

              Analysis Period: 26-NOV-2014 from 09:00:22 to 13:00:55
            Database ID/Instance: 225266799/1
…
              Database Version: 10.2.0.3.0
                Snapshot Range: from 28948 to 28952
                 Database Time: 316193 seconds
          Average Database Load: 21.9 active sessions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FINDING 1: 69% impact (217858 seconds)
-----------------------------------
Individual database segments responsible for significant user I/O wait were found.

   RECOMMENDATION 1: Segment Tuning, 19% benefit (59123 seconds)
      ACTION: Investigate application logic involving I/O on TABLE
         "XXXXXP01.BASE_DOCUMENT_R" with object id 17733.
         RELEVANT OBJECT: database object with id 17733
      RATIONALE: The I/O usage statistics for the object are: 0 full object scans, 6791261 physical
reads, 2023 physical writes and 0 direct reads.
```

AWR, ASH, ADDM
Exercise

Start EM Express

- Check the port

```
SQL> select dbms_xdb_config.gethttpsport() from dual;
```

- Connect to https://192.168.22.10x:5501/em

Run some activity

```
/home/oracle/swingbench/bin/charbench -cs //192.168.22.10x:1521/APP -u soe -
p soe -uc 10 -min 5 -max 50 -a -v -rt 00:05
```

- See performance tabs

Take an AWR snapshot

- And run a report

```
SQL> exec dbms_workload_repository.create_snapshot;
SQL> !sleep 60
SQL> exec dbms_workload_repository.create_snapshot;
SQL> start ?/rdbms/admin/awrrpt
```

Monitoring Performance

- Database Time
- AWR, ASH, ADDM
- Statspack
- dbms_xplan
- SQL Monitoring
- Application instrumentation
- SQL Trace, tkprof

Statspack
Installation

Statspack is free – even on Standard Edition

Install :

```
SQL> @?/rdbms/admin/spcreate
```

- Documented in $ORACLE_HOME/rdbms/admin/spdoc.txt
- Can be installed in SYSAUX but better to
  create a STATSPACK tablespace (2GB) in case the purge is failing.

- We like to have no gap for snap_ids:

```
SQL> ALTER SEQUENCE stats$snapshot_id NOCACHE;
```

- And gather segment statistics as well:

```
SQL> exec STATSPACK.MODIFY_STATSPACK_PARAMETER (i_snap_level=>7,
i_instance_number=>null);
```

(On RAC, has to run on each instance, with the appropriate instance_number)

Statspack
Scheduling

Schedule :

- The spauto.sql doesn't schedule purges, and is not RAC aware
  dbi services recommends, for each instance, connected as perfstat:

```
variable jobno number
variable instno number
begin
  select instance_number into :instno from v$instance;
  dbms_job.submit(:jobno, 'statspack.snap;', trunc(sysdate+1/24,'HH'),
'trunc(SYSDATE+1/24,''HH'')', TRUE, :instno);
  dbms_job.submit(:jobno,
'statspack.purge(i_num_days=>45,i_extended_purge=>true);',
trunc(SYSDATE)+7, 'trunc(SYSDATE)+7', TRUE, :instno);
  commit;
end;
/
select last_sec,next_date,next_sec,log_user,what from dba_jobs;
```

- (for each instance when on a RAC)
- or even better use the dbms_scheduler:
  - https://blog.dbi-services.com/improving-statspack-experience/

Statspack
Statspack reports

Run spreport

```
SQL> @?/rdbms/admin/spreport
```

- Select the snap id

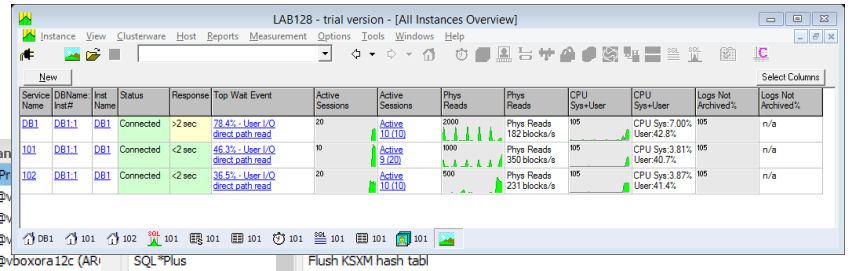
```
                                                          Snap
Instance     DB Name        Snap Id   Snap Started    Level Comment
------------ ------------ --------- ----------------- ----- ----------
DB1          DB1                  1 07 Nov 2014 14:16     7
                                  2 07 Nov 2014 14:19     7

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1
Begin Snapshot Id specified: 1

Enter value for end_snap: 2
End   Snapshot Id specified: 2
```

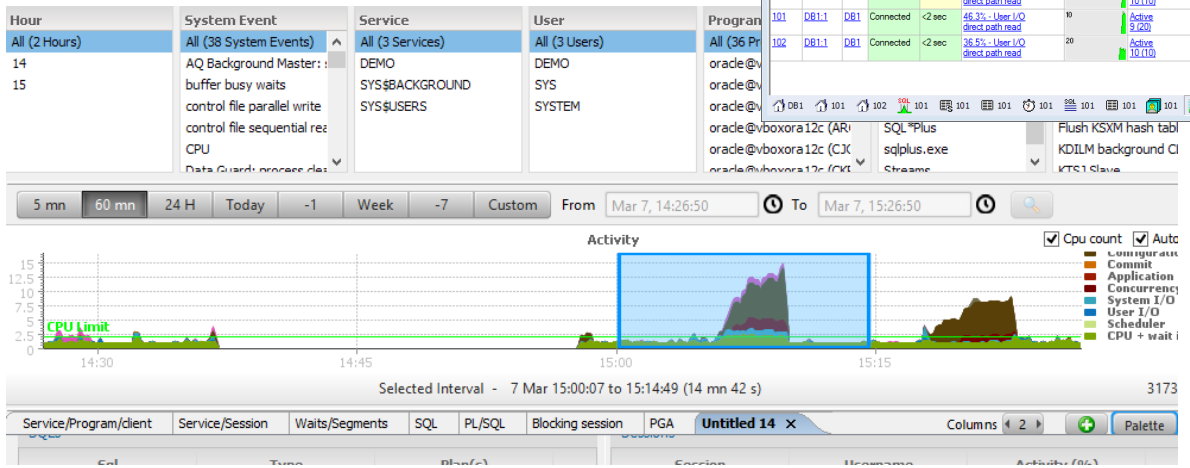
- View the report
- Consider https://blog.dbi-services.com/statspack-idle-events/

Statspack
Statspack vs AWR

Reports:

- A Statspack report is only text (AWR has html)
- Statspack has no GUI (EM shows only AWR) and no ASH
- Need to use third party tools:





- www.orachrome.com
- www.lab128.com
- etc.

Statspack
Exercise: generate Statspack report

Install Statspack

- Run spcreate.sql
- Take a first snapshot

```
SQL> exec statspack.snap(i_snap_level=>7);
```

- Schedule snapshots and purge

Run some activity

```
/home/oracle/swingbench/bin/charbench -cs //192.168.22.10x:1521/APP -u
soe -p soe -uc 10 -min 5 -max 50 -a -v -rt 00:05
```

Take another snapshot and get a report

- What is the most expensive sql statement?

AWR & Statspack
How to read an AWR report

https://prezi.com/olkkloxg-ool/soug-training-day-reading-awrstatspack-report/