PART I: INTRODUCTION TO ACTIVE SESSION HISTORY (ASH)

DAVID KURTZ
NYOUG 2021

/*Go-Faster*/ Consultancy

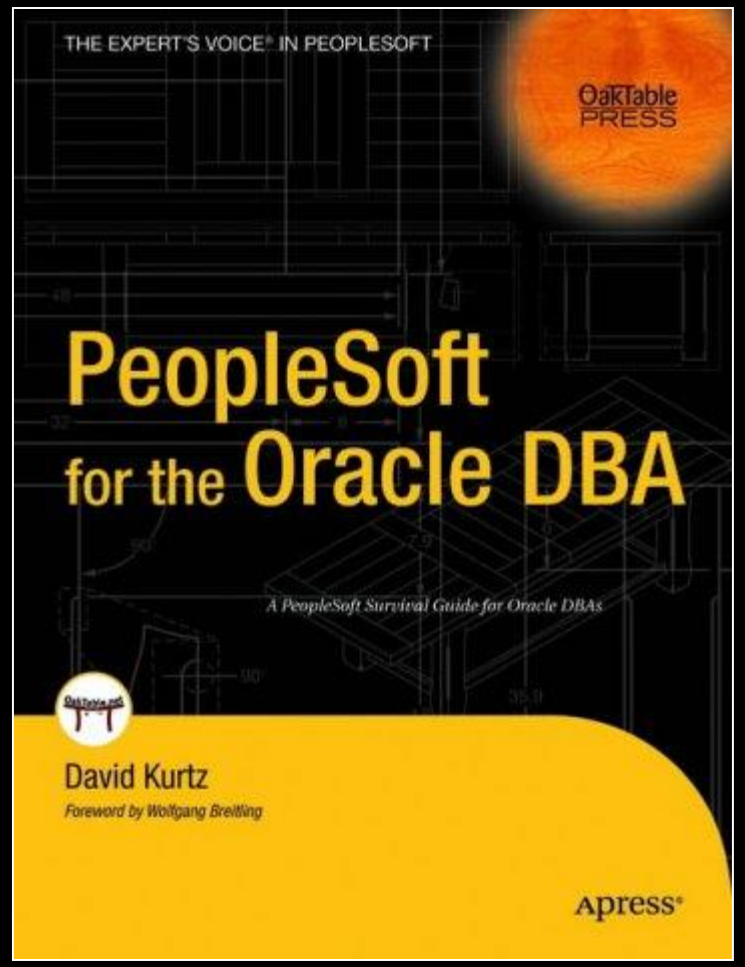
WHO AM I

Go-Faster Consultancy

- **Performance tuning**
  - **Oracle RDBMS**
  - **PeopleSoft ERP**
- www.go-faster.co.uk
- blog.go-faster.co.uk
- blog.psftdba.com

Oak Table

ORACLE ACE Director



THERE IS ONLY ONE QUESTION IN PERFORMANCE OPTIMISATION

How much time did I spent doing what?

What can I do about it?

Often, the answer is to stop doing it

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

AGENDA

ASH Fundamentals

- What is it? How does it work?
- ASH Mathematics
- ASH –v- Trace

Instrumentation

Mining the ASH data

Exact details vary depending upon application and nature of problem

Principles remain the same:

- Look for the things that consume the most time.

- Profiling DB Time
  - By SQL_ID
    - Bind variables –v- literals
  - Plan Hash Value
- Blocking Lock Analysis

ASH FUNDAMENTALS

FURTHER READING

A number of slides in this opening section have been lovingly ripped off from various presentations by Graham Wood on the subject of ASH.

- Sifting through the ASHes (2004) (https://www.oracle.com/technetwork/database/manageability/ppt-active-session-history-129612.pdf)
- ASH Architecture and Advanced Usage (2019) (https://www.youtube.com/watch?v=rxQkvXIY7X0)
- Database Time-Based Performance Tuning: From Theory to Practice (2015) (https://www.youtube.com/watch?v=Aknb_iZrPbc)

John Beresniewicz

- DB Time basics (2020) https://www.youtube.com/watch?v=xZViQXZPzx8

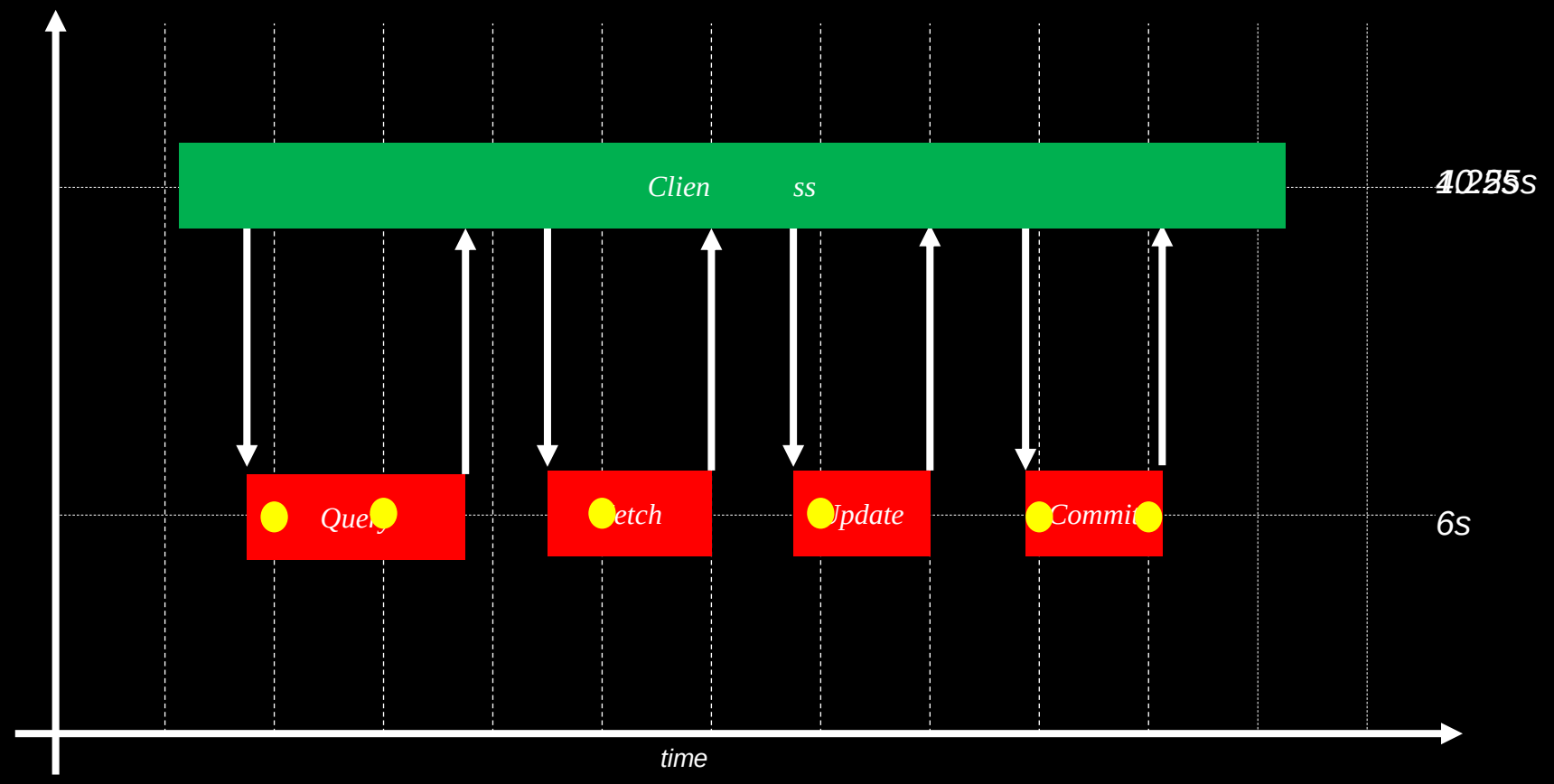
ACTIVE SESSION HISTORY IS A SAMPLING TECHNOLOGY

Sampling technology introduced in 10g

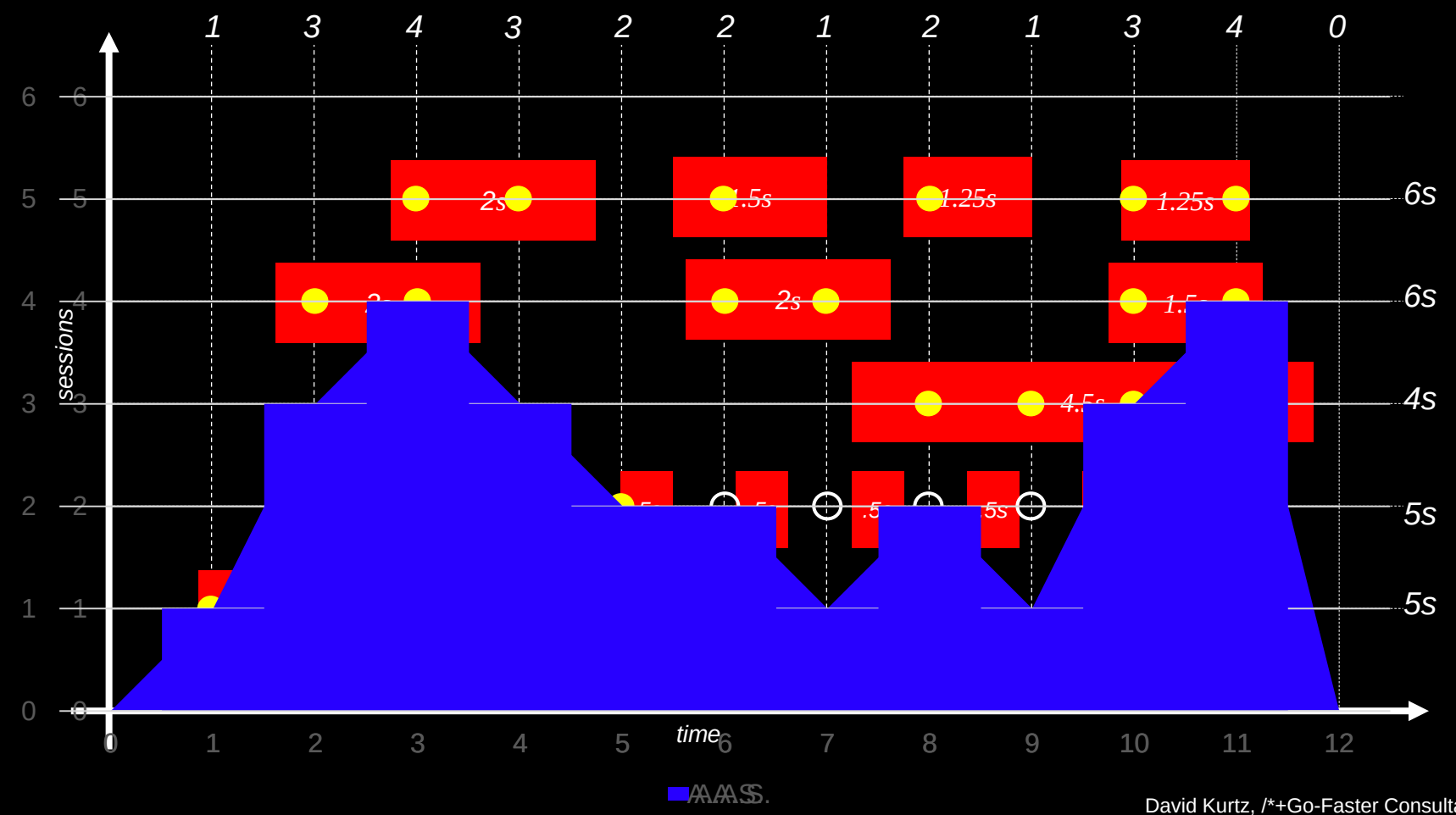
- Built into the database, so it is always on
- Low overhead, so still works well if system overloaded
- No locking
- Only sessions that are currently active are collected by the sampling process.
- Samples collected in memory
- 1 in 10 samples persisted to AWR

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

PROCESS STATES



SAMPLING DIAGRAM



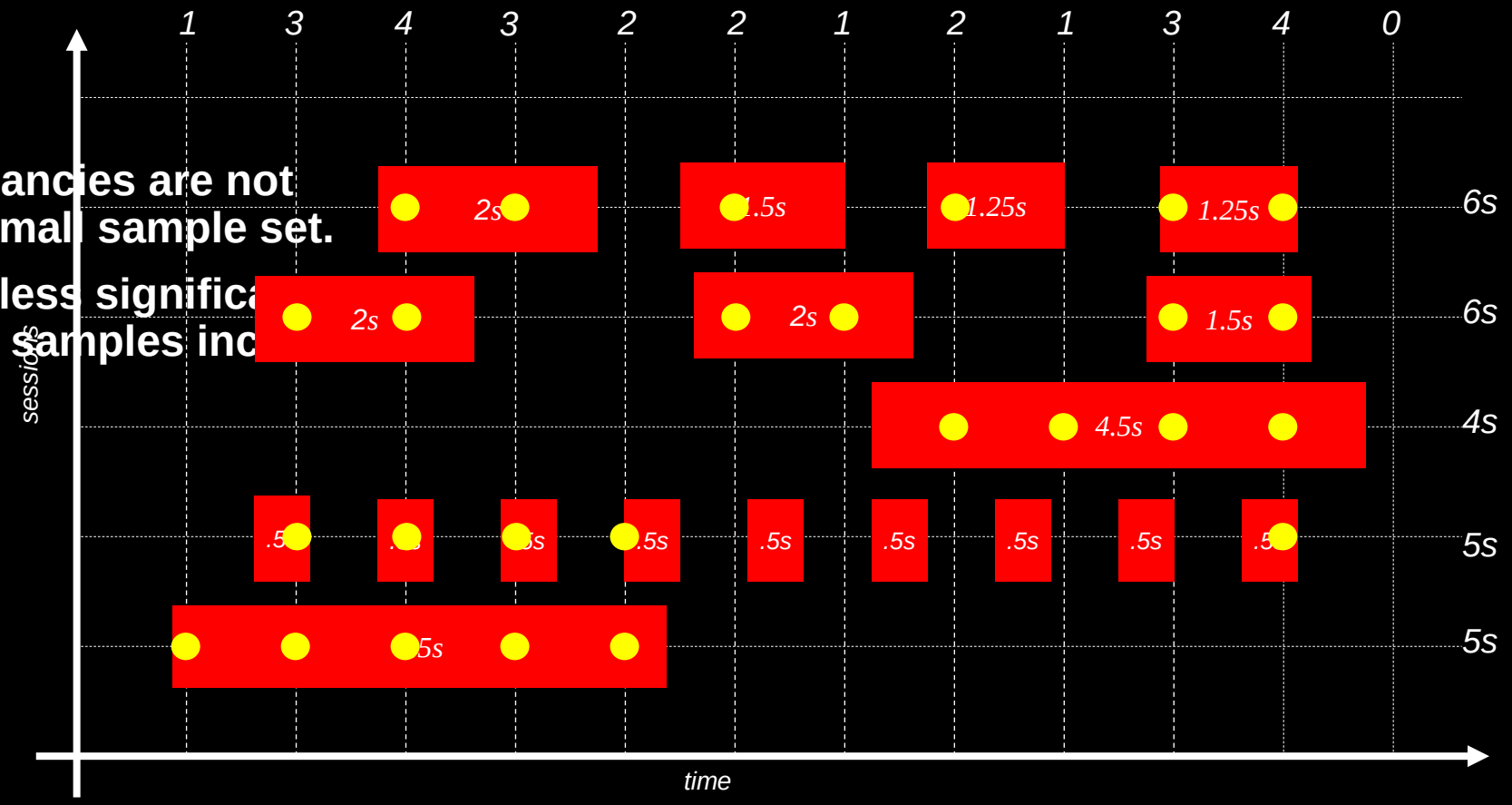
ASH SAMPLING → APPROXIMATION

26 ASH Samples ≈ 26s DB Time

Actually 24s

These discrepancies are not unusual in a small sample set.

They become less significa the number of samples inc



ASH CALCULUS

$$\frac{\partial DB\ Time}{\partial t} = Active\ Sessions$$

The number of active sessions at any time is the rate of change of the “DB Time function” at that time.

$$DB\ Time = \int_{t_0}^{t_1} Active\ Sessions.\partial t$$

DB Time is the integral of the Active Sessions function

ASH MATHEMATICS

$$DB\ Time\ \approx\ T\ .\sum Si$$

Where

- **T = duration between samples**
- **S is the set of sample**

DB Time is approximately the interval of time between each sample multiplied by the number of samples

$$DB\ Time\ \approx\ T\ .(ASH\ rows)$$

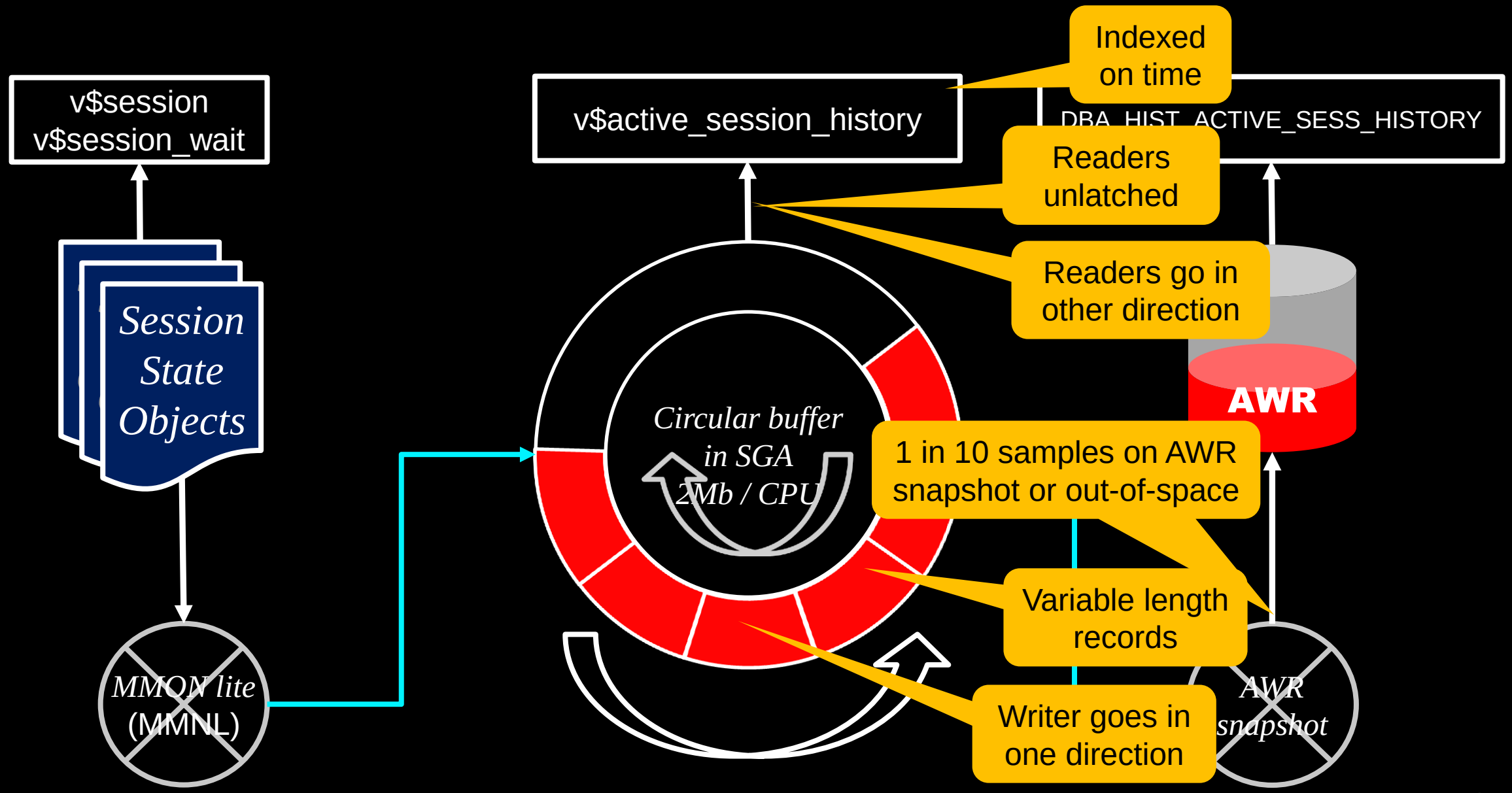
Where

- **T = duration between ASH samples**

In *v$active_session_history* each ASH sample represents 1 second.

In *DBA_HIST_ACTIVE_SESS_HISTORY* each ASH sample represents 10 second.

ASH ARCHITECTURE



ASH LICENCING

ASH is licenced as a part of the Diagnostics Pack

Only available on Enterprise Edition

- Database: $47,500/processor*
- Diagnostics: $7,500/processor*
- ≲16% of license

What if you don't have the licence?

What if you have Standard Edition

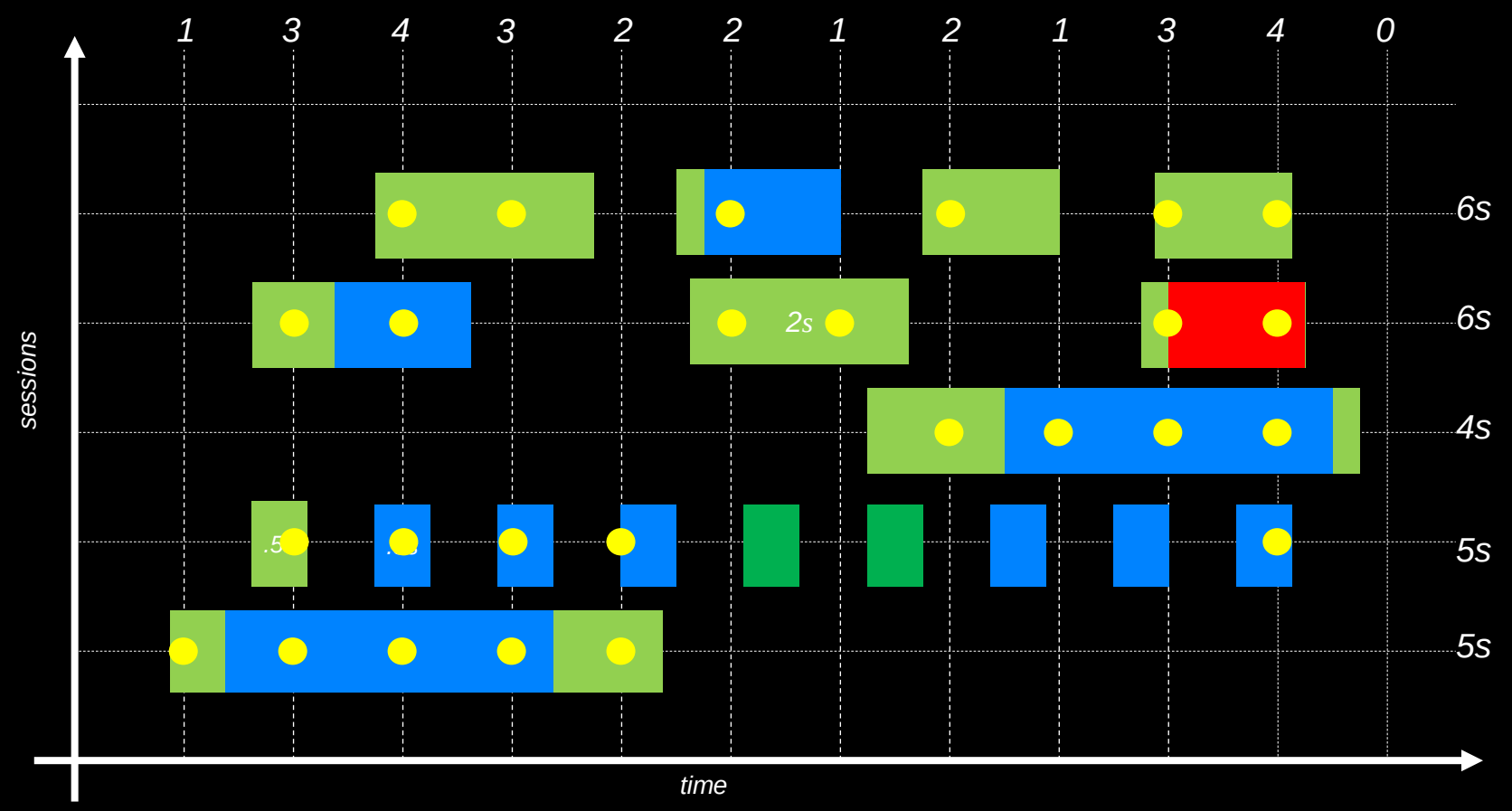
*Oracle Technology Global Price List, September 8, 2020

Conceptually, ASH is sampling and storing v$session

OraSASH or *Oracle Simulation ASH*

- Open source project
  - http://pioro.github.io/orasash/
  - Originated by Kyle Hailey
  - Maintained by Marcin Przepiorowski
- PL/SQL procedure samples *x$ksuse*
- Run from database scheduler
- Writes persisted data across database link to another database to minimize measurement intrusion effect.
- Query the data in much the same way as real ASH

ASH SAMPLING → APPROXIMATION



ASH SQL

Recent

```
SELECT …
, SUM(1) ash_secs
FROM v$active_session_history
WHERE …
GROUP BY …
ORDER BY ash_secs DESC, …
```

Historical

```
SELECT …
, SUM(10) ash_secs
FROM DBA_HIST_ACTIVE_SESS_HISTORY
WHERE …
GROUP BY …
ORDER BY ash_secs DESC, …
```

ASH SQL
NEW COLUMN FROM ORACLE 12.2

Recent

```
SELECT …
,      SUM(usecs_per_row)/1e6 ash_secs
FROM   v$active_session_history
WHERE  …
GROUP BY …
ORDER BY ash_secs DESC, …
```

usecs_per_row **= µs of database time per row**
~1.02s/row

(documented from 21c)

Historical

```
SELECT …
,      SUM(usecs_per_row)/1e6 ash_secs
FROM   dba_hist_active_sess_history
WHERE  …
GROUP BY …
ORDER BY ash_secs DESC, …
```

~10.2s/row

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

ASH –V- SQL*TRACE

ASH

- Licensed
- Always there
- No marginal cost
- Real Time, Proactive
- Statistical data
- Who is blocking me?
- SQL and plan if captured by AWR
- Estimate of duration
  - Per wait event
  - Per Operation

SQL*Trace

- Free
- Enable, Collect file, Profile
- Run-time overhead
- Reactive
- Every SQL & database call
- Being Blocked
- Actual SQL and execution plan
- Exact duration
  - Individual waits on events
  - Operations in Plan

ASH –V- SQL*TRACE

ASH can be used to resolve most of your performance issues.

Sometimes, you will still need SQL*Trace

ASH FUNDEMENTALS

QUESTIONS?

ASH: A DATA WAREHOUSE OF PERFORMANCE METRICS

ASH DIMENSIONS & ATTRIBUTES

- **Session**
  - **Session ID and serial**
  - **query coordinator**
  - **Container**
  - **in flags**

connection, parse, exec, bind, PL/SQL, Java, close, sequence, inmemory, tablespace encryption

- **Wait**
  - **Class, event id, name and parameters**
  - **Blocking session ID, instance and status**
- **Application**
  - **program, machine, port,**
  - **module, action, client_id, ECID …**
- **SQL**
  - **SQL_ID, Op Code, plan hash, plan line id, top level SQL ID, full plan hash value, exec ID**
  - **Transaction ID**
  - **force matching signature**
  - **PGA and Temp usage**
- **Object**
  - **object, file, block and row number**

WHERE IS ASH REPORTED?

Enterprise Manager / Enterprise Manager Express

AWR Report

- **including ASH Report within AWR Report**

ASH Report

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

ORACLE Enterprise Manager Database Express

ORACLE / oracle (19.3.0.0.0) Performance

SYSTEM

Performance Hub

No Auto-Refresh | Last hour: Aug 22, 2019 12:29 ...

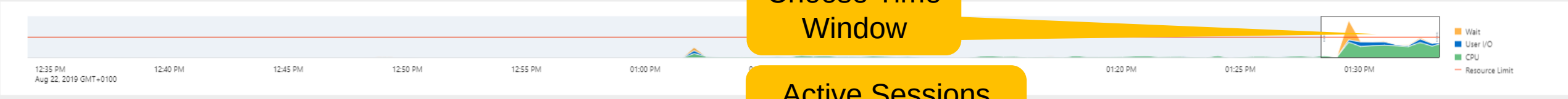
Choose Time Window

Active Sessions graph by attribute

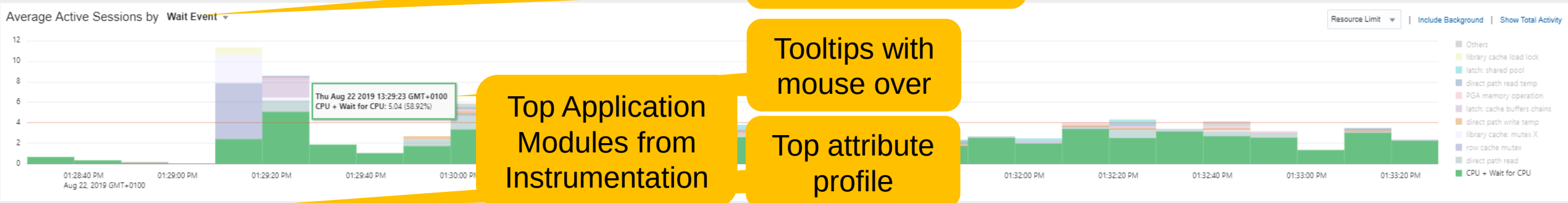
Tooltips with mouse over

Top Application Modules from Instrumentation

Top attribute profile



Activity | Monitored SQL



SQL Force Matching Signature | By Wait Event

SQL Force Matching Signature	Activity (Average Active Sessions)
N/A	0.44
7481052549858045042	0.36
1611957513104058447	0.32
15147525133108974074	0.32
447989438519293944	0.14
17141221751421062345	0.09
13909285804998054121	0.09
5461924381960967942	0.06
7059118086352512997	0.04
7913840157300941739	0.03
Others	1.03

Module | By Wait Event

Module	Activity (Average Active Sessions)
JDBC Thin Client	0.37
TopSalesWithinQuarter	0.36
SalesMovingAverage	0.34
SalesRollupByMonth	0.33
SalesByQuarterCountry	0.33
SalesByWeekCountry	0.33
SalesCubeByMonth	0.29
SalesRollupByWeek	0.24
ProductSalesRollupCube	0.11
SalesCubeByWeek	0.09
Others	0.14

WORKLOAD REPOSITORY report for

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
ORACLE	1884370030	oracle	PRIMARY	EE	19.0.0.0.0	NO	NO

Instance	Inst Num	Startup Time
oracle	1	19-Aug-19 11:55

Host Name	Platform	CPUs	Cores	Sockets
oracle-database.local	Linux x86 64-bit	4	4	1

	Snap Id	Snap Time	Sessions	Curs
Begin Snap:	1734	22-Aug-19 12:30:03	163	
End Snap:	1735	22-Aug-19 12:45:08	[illegible]	
Elapsed:		15.09 (mins)		
DB Time:		88.03 (mins)		

For pair of AWR snapshots

ASH Report in AWR Report

Report Summary

Top ADDM Findings by Average Active Sessions

Finding Name	Avg active sessions of the task	Percent active sessions of finding	Task Name	Time	Time
Hard Parse Due to Literal Usage	5.84	27.57	ADDM:1884370030_1_1735	22-Aug-19 12:30	22-Aug-19 12:45
Undersized PGA	5.84	12.58	ADDM:1884370030_1_1735	22-Aug-19 12:30	22-Aug-19 12:45
CPU Usage	5.84	10.70	ADDM:1884370030_1_1735	22-Aug-19 12:30	22-Aug-19 12:45
Undersized SGA	5.84	2.37	ADDM:1884370030_1_1735	22-Aug-19 12:30	22-Aug-19 12:45
Unusual "Concurrency" Wait Event	5.84	2.02	ADDM:1884370030_1_1735	22-Aug-19 12:30	22-Aug-19 12:45

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	5.8	135.4	0.02	0.02
DB CPU(s):	2.4	55.0	0.01	0.01
Background CPU(s):	0.0	0.7	0.00	0.00
Redo size (bytes):	5,055.1	117,344.6		
Logical read (blocks):	85,917.0	1,994,412.6		
Block changes:	22.4	519.9		
Physical read (blocks):	3,390.3	78,700.1		
Physical write (blocks):	807.2	18,738.5		
Read IO requests:	444.4	10,315.7		
Write IO requests:	111.7	2,591.7		
Read IO (MB):	26.5	614.8		
Write IO (MB):	6.3	146.4		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
User calls:	250.2	5,806.9		
Parses (SQL):	95.6	2,218.7		
Hard parses (SQL):	48.7	1,131.0		
SQL Work Area (MB):	156.1	3,622.8		
Logons:	0.1	1.4		
User logons:	0.0	0.0		
Executes (SQL):	276.2	6,410.7		
Rollbacks:	0.0	0.0		
Transactions:	0.0			

Instance Efficiency Percentages (Target 100%)

Active Session History (ASH) Report

- Top SQL with Top Events
- Top SQL with Top Row Sources
- Top Sessions
- Top Blocking Sessions
- Top PL/SQL Procedures
- Top Events
- Top Event P1/P2/P3 Values
- Top DB Objects
- Activity Over Time

Back to Top

Top SQL with Top Events

- Top SQL statements by DB Time along with the top events by DB Time for those SQLs.
- % Activity is the percentage of DB Time due to the SQL.
- % Event is the percentage of DB Time due to the event that the SQL is waiting on.
- % Row Source is the percentage of DB Time due to the row source for the SQL waiting on the event.
- Executions is the number of executions of the SQL that were sampled in ASH.

SQL ID	Plan Hash	Executions	% Activity	Event	% Event	Top Row Source	% Row Source	SQL Text
6gqww61rvkbrx	2841050720	11	1.89	CPU + Wait for CPU	1.54	HASH - GROUP BY	0.34	SELECT * FROM (SELECT times.ca...
gmq00as592t0q	2841050720	7	1.20	CPU + Wait for CPU	0.86	HASH - GROUP BY	0.34	SELECT * FROM (SELECT times.ca...
75wjjk1a92xhr	2841050720	6	1.03	CPU + Wait for CPU	0.51	HASH - GROUP BY	0.17	SELECT * FROM (SELECT times.ca...
8vcsth77yk825	2841050720	6	1.03	direct path read temp	0.51	HASH JOIN	0.51	SELECT * FROM (SELECT times.ca...

Back to Active Session History (ASH) Report
Back to Top

Top SQL with Top Row Sources

- Top SQL statements by DB Time along with the top row sources by DB Time for those SQLs.
- % Activity is the percentage of DB Time due to the SQL.
- % Row Source is the percentage of DB Time spent on the row source by that SQL.
- % Event is the percentage of DB Time spent on the event by the SQL executing the row source.
- Executions is the number of executions of the SQL that were sampled in ASH.

SQL ID	Plan Hash	Executions	% Activity	Row Source	% Row Source	Top Event	% Event	SQL Text
6gqww61rvkbrx	2841050720	11	1.89	HASH JOIN	0.69	CPU + Wait for CPU	0.34	SELECT * FROM (SELECT times.ca...
gmq00as592t0q	2841050720	7	1.20	HASH - GROUP BY	0.34	CPU + Wait for CPU	0.34	SELECT * FROM (SELECT times.ca...
75wjjk1a92xhr	2841050720	6	1.03	HASH JOIN	0.69	direct path read temp	0.34	SELECT * FROM (SELECT times.ca...
8vcsth77yk825	2841050720	6	1.03	HASH JOIN	0.86	direct path read temp	0.51	SELECT * FROM (SELECT times.ca...

Back to Active Session History (ASH) Report
Back to Top

Top Sessions

- '# Samples Active' shows the number of ASH samples in which the session was found waiting for that particular event. The percentage shown in this column is calculated with respect to wall clock time
- 'XIDs' shows the number of distinct transaction IDs sampled in ASH when the session was waiting for that particular event

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
608,46322	2.23	CPU + Wait for CPU	1.54	SH	JDBC Thin Client	9/91 [10%]	0
46,55800	2.06	CPU + Wait for CPU	1.72	SH	JDBC Thin Client	10/91 [11%]	0
620,16933	2.06	CPU + Wait for CPU	1.54	SH	JDBC Thin Client	9/91 [10%]	0
18,42515	1.72	CPU + Wait for CPU	1.03	SH	JDBC Thin Client	6/91 [7%]	0
32,51797	1.72	CPU + Wait for CPU	1.37	SH	JDBC Thin Client	8/91 [9%]	0

Back to Active Session History (ASH) Report
Back to Top

Top Blocking Sessions

ASH Report For ORACLE/oracle

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORACLE	1884370030	oracle	1	19.0.0.0.0	NO	oracle-database.local

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size	In-memory Area Size
4	1,000M (100%)	80M (8.0%)	512M (51.2%)	8.0M (0.8%)	0.0M (0.0%)

	Sample Time	Data Source
Analysis Begin Time:	22-Aug-19 12:29:00	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	22-Aug-19 12:34:00	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	5.0 (mins)	
Sample Count:	1,048	
Average Active Sessions:	3.49	
Avg. Active Session per CPU:	0.87	
Report Target:	None specified	

Specific time window

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top PL/SQL
- Top Java
- Top Call Types
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

Back to Top

Top Events

- Top User Events
- Top Background Events
- Top Event P1/P2/P3 Values

Back to Top

Top User Events

Event	Event Class	% Event	Avg Active Sessions
CPU + Wait for CPU	CPU	69.56	2.43
direct path read	User I/O	8.97	0.31
row cache mutex	Concurrency	5.73	0.20
library cache: mutex X	Concurrency	3.24	0.11
direct path write temp	User I/O	2.10	0.07

Back to Top Events
Back to Top

Top Background Events

Event	Event Class	% Activity	Avg Active Sessions
CPU + Wait for CPU	CPU	1.91	0.07

Back to Top Events
Back to Top

Top Event P1/P2/P3 Values

Event	% Event	P1 Value, P2 Value, P3 Value	% Activity	Parameter 1	Parameter 2	Parameter 3
direct path read	8.97	"1","59225","7"	0.48	file number	first dba	block cnt
row cache mutex	5.73	"10","19","0"	5.73	cache id	where requested	NOT DEFINED

Top Service/Module

Service	Module	% Activity	Action	% Action
oracle	TopSalesWithinQuarter	11.93	UNNAMED	11.93
	SalesMovingAverage	11.07	UNNAMED	11.07
	SalesByWeekCountry	10.97	UNNAMED	10.97
	SalesCubeByMonth	10.97	UNNAMED	10.97
	JDBC Thin Client	10.50	UNNAMED	10.50

Back to Load Profile
Back to Top

Top Client IDs

No data exists for this section of the report.

Back to Load Profile
Back to Top

Top SQL Command Types

- 'Distinct SQLIDs' is the count of the distinct number of SQLIDs with the given SQL Command Type found over all the ASH samples in the analysis period

SQL Command Type	Distinct SQLIDs	% Activity	Avg Active Sessions
SELECT	733	84.16	2.94
PL/SQL EXECUTE	5	1.24	0.04

Back to Load Profile
Back to Top

Top Phases of Execution

Phase of Execution	% Activity	Avg Active Sessions
SQL Execution	62.12	2.17
Parse	23.76	0.83
Hard Parse	22.52	0.79
Connection Management	10.50	0.37

Back to Load Profile
Back to Top

Top SQL

- Top SQL with Top Events
- Top SQL with Top Row Sources
- Top SQL using literals
- Top Parsing Module/Action
- Complete List of SQL Text

Back to Top

Top SQL with Top Events

SQL ID	FullPlanhash	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% RwSrc	SQL Text	Container Name
2bqd4t16hkyht	1669843298	2841050720	12	1.15	CPU + Wait for CPU	0.86	HASH JOIN	0.38	SELECT * FROM (SELECT times.ca...	oracle
3br1m0c6mfq15	1669843298	2841050720	12	1.15	CPU + Wait for CPU	0.86	HASH - GROUP BY	0.38	SELECT * FROM (SELECT times.ca...	oracle
8vcsth77yk825	1669843298	2841050720	12	1.15	CPU + Wait for CPU	0.76	HASH JOIN	0.38	SELECT * FROM (SELECT times.ca...	oracle

Back to Top SQL
Back to Top

Top SQL with Top Row Sources

ASH DATA IN OEM

You can run ASH reports via EM

Filter by Module

EXAMPLE ASH REPORT

These processes were responsible for 86% of total DB activity

Average 14.8 active sessions (out 32 processes)

If I go on I get SQL statements

But I don’t get execution plans.

ASH Report For XXXXXXXX/XXXXXXXX

(1 Report Target Specified)

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
XXXXXXXX	4200535484	XXXXXXXX	1	10.2.0.4.0	NO	xxxxr03

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
4	13,312M (100%)	11,440M (85.9%)	1,252M (9.4%)	40.5M (0.3%)

	Sample Time	Data Source
Analysis Begin Time:	21-Feb-10 03:01:49	DBA_HIST_ACTIVE_SESS_HISTORY in AWR snapshot 64162
Analysis End Time:	21-Feb-10 04:51:49	DBA_HIST_ACTIVE_SESS_HISTORY in AWR snapshot 64169
Elapsed Time:	110.0 (mins)	
Sample Count:	9,765	
Average Active Sessions:	14.80	
Avg. Active Session per CPU:	0.[illegible]	
Report Target:	MODULE like 'GPPDPRUN'	86% of total database activity

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top PL/SQL
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

DATA WAREHOUSE OF PERFORMANCE METRICS

Keep it

- **7-42 days locally**
- **Copy to central repository**
  - **Retain for 400 days**
  - ***awrextr.sql / awrload.sql***
  - **OEM 12c4 manage this**
- **Compare different databases**
  - **Eg. artefacts common to multiple databases might indicate an infrastucture issue?**

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

INSTRUMENTATION, INSTRUMENTATION, INSTRUMENTATION.

APPLICATION INSTRUMENTATION

It is essential to be able to match

- database sessions
- application processes

DBMS_APPLICATION_INFO

- *set_module, set_action*
  - Set session module and action attributes
  - Picked up by ASH, AWR and session trace

- Put calls in application
  - Few packaged application vendors do this

PL/SQL INSTRUMENTATION

```
k_module      CONSTANT VARCHAR2(64) := $$PLSQL_UNIT;
…
PROCEDURE my_procedure IS
  l_module VARCHAR2(64);
  l_action VARCHAR2(64);
BEGIN
  dbms_application_info.read_module(module_name=>l_module
                                   ,action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module
                                  ,action_name=>'MY_PROCEDURE');
…
  dbms_application_info.set_module(module_name=>l_module
                                  ,action_name=>l_action);
EXCEPTION
  WHEN … THEN
    dbms_application_info.set_module(module_name=>l_module
                                    ,action_name=>l_action);
    RAISE / EXIT
…
END my_procedure;
```

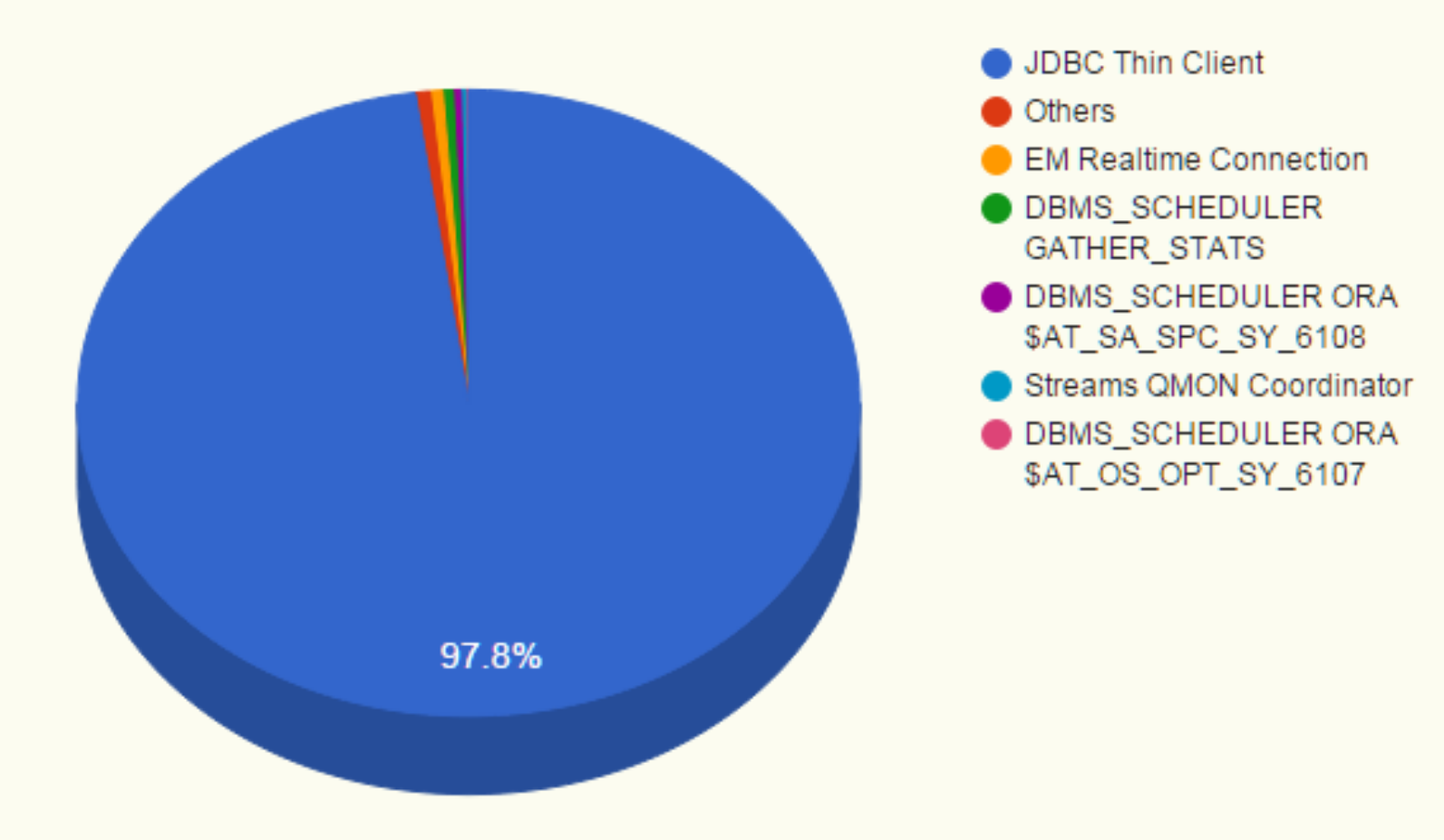
DIY INSTRUMENTATION

You may need to be creative!

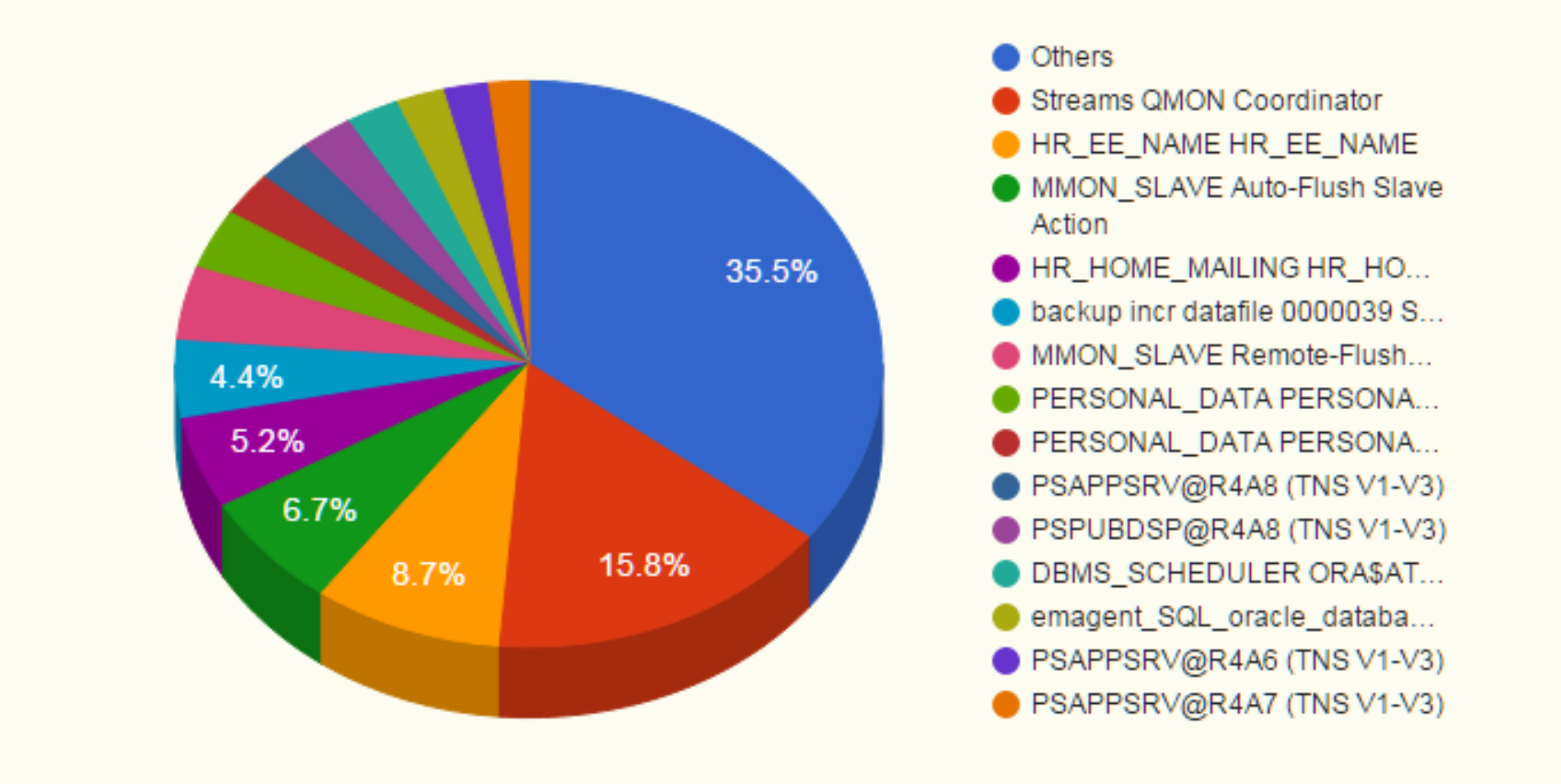
- PeopleTools: When a process starts (and sets its own status), I have a trigger that calls a PL/SQL API that sets module and action

```
CREATE OR REPLACE TRIGGER sysadm.psftapi_store_prcsinstance
BEFORE UPDATE OF runstatus ON sysadm.psprcsrqst FOR EACH ROW
WHEN ((new.runstatus IN('3','7','8','9','10') OR old.runstatus IN('7','8')) AND new.prcstype
!= 'PSJob')
BEGIN
…
  psftapi.set_action(p_prcsinstance=>:new.prcsinstance
   ,p_runstatus=>:new.runstatus, p_prcsname=>:new.prcsname);
…
EXCEPTION WHEN OTHERS THEN NULL; --do not crash the scheduler
END;
/
```

NO INSTRUMENTATION!



WITH INSTRUMENTATION!



INSTRUMENTATION, INSTRUMENTATION, INSTRUMENTATION.

QUESTIONS

ASH MINING

WHAT ARE YOU LOOKING FOR?

You need a clear idea of the question you are asking of the ASH data.

What are you interested in?

- **Time Window**
  - **Recent –v- Historical**
- **Single Session / Group of Sessions / Whole Database**
- **All ASH Data / One Event / One SQL ID / One Plan**
- **Related ASH data (sessions blocked by lock)**

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

PROFILE DB TIME BY DIFFERENT ATTRIBUTE TO FIND TOP SQLS

- SQL_ID
  - FORCE_MATCHING_SIGNATURE – to treat literals as binds
- TOP_LEVEL_SQL_ID – PL/SQL call

- SQL_PLAN_HASH_VALUE
  - SQL_PLAN_LINE_ID
- SQL_FULL_PLAN_HASH_VALUE – adaptive plan

- EVENT_CLASS
- EVENT

- IN_PARSE, IN_HARD_PARSE, IN_BIND, IN_SQL_EXECUTION, IN_PLSQL_EXECUTION, IN_CURSOR_CLOSE

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

QUERYING ASH REPOSITORY

DBA_HIST_ACTIVE_SESS_HISTORY

- **WRH$_ACTIVE_SESSION_HISTORY partitioned on DBID and SNAP_ID**

DBA_HIST_SNAPSHOT

- **WRM$_SNAPSHOT**

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

QUERYING ASH REPOSITORY

```
SELECT /*+LEADING(x) USE_NL(h)*/
…
,      SUM(10) ash_secs
FROM   dba_hist_active_sess_history h
,      dba_hist_snapshot x
WHERE  x.snap_id = h.snap_id
AND    x.dbid = h.dbid
AND    x.instance_number = h.instance_number
AND    x.end_interval_time >= …
AND    x.begin_interval_time <= …
AND    h.sample_time BETWEEN … AND …
AND    …
GROUP BY …
```

EXAMPLE

```
SELECT /*+LEADING(x h) USE_NL(h)*/
              h.sql_id
              h.sql_plan_hash_value
,             SUM(10) ash_secs
FROM          dba_hist_snapshot x
,             dba_hist_active_sess_history h
WHERE         x.end_interval_time
                          >= TO_DATE('201902010730','yyyymmddhh24mi')
AND           x.begin_interval_time
                          <= TO_DATE('201902010830','yyyymmddhh24mi')
AND           h.sample_time BETWEEN TO_DATE('201902010730','yyyymmddhh24mi')
                                AND TO_DATE('201902010830','yyyymmddhh24mi')
AND           h.SNAP_id = X.SNAP_id
AND           h.dbid = x.dbid
AND           h.instance_number = x.instance_number
AND           h.module like 'PSAPPSRV%'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs DESC
/
```

Snapshots for Period for which process was running

ASH Data for period for which process was running

Application not instrumented Can Only Filter by Module

TOP SQL

```
                SQL Plan
SQL_ID          Hash Value    ASH_SECS
--------------- ----------- -----------
7hvaxp65s70qw   1051046890        1360
fdukyw87n6prc    313261966         760
8d56bz2qxwy6j   2399544943         720
876mfmryd8yv7    156976114         710
bphpwrud1q83t   3575267335         690
…
```

GET EXECUTION PLAN FROM AWR

```
SELECT * from table(
    dbms_xplan.display_awr(
        '7hvaxp65s70qw', 1051046890, NULL, 'ADVANCED')
    );
```

GET EXECUTION PLAN FROM LIBRARY CACHE

```
SELECT * from table(
    dbms_xplan.display_cursor(
        '7hvaxp65s70qw', 0, 'ADVANCED')
    );
```

USE SQL QUERY TO GENERATE CODE TO OBTAIN EXECUTION PLAN

```
SELECT DISTINCT
'SELECT * from table( dbms_xplan.display_cursor('''
||sql_id ||''','' ||sql_child_number ||','''ADVANCED'''));'
FROM (
…
)
```

WHICH PART OF EXECUTION PLAN CONSUMED THE MOST TIME?

```
SELECT …
, h.sql_plan_line_id
, sum(10) ash_secs
FROM dba_hist_snapshot x
, dba_hist_active_sess_history h
…
WHERE …
AND h.sql_id = 'a47fb0x1b23jn'
GROUP BY …
, h.sql_plan_line_id
ORDER BY ASH_SECS DESC
```

WHICH PART OF EXECUTION PLAN CONSUMED THE MOST TIME?

PRCSINSTANCE	SQL_PLAN_HASH_VALUE	SQL_PLAN_LINE_ID	ASH_SECS
4945802	483167840	25	2410
	483167840	24	1190
	483167840	26	210
	483167840	20	190
	483167840	21	30
	483167840	16	20
	483167840	23	10
	483167840	22	10
	483167840	18	10
	483167840		10
	483167840	7	10

WHICH PART OF EXECUTION PLAN CONSUMED THE MOST TIME?

```
Plan hash value: 483167840
-------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------------------------------------------------------------
…
|  14 |         NESTED LOOPS               |                     |       |       |            |          |  Q1,04 | PCWP |            |
|  15 |          NESTED LOOPS              |                     |  3988 |  669K|   113K  (1)| 00:06:08 |  Q1,04 | PCWP |            |
|  16 |           HASH JOIN SEMI           |                     |  3851 |  481K|   112K  (1)| 00:06:05 |  Q1,04 | PCWP |            |
|  17 |            PX RECEIVE              |                     |  3771K|  233M| 61175   (1)| 00:03:19 |  Q1,04 | PCWP |            |
|  18 |             PX SEND HASH           | :TQ10003            |  3771K|  233M| 61175   (1)| 00:03:19 |  Q1,03 | P->P | HASH       |
|  19 |              PX BLOCK ITERATOR     |                     |  3771K|  233M| 61175   (1)| 00:03:19 |  Q1,03 | PCWC |            |
|  20 |               TABLE ACCESS FULL    | PS_CM_DEPLETE       |  3771K|  233M| 61175   (1)| 00:03:19 |  Q1,03 | PCWP |            |
|  21 |            BUFFER SORT             |                     |       |       |            |          |  Q1,04 | PCWC |            |
|  22 |             PX RECEIVE             |                     |  6058K|  364M| 50906   (1)| 00:02:46 |  Q1,04 | PCWP |            |
|  23 |              PX SEND HASH          | :TQ10001            |  6058K|  364M| 50906   (1)| 00:02:46 |        | S->P | HASH       |
|  24 |               INDEX FULL SCAN      | PS_CM_DEPLETE_COST  |  6058K|  364M| 50906   (1)| 00:02:46 |        |      |            |
|  25 |           INDEX UNIQUE SCAN        | PS_TRANSACTION_INV  |     1 |       |     1   (0)| 00:00:01 |  Q1,04 | PCWP |            |
|  26 |          TABLE ACCESS BY INDEX ROWID| PS_TRANSACTION_INV  |     1 |    44 |     1   (0)| 00:00:01 |  Q1,04 | PCWP |            |
…
-------------------------------------------------------------------------------------------------------------------------------------
```

PROFILE BY LINE AND EVENT

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                      | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT               |                   |       |       |   318 (100)|          |
|   1 |  LOAD TABLE CONVENTIONAL       |                   |       |       |            |          |
|   2 |   FILTER                       |                   |       |       |            |          |
|   3 |    TABLE ACCESS BY INDEX ROWID| PS_PROJ_RESOURCE  |   724 |  337K|   314   (0)| 00:00:04 |
|   4 |     INDEX RANGE SCAN           | PSEPROJ_RESOURCE  |   724 |       |     9   (0)| 00:00:01 |
|   5 |    INDEX RANGE SCAN            | PS_PRJRE_SBF_TMP  |     1 |     6 |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

                                                                    ASH
SQL_PLAN_LINE_ID EVENT                                             Secs
---------------- ---------------------------------------- --------
               3 db file sequential read                      40320
               3 read by other session                        12280
               1                                              10680
               1 db file sequential read                       4320
               3                                               2800
                 db file sequential read                       1560
               4 db file sequential read                       1160
                                                                520
               4                                                240
               4 read by other session                          200
                 read by other session                           80
               5 db file sequential read                         40
                                                            --------
sum                                                           74200
```

ASH MINING

QUESTIONS?

BIND VARIABLES

BIND VARIABLES

Developers Should Use Them!

Unfortunately…

DIFFERENT SQL, SAME PLAN

P.I.	SQL_ID	SQL_PLAN_HASH_VALUE	EXEC_SECS	ASH_SECS
1949129	0uj7k70z1s76y	2239378934	619	210
1949819	0sd03jvun7us6	2239378934	336	20
1953197	22kn2sb7vttnp	2239378934	753	150
1956338	0xkjtywub2861	2602481067	19283	18550
1956338	998wf4g84dk8z	1041940423	19283	10
1956805	7c7dzavm70yku	2602481067	16350	15690
1956925	1knvx57dnrz29	2602481067	15654	15010
1956925	a9mw8hjxfwczm	338220129	15654	10
1957008	9s2jct0jfmwgy	2602481067	15077	14430
1957008	9s2jct0jfmwgy	3265949623	15077	10
1957087	cwarnq7kv4d84	2602481067	14638	14000
1957691	9nv93p134xjb0	2602481067	13477	12980
1958659	9s2jct0jfmwgy	2602481067	9354	9140
1958697	1bd0fg0fvsfyp	2602481067	9176	8950

NON-SHAREABLE SQL

```
SQL_ID djqf1zcypm5fm
--------------------
SELECT ...
FROM PS_TL_EXCEPTION A,
     PS_PERSONAL_DATA B,
     PS_PERALL_SEC_QRY B1,
…
WHERE B.EMPLID = B1.EMPLID
AND B1.OPRID = '12345678'
…
```

SAME PROBLEM IN ONE PROCESS

PRCSINSTANCE	SQL_ID	SQL_PLAN_HASH_VALUE	EXEC_SECS	ASH_SECS
50002824		0	10306	50
50002824	2ybtak62vmx58	2262951047	10306	20
50002824	ck3av6cnquwfc	2262951047	10306	20
50002824	gvys6kd9fqn7u	2262951047	10306	20
50002824	7ymcbn6q8utj8	2262951047	10306	10
50002824	9qud2n3qq7nzr	2262951047	10306	10
50002824	6pxvns97m1fua	2262951047	10306	10
50002824	5ngqj5zg8vbz8	2262951047	10306	10
50002824	9zp6nndfvn66b	2262951047	10306	10
50002824	15kfs3c3005xm	2262951047	10306	10
50002824	4qvhpygc7cq2t	2262951047	10306	10
50002824	23yc8dcz9z4yj	2262951047	10306	10
50002824	bn8xczrvs2hpr	2262951047	10306	10
50002824	9g6k9dnrjap08	2262951047	10306	10
50002824	1art8dhzbvpwt	2262951047	10306	10
50002824	6gqj337xnr5y4	2262951047	10306	10

...

AGGREGATE BY SQL_PLAN_HASH_VALUE

PRCSINSTANCE	SQL_PLAN_HASH_VALUE	EXEC_SECS	ASH_SECS
50002824	2262951047	10306	2300
50002824	0	10306	60
50002824	3085938243	10306	20
50002824	563410926	10306	10
50002824	1068931976	10306	10

CAN YOU FIND THE SQL IN AWR?

Now Find the SQL with that plan.

If it was captured by AWR

- **Lots of parsing causes statements to be aged out of library cache before they get stored in AWR by a snapshot**
- **Only Top-n statements are captured.**

David Kurtz, /*+Go-Faster Consultancy*/ © 2021 www.go-faster.co.uk

CAN YOU FIND THE SQL IN AWR?

```
SELECT h.sql_id, h.sql_plan_hash_value
,      SUM(10) ash_secs
,      10*COUNT(t.sql_id) awr_secs
FROM   dba_hist_snapshot x
,      dba_hist_active_sess_history h
       LEFT OUTER JOIN dba_hist_sqltext t
         ON t.sql_id = h.sql_id
WHERE        …
GROUP BY h.sql_id, h.sql_plan_hash_value
```

CAN YOU FIND THE SQL IN AWR?

```
WITH x AS (
SELECT h.sql_id, h.sql_plan_hash_value
,      SUM(10) ash_secs
,      10*COUNT(t.sql_id) awr_secs
FROM   dba_hist_snapshot X
,      dba_hist_active_sess_history h
       LEFT OUTER JOIN dba_hist_sqltext t ON t.sql_id = h.sql_id
WHERE            …
GROUP BY h.sql_id, h.sql_plan_hash_value
), y as (
SELECT ROW_NUMBER() OVER (PARTITION BY x.sql_plan_hash_value
                          ORDER BY x.awr_secs DESC, x.sql_id) ranking
,      x.sql_id, x.sql_plan_hash_value
,      SUM(x.ash_secs) OVER (PARTITION BY x.sql_plan_hash_value) ash
,      SUM(x.awr_secs) OVER (PARTITION BY x.sql_plan_hash_value) awr
,      COUNT(DISTINCT sql_id) OVER (PARTITION BY x.sql_plan_hash_value) sql_ids
FROM   x
)
SELECT * FROM y
WHERE    y.ranking = 1
ORDER BY tot_ash_secs desc, ranking
```

CAN YOU FIND THE SQL IN AWR?

RNK	SQL_ID	SQL_PLAN HASH_VALUE	TOTAL ASH_SECS	TOTAL AWR_SECS	SQL_IDS
1	8mkvraydrxycn	0	38270	480	74[1]
1	027qsfj7n71cy	1499159071	4230	4230	1[2]
1	cxwz9m3auk4y7	1898065720	4190	4190	198[3]
1	9513hhu1vucxz	2044891559	3590	3590	1
1	95dx0mkjq38v5	1043916244	3450	3450	23

1. Special case. There is no plan because it's the *dbms_stats* function. There were 74 statements, but in reality they were all totally different. PHV=0 indicates PL/SQL or *INSERT…VALUES*.
2. One SQL, one plan. Either this is a shareable SQL_ID, or it just executed once.
3. This is many statements with the same plan, at least 198, possibly more.

BIND VARIABLES

QUESTIONS?

LOCK ANALYSIS WITH ASH

WHERE DID WE WAIT ON A LOCK?

```
SELECT   /*+LEADING(x h) USE_NL(h)*/
         h.sql_id
,        h.sql_plan_hash_value
,        SUM(10) ash_secs
FROM     dba_hist_snapshot x
,        dba_hist_active_sess_history h
WHERE    x.end_interval_time >=TO_DATE('201901261100','yyyymmddhh24mi')
AND      x.begin_interval_time <=
                               TO_DATE('201901261300','yyyymmddhh24mi')
AND      h.sample_time BETWEEN TO_DATE('201901261100','yyyymmddhh24mi')
AND                            TO_DATE('201901261300','yyyymmddhh24mi')
AND      h.snap_id = x.snap_id
AND      h.dbid = x.dbid
AND      h.instance_number = x.instance_number
AND      h.event = 'enq: TX - row lock contention'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/
```

WHERE DID WE WAIT ON A LOCK?

```
                 SQL Plan
SQL_ID          Hash Value   ASH_SECS
--------------- ----------- -----------
7qxdrwcn4yzhh 3723363341       26030
652mx4tffq415 1888029394       11230
c9jjtvk0qf649 3605988889        6090
artqgxug4z0f1    8450529         240
gtj7zuzy2b4g6 2565837323         100
```

STATEMENTS BLOCKED BY TX LOCKS

```
SQL_ID 7qxdrwcn4yzhh
--------------------
UPDATE PSIBQUEUEINST SET QUEUESEQID=QUEUESEQID+:1
WHERE QUEUENAME=:2

SQL_ID 652mx4tffq415
--------------------
UPDATE PSAPMSGPUBSYNC SET LASTUPDDTTM=SYSDATE
WHERE QUEUENAME=:1

SQL_ID c9jjtvk0qf649
--------------------
UPDATE PSAPMSGSUBCSYNC SET LASTUPDDTTM=SYSDATE
WHERE QUEUENAME=:1
```

THE REAL QUESTION ABOUT LOCKING:

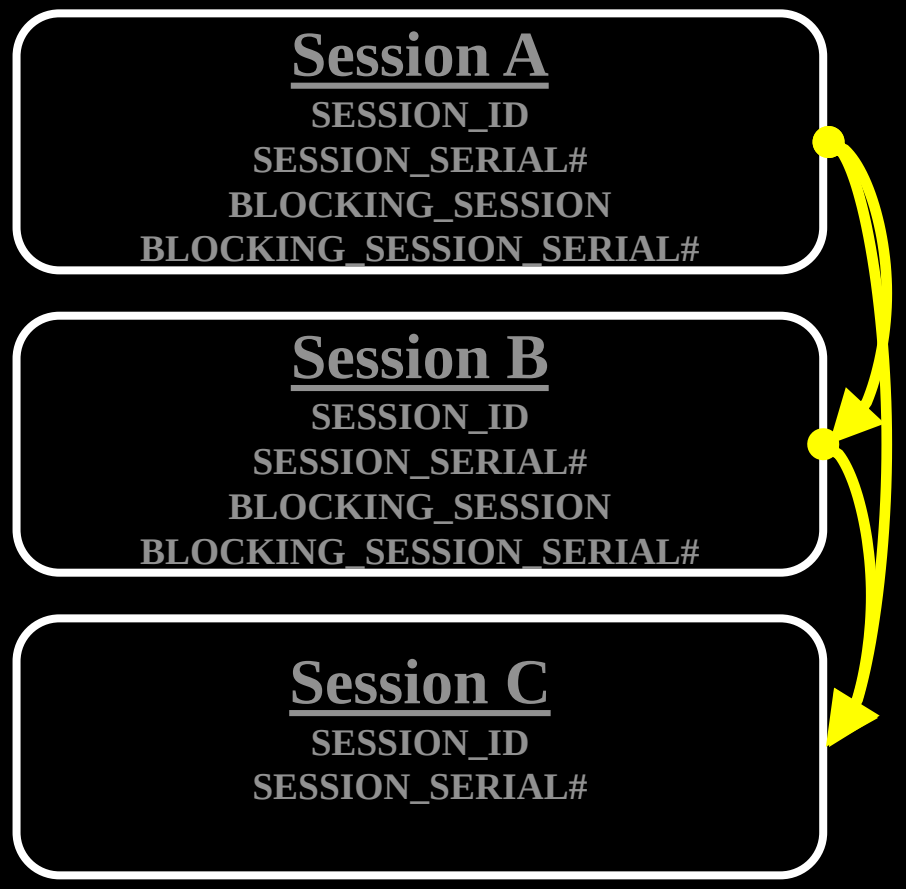
What is the session that is holding the lock doing while it is holding the lock?

- **and can I do something about that?**

Home-made sequences are not scalable. Should really be using an Oracle Sequence.

- ***Not possible in a PeopleSoft Application***

RESOLVE THE LOCK CHAIN



Navigating the lock chain works across RAC instances from 11g.

There may not be any ASH data for the ultimate blocking session (C) because it is not active on the database.

EXTRACT ASH DATA FOR PERIOD IN QUESTION

```
CREATE TABLE my_ash AS
SELECT /*+LEADING(x) USE_NL(h)*/ h.*
FROM   dba_hist_snapshot x
,      dba_hist_active_sess_history h
WHERE  x.end_interval_time >= TO_DATE('201901261100','yyyymmddhh24mi')
AND    x.begin_interval_time <= TO_DATE('201901261300','yyyymmddhh24mi')
AND    h.sample_time BETWEEN TO_DATE('201901261100','yyyymmddhh24mi')
                     AND     TO_DATE('201901261300','yyyymmddhh24mi')
AND    h.Snap_id = X.snap_id
AND    h.dbid = x.dbid
AND    h.instance_number = x.instance_number;

CREATE UNIQUE INDEX my_ash ON my_ash (dbid, instance_number, snap_id, sample_id, session_id, sample_time,
session_serial#) COMPRESS 4;
CREATE INDEX my_ash2 ON my_ash (event, dbid, instance_number, snap_id) COMPRESS 3;
```

Repeatedly update the blocking instance/session information to the next session along the chain

- **Stop when number of updates does not decrease.**

WHAT ARE THE BLOCKING SESSIONS DOING (SAME INSTANCE)?

```
SELECT /*+LEADING(x w) USE_NL(h w)*/
       h.sql_id
,      h.sql_plan_hash_value
,      SUM(10) ash_secs
FROM   my_ash w
       LEFT OUTER JOIN my_ash h
       ON  h.snap_id = w.snap_id
       AND h.dbid = w.dbid
       AND w.blocking_inst_id = w.instance_number
       AND h.sample_id = w.sample_id
       AND h.sample_time = w.sample_time
       AND h.instance_number = w.blocking_inst_id
       AND h.session_id = w.blocking_session
       AND h.session_serial# = w.blocking_session_serial#
WHERE  w.event = 'enq: TX - row lock contention'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/
```

WHAT ARE THE BLOCKING SESSIONS DOING (CROSS-INSTANCE)?

```
SELECT /*+LEADING(x w) USE_NL(h w)*/
       h.sql_id
,      h.sql_plan_hash_value
,      SUM(10) ash_secs
FROM   my_ash w
       LEFT OUTER JOIN my_ash h
       ON  h.snap_id = w.snap_id
       AND h.dbid = w.dbid
       AND h.instance_number = w.blocking_inst_id
       AND h.session_id = w.blocking_session
       AND h.session_serial# = w.blocking_sessio_serial#
       AND ((    w.blocking_inst_id = w.instance_number
             AND h.sample_id = w.sample_id
             AND h.sample_time = w.sample_time)
       OR  (     w.blocking_inst_id != w.instance_number
             AND h.sample_time >= w.sample_time-5/86400
             AND h.sample_time <  w.sample_time+5/86400))
WHERE w.event = 'enq: TX - row lock contention'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/
```

Each instance is responsible for it's own ASH samples.

Therefore, while the SNAP_IDs are consistent between different RAC instances, SAMPLE_IDs will usually be different.

Every 10th sample is persisted to AWR, therefore SAMPLE_TIME could be up to ±5 seconds apart across any 2 instances.

Just because the waiting session was blocked on it's instance when sampled, it doesn't mean the blocking instance was still blocking or active on its instance when sampled!

WHAT ARE THE BLOCKING SESSIONS DOING?

SQL_ID	SQL_PLAN_HASH_VALUE	ASH_SECS
		29210[1]
5st32un4a2y92	2494504609	10670[2]
652mx4tffq415	1888029394	7030
artqgxug4z0f1	8450529	580
7qxdrwcn4yzhh	3723363341	270

1. This SQL_ID is blank. May not be able to find ASH sample for blocking session because it is idle – busy on the client not the database.
2. This statement is running while the session holds a lock that is blocking another session.

AWR PLAN COST TRAP

EXECUTION PLAN CAPTURED BY AWR: CORRECT PLAN, OLD COSTS, OLD BINDS

DISPLAY_AWR()

```
SQL_ID 5st32un4a2y92
--------------------
SELECT 'X' FROM PS_CDM_LIST  WHERE CONTENTID = :1

Plan hash value: 2494504609

-----------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes   Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |       |            22 (100)|          |
|   1 |  INDEX FAST FULL SCAN| PS_CDM_LIST |     1 |     5      22  (10)| 00:00:01 |
-----------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1 / PS_CDM_LIST@SEL$1

Peeked Binds (identified by position):
--------------------------------------

   1 - :1 (NUMBER): 17776
```

FRESH EXECUTION PLAN GENERATED BY EXECUTE EXPLAIN PLAN

Note increase of cost of full scan.

```
Plan hash value: 2494504609
-----------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)  Time     |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |     1 |     6 |  3178   (9)  00:00:05 |
|*  1 |  INDEX FAST FULL SCAN| PS_CDM_LIST |     1 |     6 |  3178   (9)  00:00:05 |
-----------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("CONTENTID"=TO_NUMBER(:1))
```

EXECUTION PLAN CAPTURED BY AWR: CORRECT PLAN, OLD COSTS, OLD BINDS

The captured plan can is a valid guide to the path of execution

Do not rely on costs or number of rows as an indication of the amout of work done.

See also

- http://blog.go-faster.co.uk/2019/09/working-with-awr-old-statistics-in.html
- http://blog.go-faster.co.uk/2019/10/purging-sql-statements-and-execution.html

OTHER USES OF ASH

OTHER USES OF ASH

- Plan Stability
  - Did the plan change?
  - Adaptive Plans
- Sources of physical I/O
  - Hot tables/indexes
  - Which processes
  - Index Usage
- PGA Usage
- Temporary Tablespace Usage
- Effective parallelism

- Consolidated data
  - Export/Import into repository
  - Which database is active?
  - Which PDB is active?

QUESTIONS