

Active Session History (ASH): анализ нагрузки и решение проблем производительности БД Oracle

Михаил Базгутдинов
02 июня 2022 г.



30 лет

Agenda

- Как работает и из чего состоит Active Session History (ASH)
- Интерпретация ASH-отчетов и представлений ASH
- ASH-аналитика в Oracle Cloud Control
- Почему в ASH встречаются неверные данные? Как их обойти при анализе
- Примеры использования ASH для анализа проблем производительности
- Вопросы и ответы



Как работает и из чего состоит Active Session History (ASH)

В каких случаях DBA полезна ASH?

Заказчик желает узнать почему:

- Время отклика базы хуже обычного
- База данных «зависла»
- Пакетное задание не закончилось к сроку

Ответ нужен сейчас!

Нет, он не готов повторить проблему, когда вы включите трассировку.

Зачем добавили ASH в базу данных Oracle?

Средства диагностики производительности: **GV\$, AWR, SQL Monitoring.**

- **GV\$:** текущее состояние + счетчики, накопленные с момента старта.
 - Есть статистика по отдельным сессиям/транзакциям (с момента их старта, теряется без сохранения для истории при окончании сессии/транзакции).
 - Полезное исключение: `gv$sysmetric_history` – часть метрик, усреднение за 15/60 с.
- **AWR:** Полный набор статистик инстанса с усреднением за 10-60 минут.
- **Active Session History:** 1-секундное сэмплирование на уровне сессий.
- **SQL Monitor:** Подробные статистики по каждому выполнению запроса, включая шаги плана.
 - Некоторые метрики берет из ASH

Терминология

Active sessions - сессии БД, которые работают на CPU или находятся на non-idle ожиданиях.

Database time (db time) - суммарное время работы на CPU и non-idle-ожиданиях для всех сессий БД (т.е. активных сессий) . Не включая background процессы.

Database time / астрономическое время \approx Average active sessions

На какие вопросы можно найти ответы в ASH

- Глядя в **AWR/GV\$** (почти) нельзя сделать вывод о том:
 - Как DB time изменялось с течением времени?
 - Сколько времени пользователь ждал с момента начала выполнения SQL-запроса до его полного окончания
 - в GV\$SQL мы видим время, потраченное внутри БД
 - В каком месте (machine/module/action/sid) началась проблема?
 - Какая сессия была блокирующей?
 - У нас разные SQL с одинаковым планом (без bind-переменных).
 - Какая сессия «съела» львиную долю IO/PGA/TEMP/Interconnect?
 - Что делали сессии БД за минуту/несколько секунд до падения инстанса (уже после того, как был сделан последний AWR snap)?

Лицензии для использования ASH

ASH – часть Diagnostic pack для Oracle Enterprise Edition.

Вкл/выкл - `control_management_pack_access`

Не доступно для Standard Edition.

В SE в AWR заполняется только `dba_hist_sysmetric_history`.

Есть несколько "аналогов" ASH для SE.

Как база Oracle собирает ASH?

Фоновый процесс MMNL (MMON lightweighted) и MMON.

Каждую 1 сек – в память, каждые 10 секунд - на диск.

`GV$ACTIVE_SESSION_HISTORY` / `DBA_HIST_ACTIVE_SESS_HISTORY`

Значения для некоторых столбцов ASH в момент сэмпла могут быть неизвестны

Например: `SQL_PLAN_HASH_VALUE`, если парсинг еще не завершен.

Длющиеся события ожиданий.

События, длющиеся дольше 1 сек всегда попадут в ASH.

Одно событие может попасть в несколько сэмплов

По завершению ожидания время ожидания в колонке `TIME_WAITED` суммируется (fixed up) из предыдущих сэмплов и пишется в последнюю строку, в предыдущих обнуляется.

Последние строки ASH могут содержать незафиксированные значения в `TIME_WAITED`. (un-fixed).

Fix-up TIME_WAITED

```
select session_id,to_char(sample_time,'hh24:mi:ss') "Time", event,seq#,  
time_waited/1000/1000 "Waited, sec"  
from v$active_session_history where session_id=1409 -- and event='log file parallel write'  
and sample_time>sysdate-1
```

Output x Query Result x

SQL | Fetched 150 rows in 0,094 seconds

SESSION_ID	Time	EVENT	SEQ#	Waited, sec
1409	01:49:02	log file parallel write	44453	0.000484
1409	01:58:46	log file parallel write	46272	0.001449
1409	02:03:46	log file parallel write	47466	0.01032
1409	02:04:32	log file parallel write	47593	0.001909
1409	02:05:23	log file parallel write	47652	0.001294
1409	02:05:28	log file parallel write	47690	0.001325
1409	02:05:49	log file parallel write	48046	0.00045
1409	02:09:36	log file parallel write	49617	0
1409	02:09:37	log file parallel write	49617	2.211248
1409	02:14:12	log file parallel write	49930	0.000489
1409	02:17:31	log file parallel write	50105	0.00042
1409	02:19:13	log file parallel write	51047	0.000502
1409	02:20:50	log file parallel write	51179	0.001603
1409	02:24:02	log file parallel write	51478	0.000562
1409	02:28:06	log file parallel write	52214	0.000594
1409	02:30:23	log file parallel write	52566	0.000854



Событие с порядковым номером 49617 длилось более 2 сек. Попало в 2 сэмпла. 1-й – с нулем сек, последний – с актуальной длительностью.

ASH в памяти инстанса Oracle

- V\$ACTIVE_SESSION_HISTORY
- Циклический буфер памяти фиксированного размера
 - Размер не меняется => время хранения зависит от активности сессий
 - Oracle пытается подгадать размер буфера под 1 час активности базы.
 - Не больше 256 MB.
- Состояние всех «Активных» сессий собирается 1 раз в секунду
 - Быстрый прямой доступ к состоянию сессий в SGA
 - «Активная» сессия: либо на CPU, либо на non-idle событии ожидания
 - Сборщик ASH (процесс MMNL) не попадает в ASH.
 - и только лишь он
- Запросы к V\$ASH не защищены латчами => могут быть неконсистетными

Сколько ASH-данных есть в базе?

```
SELECT inst_id, oldest_sample_time, sysdate - oldest_sample_time ash_in_memory
FROM gv$ash_info;
```

INST_ID	OLDEST_SAMPLE_TIME	ASH_IN_MEMORY
1	24-JUL-17 10.00.01.886000000 AM	+000000001 05:32:14

```
select bytes/(1024*1024) MB from v$sgastat where name like
'ASH buffers';
```

MB

29

ASH на диске

- 1 из 10 СЭМПЛОВ из памяти пишется в SYSAUX
 - V\$ACTIVE_SESSION_HISTORY.IS_AWR_SNAPSHOT=Y
- DBA_HIST_ACTIVE_SESS_HISTORY
 - Секционирована по DBID, INSTANCE_NUMBER, SNAP_ID
 - Emergency flush under buffer pressure (ASH memory buffer is 2/3 full)
- AWR retention: по умолчанию 8 дней
 - Данных может накопиться много
- Для partition pruning желательно использовать фильтры по DBID, INSTANCE_NUMBER и диапазону SNAP_ID (см. DBA_HIST_SNAPSHOT)

SNAP_ID between 100 and 200 AND instance_number=1 AND DBID=(select dbid from v\$database)

Как настраивать сбор данных ASH?

_ash_enable	TRUE	Вкл/выкл сбор ASH.
_ash_size	1048618	Размер буфера в памяти (макс. 254 MB)
_ash_compression_enable	TRUE	To enable or disable string compression in ASH
_ash_disk_filter_ratio	10	Каждая N-я строка из памяти попадает на диск.
_ash_disk_write_enable	TRUE	Вкл/выкл хранение ASH на диске
_ash_eflush_trigger	66	The percentage above which if the in-memory ASH is full the emergency flusher will be triggered
_ash_min_mmnl_dump	90	Minimum Time interval passed to consider MMNL Dump
_ash_sample_all	FALSE	<p>Вкл/выкл запись в ASH всех сессий, не только активных</p> <p>Настройка 18с значение TRUE помогло избавиться от потерянных данных в ASH при более 500 активных сессий.</p> <p>Shallahamer с его помощью <u>доказывает</u>, что сессия висела на SQL*Net message from client.</p>
_ash_sampling_interval	1000	Time interval between two successive Active Session samples in millisecs

- Без острой необходимости эти параметры лучше не менять

ASH Pros and Cons

PROS

- Позволяет найти, из чего состоит DB Time в AWR-отчете за «плохой» период.
- Всегда доступно (с ограничением по лицензированию).
- Минимальная доп. нагрузка на сервер БД для сбора.

CONS

- Содержит гарантированно неполные данные, т.к. сэмплирует их.
- Запросы к v\$active_session_history могут быть весьма сложными
Для сравнения пригодны вероятности, а не абсолютные данные.
- 1-сек данные довольно быстро исчезают

Hint: create table ASH as SELECT * FROM V\$ACTIVE_SESSION_HISTORY;



Интерпретация ASH-отчетов и представлений ASH

Как построить ASH - отчет?

OEM -> Performance -> Performance Home -> "Run ASH Report".

Run ASH Report

Putty -> \$ORACLE_HOME/rdbms/admin/ashrpt.sql (ashrpti.sql for RAC)

SQL Developer -> DBA tab -> choose connection -> Performance -> ASH report Viewer


Some open-source and commercial 3rd party tools named like "ASH Viewer"


<https://sourceforge.net/projects/ashv/>

OEM. You can filter data for the report. (Filter by SQL_ID as an example).

Run ASH Report

Specify the time period for the report.

Start Date 
(Example: 12/15/03)

End Date 
(Example: 12/15/03)

Start Time ☐ AM ☒ PM

End Time ☐ AM ☒ PM

Filter

Other filters are: SID, Wait class, Service, Module, Action, Client.

Что читать в ASH-отчете? - Заголовок

- Data Source - V\$ or DBA_HIST
- Elapsed time - clock time
- Average Active Session - some kind of db time
- % of total db activity

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
MORPHEUS	4127582969	morpheus	1	11.2.0.4.0	NO	anna.testdomain.ru

CPU's	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
2	597M (100%)	396M (66.3%)	176M (29.5%)	4.0M (0.7%)

	Sample Time	Data Source
Analysis Begin Time:	28-Nov-17 12:09:53	V\$ACTIVE_SESSION_HISTORY
Analysis End Time:	28-Nov-17 12:18:53	V\$ACTIVE_SESSION_HISTORY
Elapsed Time:	9.0 (mins)	
Sample Count:	643	
Average Active Sessions:	1.19	
Avg. Active Session per CPU:	0.60	
Report Target:	SQL_ID like 'dskp9f0r23dj3'	98.2% of total database activity

Что читать в ASH-отчете? - Different "Top's"

As an example - Top sessions

Top Sessions

- '# Samples Active' shows the number of ASH samples in which the session was found waiting for that particular event. The
- 'XIDs' shows the number of distinct transaction IDs sampled in ASH when the session was waiting for that particular event
- For sessions running Parallel Queries, this section will NOT aggregate the PQ slave activity into the session issuing the PQ

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
156, 115	83.83	CPU + Wait for CPU	83.83	ASH	sqlplus@anna.t...u (TNS V1-V3)	539/540 [100%]	0
149, 771	9.02	CPU + Wait for CPU	9.02	ASH	sqlplus@anna.t...u (TNS V1-V3)	58/540 [11%]	0
36, 23	7.15	CPU + Wait for CPU	7.15	ASH	sqlplus@anna.t...u (TNS V1-V3)	46/540 [9%]	0

Что читать в ASH-отчете? - Wait events

Пример «Топ»-секции: Top waitevents.

Top User Events

Event	Event Class	% Event	Avg Active Sessions
row cache lock	Concurrency	54.55	352.53
library cache pin	Concurrency	31.38	202.80
library cache lock	Concurrency	9.48	61.27
enq: TX - index contention	Concurrency	3.88	25.07

- NB: ASH-отчет не пишет суммарное время ожидания или среднее время ожидания
- Смотрим вAWR:

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
library cache pin	663,203	1573.8K	2373	34.9	Concurrency
DB CPU		730.2K		16.2	
library cache lock	90,292	430.1K	4764	9.5	Concurrency
row cache lock	150,950	289.5K	1918	6.4	Concurrency
enq: SQ - contention	7,348	172.4K	23465	3.8	Configuration

Что читать в ASH-отчете? Параметры ожиданий

Top Event P1/P2/P3 Values

Event	% Event	P1 Value, P2 Value, P3 Value	% Activity	Parameter 1	Parameter 2	Parameter 3
row cache lock	54.55	"13","0","5"	54.54	cache id	mode	request
library cache pin	31.52	"5256317615496","5241691673728","3681830649856002"	0.02	handle address	pin address	100*mode+namespace
library cache lock	9.51	"5564666640808","5231897287152","3211267"	0.03	handle address	lock address	100*mode+namespace
enq: TX - index contention	3.88	"1415053316","197328909","1027724"	1.42	name mode	usn<<16 slot	sequence

% of this event in
the ASH report

% of these P1/P2/P3 set in
the ASH report

- Row cache lock – одинаковые параметры ("горячий" cache_id - 13) => v\$rowcache
- Library cache pin и library cache lock – разные параметры, т.е. нет единого «горячего» объекта.
- В AWR-отчете этих данных почти нет.
- Как из цифр в P1/P2/P3 найти название «горячего» объекта - зависит от события ожидания.

Параметры ожиданий => имена объектов БД.

- Как из цифр в P1/P2/P3 найти название «горячего» объекта?

Найти SQL-запрос в MOS-нотах или в блогах Oracle-энтузиастов.

NB: Для разных версий может быть разный запрос.

Пример: library cache: mutex X - сотни сессий одновременно выполняют триггер (массовый INSERT), каждая желает сделать PIN триггера в LC.

Решение:

```
select
    OWNER, name, LOCKED_TOTAL "Locked", PINNED_TOTAL "Pinned", FULL_HASH_VALUE
from v$db_object_cache
where HASH_VALUE={P1 value};
```

```
alter system set "_kgl_debug"="hash='c4982a06e69512c0348c4331433f51de' debug=33554432";
```

```
alter system set _kgl_hot_object_copies=128;
```

Длинный хеш для _kgl_debug - поле FULL_HASH_VALUE. 33554432 - магическое число.

Альтернативный способ: `exec dbms_shared_pool.markhot('&FULL_HASH_VALUE', o)`

Что читать в ASH-отчете? - Top SQLs'

- SQL commands related to top waitevents

Top SQL with Top Events

SQL ID	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% Rwsr	SQL Text
<u>3xcgx55t55juc</u>	379421932	1	64.66	CPU + Wait for CPU	40.00	TABLE ACCESS - FULL	40.00	select /*+ LEADING(A) USE_NL(B...
				direct path read	24.66	TABLE ACCESS - FULL	24.66	
<u>7wwa8agka0xg8</u>	2139464690	2	32.33	CPU + Wait for CPU	21.92	TABLE ACCESS - FULL	21.92	select /*+ LEADING(A) USE_NL(B...
				direct path read	10.41	TABLE ACCESS - FULL	10.41	

Count of distinct SQL
executions

Top SQL PLAN operation and it's
part in the total activity.

- ASH-отчет не содержит "elapsed time", "number of execs". Это в AWR.
 - По сырым ASH-данным тоже можно оценить.
- ASH-отчет показывает Top-операции плана выполнения, но не номер строки плана, хотя в V\$ASH номер строки плана есть: SQL_PLAN_LINE_ID.

Что читать в ASH-отчете? - Top Objects

- Top-objects (не для всех waitevents)

Top DB Objects

- With respect to Application, Cluster, User I/O and buffer busy waits only.

Object ID	% Activity	Event	% Event	Object Name (Type)	Tablespace
1641756	21.47	buffer busy waits	21.46	ODDO.SYS_LOB0000099610C00026\$\$SYS_LOB_P72695 (LOB PARTITION)	ODDO_TABLE
341299	1.02	enq: TX - row lock contention	1.02	ODDO.ODDO_WORTH_REMAINS.TB_38 (TABLE PARTITION)	ODDO_TABLE

- This information and even more can be found in AWR report too

gv\$active_session_history - How many columns?

- Oracle 11.2 - 96 columns
- Oracle 12.2 - 113 columns

v\$active_session_history - содержание колонок - 1

Name	Null?	Type
SAMPLE_ID		NUMBER
SAMPLE_TIME		TIMESTAMP(3)
IS_AWR_SAMPLE		VARCHAR2(1)
SESSION_ID		NUMBER
SESSION_SERIAL#		NUMBER
SESSION_TYPE		VARCHAR2(10)
FLAGS		NUMBER
USER_ID		NUMBER
SQL_ID		VARCHAR2(13)
IS_SQLID_CURRENT		VARCHAR2(1)
SQL_CHILD_NUMBER		NUMBER
SQL_OPCODE		NUMBER
SQL_OPNAME		VARCHAR2(64)
FORCE_MATCHING_SIGNATURE		NUMBER
TOP_LEVEL_SQL_ID		VARCHAR2(13)
TOP_LEVEL_SQL_OPCODE		NUMBER
SQL_PLAN_HASH_VALUE		NUMBER
SQL_PLAN_LINE_ID		NUMBER
SQL_PLAN_OPERATION		VARCHAR2(30)
SQL_PLAN_OPTIONS		VARCHAR2(30)
SQL_EXEC_ID		NUMBER
SQL_EXEC_START		DATE

ASH sample metadata

Session info

USER_ID -> Username
dba_users

SQL statement info

SQL_ID -> SQL Text
V\$sqlarea
dba_hist_sqltext

SQL execution plan info

Individual SQL execution info

v\$active_session_history - содержание колонок - 2

...	
PLSQL_ENTRY_OBJECT_ID	NUMBER
PLSQL_ENTRY_SUBPROGRAM_ID	NUMBER
PLSQL_OBJECT_ID	NUMBER
PLSQL_SUBPROGRAM_ID	NUMBER
QC_INSTANCE_ID	NUMBER
QC_SESSION_ID	NUMBER
QC_SESSION_SERIAL#	NUMBER
PX_FLAGS	NUMBER
EVENT	VARCHAR2(64)
EVENT_ID	NUMBER
EVENT#	NUMBER
SEQ#	NUMBER
P1TEXT	VARCHAR2(64)
P1	NUMBER
P2TEXT	VARCHAR2(64)
P2	NUMBER
P3TEXT	VARCHAR2(64)
P3	NUMBER
WAIT_CLASS	VARCHAR2(64)
WAIT_CLASS_ID	NUMBER
WAIT_TIME	NUMBER
SESSION_STATE	VARCHAR2(7)
TIME_WAITED	NUMBER

PL/SQL object info, join to
dba_procedures / @procid.sql

Parallel execution info

Wait event info

Wait event parameters
(extra info)

Remember, you should *not* sum
any wait columns, use
COUNT(*) to estimate DB Time

v\$active_session_history - содержание колонок - 3

BLOCKING_SESSION_STATUS	VARCHAR2(11)	Blocking session info
BLOCKING_SESSION	NUMBER	
BLOCKING_SESSION_SERIAL#	NUMBER	
BLOCKING_INST_ID	NUMBER	
BLOCKING_HANGCHAIN_INFO	VARCHAR2(1)	
CURRENT_OBJ#	NUMBER	DB object involved in a wait (not populated for all waits, not always cleaned up properly)
CURRENT_FILE#	NUMBER	
CURRENT_BLOCK#	NUMBER	
CURRENT_ROW#	NUMBER	
TOP_LEVEL_CALL#	NUMBER	Database call (OPI call) info
TOP_LEVEL_CALL_NAME	VARCHAR2(64)	
CONSUMER_GROUP_ID	NUMBER	Current transaction info
XID	RAW(8)	
REMOTE_INSTANCE#	NUMBER	
TIME_MODEL	NUMBER	Time model phase info. These Y / N flags tell in which phase (SQL parse, SQL execute, login, PL/SQL, login) the session happened to be when sampled
IN_CONNECTION_MGMT	VARCHAR2(1)	
IN_PARSE	VARCHAR2(1)	
IN_HARD_PARSE	VARCHAR2(1)	
IN_SQL_EXECUTION	VARCHAR2(1)	
IN_PLSQL_EXECUTION	VARCHAR2(1)	
IN_PLSQL_RPC	VARCHAR2(1)	
IN_PLSQL_COMPILATION	VARCHAR2(1)	
IN_JAVA_EXECUTION	VARCHAR2(1)	
IN_BIND, IN_CURSOR_CLOSE, IN_SEQUENCE_LOAD ...		

v\$active_session_history - содержание колонок - 4

CAPTURE_OVERHEAD	VARCHAR2(1)	DB Replay & workload capture
REPLAY_OVERHEAD	VARCHAR2(1)	
IS_CAPTURED	VARCHAR2(1)	
IS_REPLAYED	VARCHAR2(1)	
DBREPLAY_FILE_ID	NUMBER	Client application info
DBREPLAY_CALL_COUNTER	NUMBER	
SERVICE_HASH	NUMBER	
PROGRAM	VARCHAR2(48)	
MODULE	VARCHAR2(64)	
ACTION	VARCHAR2(64)	
CLIENT_ID	VARCHAR2(64)	Execution context identifier (end-to-end request ID)
MACHINE	VARCHAR2(64)	
PORT	NUMBER	More precise measurement of DB/CPU time between samples
ECID	VARCHAR2(64)	
TM_DELTA_TIME	NUMBER	
TM_DELTA_CPU_TIME	NUMBER	I/O counters. These can be summed over multiple samples
TM_DELTA_DB_TIME	NUMBER	
DELTA_TIME	NUMBER	Session memory usage when sampled (use MAX or AVG*)
DELTA_READ_IO_REQUESTS	NUMBER	
DELTA_WRITE_IO_REQUESTS	NUMBER	
DELTA_READ_IO_BYTES	NUMBER	
DELTA_WRITE_IO_BYTES	NUMBER	
DELTA_INTERCONNECT_IO_BYTES	NUMBER	
PGA_ALLOCATED	NUMBER	
TEMP_SPACE_ALLOCATED	NUMBER	

**TM_DELTA_TIME /
DELTA_TIME**
длительность между ASH-сэмплами одной и той же сессии. Например, если сессия не попала в ASH 13 сек, в этих колонках будет около 13 млн (микросекунд)

Основные колонки ASH

<code>SAMPLE_TIME</code>	Время сбора (сэмплирования)
<code>SESSION_STATE</code>	Ожидание или CPU (WAITING/ON CPU)
<code>SESSION_ID, SESSION_SERIAL#, SESSION_TYPE</code>	Атрибуты сессии
<code>SQL_ID</code>	Выполняемый SQL-запрос
<code>EVENT</code>	Событие ожидания (NULL для CPU)
<code>TIME_WAITED</code>	Длительность ожидания (o for CPU)

Waitevents

EVENT	Название события ожидания. NULL => ON CPU.
SEQ#	Порядковый номер события. У длинных событий ожидания одинаковое значение в нескольких строках. У коротких - большие пропуски в нумерации.
SESSION_STATE	WAITING или ON CPU
P1/P2/P3 , P1Text/P2Text/P3Text	Параметры события: значения и описание
TIME_WAITED	Длительность ожидания на момент. Fix-up механизм для событий дольше 1 сек. Oracle просит не делать SUM или AVG по этому полю.
WAIT_TIME	



SQL - related fields

SQL_ID

SQL_OPNAME

TOP_LEVEL_SQL_ID

FORCE_MATCHING_SIGNATURE

SQL_PLAN_HASH_VALUE

SQL_PLAN_LINE_ID

SQL_EXEC_ID, SQL_EXEC_START

PLSQL_OBJECT_ID/
PLSQL_SUBPROGRAM_ID

PLSQL_ENTRY_OBJECT_ID
PLSQL_ENTRY_SUBPROGRAM_ID

Текущий SQL (except for triggers and recursive server SQL)

Для текста SQL джойним с v\$sqlarea или dba_hist_sqltext

SELECT / INSERT / UPDATE / DELETE / PL-SQL / разные DDL

SQL_ID родительского запроса

Чтобы "ловить" SQL с литералами (не используют bind-переменные)

План выполнения. Тоже можно использовать для SQL с литералами.

Группировка по этому полю дает нам самую тяжелую операцию в плане выполнения. Также помогает определить горячий объект (обычно индекс) для некоторых ожиданий.

Позволяют отличить разные выполнения одного SQL_ID. SQL_EXEC_ID начинается не с 0, с с 16 млн.

Пакет и его процедура, которая сейчас выполняется. _ENTRY_ - пакет и процедура внизу стэка вызовов (процедуру одного пакета вызвали из другого).

Для имени объекта - джойним с dba_objects по полю object_id.



SQL - related fields - пример из практики

TOP_LEVEL_SQL_ID

SQL_ID родительского запроса

В AWR-отчете запрос, который выполнялся **47 млн** раз за 30 минут генерит много I/O.

Какая часть приложения выполняет запрос? SQL Module у него пустой.

Executions	Rows Processed	Rows per Exec	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
46,790,948	6,424,230	0.14	65,332.44	5.7	95.4	<u>2q1pc1y8uf0nr</u>		SELECT CASE WHEN :B2 ='CC' THE...
46,712,382	6	0.00	9,964.75	96.8	0	<u>6uhxvn601tfp6</u>		WITH CM_ALL AS(SELECT /*+ lea...
3,097,802	17,617,817	5.69	43,160.71	18.3	85.9	<u>1mw02zv6yszwu</u>	SBRF CC Contact FA Cards View	WITH MAIN_ASSET AS (SELECT /*+...

Ответ: в ASH видим, что у этого быстрого запроса везде одно и то же значение в TOP_LEVEL_SQL_ID.

В том же ASH видим, что этот Top-Level SQL выполняется уже 10 часов и ни разу не выполнялся. В SQL Monitoring'e - прикладная функция на шаге плана, выполняющемся несколько миллионов раз.

Вывод: немасштабируемый прикладной код.

Найти время запуска SQL-запроса

Анализируем пакетные задания

```
select  a.sql_exec_id,      a.sql_exec_start,
        a.module,          a.session_id,
        min(a.sample_time) "Started",
        max(a.sample_time) "Ended",
        Round((cast(max(a.sample_time) as date) - cast(min(a.sample_time) as date))*24*60*60) "Elapsed time, sec"
from    v$active_session_history a
where    sql_id = 'dskp9f0r23dj3'
group by a.sql_exec_id,      a.sql_exec_start,      a.module,      a.session_id
order by min(a.sample_time);
```

SQL_EXEC_ID	SQL_EXEC_START	MODULE	SESSION_ID	Started	Ended	Elapsed time, sec
16777219	28.11.2017 12:09:14	SQL*Plus	156	28.11.2017 12:09:15	28.11.2017 12:18:53	578
16777220	28.11.2017 12:09:22	SQL*Plus	36	28.11.2017 12:09:23	28.11.2017 12:09:25	2
16777221	28.11.2017 12:09:33	SQL*Plus	36	28.11.2017 12:09:34	28.11.2017 12:10:38	64
16777222	28.11.2017 12:09:45	SQL*Plus	149	28.11.2017 12:09:46	28.11.2017 12:10:50	64

Найти время запуска SQL-запроса

SQL Monitoring

Status	Duration	SQL Plan Hash	User	Parallel	Database Time	IO Requests	Start	Ended
	 3.5m	1534487942	ASH		 3.5m		12:09:14 PM	
	 1.1m	1534487942	ASH		 1.1m		12:09:45 PM	12:10:50 PM
	 1.1m	1534487942	ASH		 1.1m		12:09:33 PM	12:10:37 PM

На что было потрачено время выполнения SQL?

```
-- CPU and waits for every SQL execution
```

```
select
```

```
  a.sql_exec_start,
```

```
  NVL(a.event, 'ON CPU') "Wait event",
```

```
  COUNT(*) "ASH samples cnt",
```

```
  Round(SUM(TM_DELTA_TIME)/1e6) TM_DELTA_TIME,
```

```
  Round(SUM(DELTA_TIME)/1e6) DELTA_TIME,
```

```
  Round((cast(max(a.sample_time) as date) -
```

```
         cast(min(a.sample_time) as date))*24*60*60) "Elapsed time, sec"
```

```
from
```

```
  v$active_session_history a
```

```
where
```

```
  sql_id = 'dskp9f0r23dj3'
```

```
group by      a.sql_exec_start, NVL(a.event, 'ON CPU')
```

```
order by min(a.sample_time);
```

	SQL_EXEC_START	Wait event	ASH samples cnt	TM_DELTA_TIME	DELTA_TIME	Elapsed time, sec
1	12:09:14	ON CPU	578	578	580	578
2	12:09:22	ON CPU	3	3	3	2
3	12:09:33	ON CPU	65	73	73	64
4	12:09:45	ON CPU	65	65	65	64

Параллельное выполнение запросов и ASH

```
select
  a.sql_exec_id, NVL(a.QC_SESSION_ID, a.session_id) session_id,
  ROUND(PX_FLAGS/2097152) "Degree",
  a.sql_exec_start,
  NVL(a.event, 'ON CPU') "Wait event",
  COUNT(*) "ASH samples cnt",
  Round(SUM(TM_DELTA_CPU_TIME)/1e6) CPU_TIME,
  Round(SUM(TM_DELTA_DB_TIME)/1e6) DB_TIME
from
  v$active_session_history a
where
  sql_id in ('7wwa8agka0xg8', '3xcgx55t55juc')
group by
  NVL(a.QC_SESSION_ID, a.session_id), a.PX_FLAGS, a.sql_exec_id,
  a.sql_exec_start, NVL(a.event, 'ON CPU')
order by min(a.sample_time);
```

QC_INSTANCE_ID

QC_SESSION_ID

QC_SESSION_SERIAL#

PX_FLAGS

QC stands for "Query coordinator"

`Round((cast(max(a.sample_time) as date) - cast(min(a.sample_time) as date))*24*60*60) "Elapsed time, sec"`

Program column can also be an indicator of PX execution: **oracle@anna.testdomain.ru (P001)**

	SQL_EXEC_ID	SESSION_ID	Degree	SQL_EXEC_START	Wait event	ASH samples cnt	CPU_TIME	DB_TIME	Elapsed time, sec
1	16777216	35	4	14:23:38	ON CPU	4247	1272	4255	1626
2	16777216	35	4	14:23:38	direct path read	2233	674	2253	1625
3	16777216	36	(null)	14:23:53	ON CPU	1050	324	1062	1611
4	16777216	36	(null)	14:23:53	direct path read	555	167	550	1605
5	16777217	149	(null)	14:24:07	ON CPU	1035	296	1012	1598
6	16777217	149	(null)	14:24:07	direct path read	557	172	587	1594

ASH Top - scripts - Tanel Poder - example 1

Tanel Poder script: **ashtop.sql** - <https://github.com/tanelpoder/tpt-oracle/tree/master/ash>

Parameters: **GROUP BY, FILTER, FROM, TO**

Example1:

@ashtop.sql **event2** session_type='FOREGROUND' sysdate-10/1440 sysdate

Samples	AAS	%This	EVENT2	Distinct	
				FIRST_SE	LAST_SEE Execs Seen
709	2.4	65%	ON CPU	09:43:41	09:47:33 4
309	1.0	28%	direct path read	09:43:40	09:47:02 3
64	0.2	6%	direct path write temp	09:44:00	09:47:01 1
7	0.0	1%	direct path read temp	09:47:10	09:47:26 1

AAS - Average active sessions за период FROM and TO

Event2 - "улучшенная" колонка EVENT

ASH Top - scripts - Tanel Poder's - example 2

```
@ashtop.sql program2 "wait_class='User I/O'"
```

```
"to_timestamp(to_char(sysdate,'ddmmyyyy ')||'09:43:00','ddmmyyyy hh24:mi:ss')"
```

```
"to_timestamp(to_char(sysdate,'ddmmyyyy ')||'09:48:00','ddmmyyyy hh24:mi:ss')"
```

Samples	AAS	%This	PROGRAM2	First Seen	Last Seen	Distinct Execs Seen
311	1.0	82%	(Pnnn)	09:43:40	09:47:26	1
69	0.2	18%	(sqlplus)	09:44:00	09:45:35	2

Program2 - a "cleared" PROGRAM column

```
CASE WHEN a.session_type = 'BACKGROUND' OR REGEXP_LIKE(a.program, '.*\[PJ\]d+\') THEN
  REGEXP_REPLACE(SUBSTR(a.program, INSTR(a.program, '(')), '\d', 'n') -- меняем цифры на "n" в параллельных процессах и джобах
ELSE
  '('||REGEXP_REPLACE(REGEXP_REPLACE(a.program, '(.)@(.)\(.*)', '\1', '\d', 'n'))||')' -- убираем все, что после @
END || ' ' program2
```

Blocking session in ASH - fields

BLOCKING_SESSION/BLOCKING_SESSION_SERIAL#

SID/SERIAL# блокировщика

BLOCKING_INST_ID

Instance блокировщика (11g)

В документации описаны неправильно:

BLOCKING_SESSION: Populated only if the blocker is on the same instance and the session was waiting for enqueues or a "buffer busy" wait. Maps to V\$SESSION.BLOCKING_SESSION.

На самом деле для других событий тоже работает. И в RAC тоже.

SELF JOIN для дерева блокировок: **A.sample_id=B.sample_id** and **A.BLOCKING_SESSION=B.SESSION_ID** and **A.session_serial# = B.blocking_session_serial#**

Для RAC sample_time разные на разных инстансах: неочевидно как делать JOIN.

Значение в blocking_session может быть пустым, а может быть неверным.

Если блокировщик на Idle-ожидании, его не будет в ASH по определению.

Blocking session in ASH - scripts

Скрипт от Oracle (PL/SQL): `find_ASH_hang_chains.sql`

Скрипты Танела Подера (pure SQL, unlimited nesting):

<https://github.com/tanelpoder/tpt-oracle/tree/master/ash>
`ash_wait_chains.sql`, `dash_wait_chains.sql`

Простой скрипт (один уровень блокировок):

```
with waiters as
(
    select blocking_Session,blocking_session_Serial#,sample_id,count(*) cnt
    from gv$active_session_history
    where blocking_session is not null
    and event in ('library cache lock','cursor: pin S wait on X')
    group by inst_id,blocking_Session,blocking_session_Serial#,sample_id
)
select /*+ NO_MERGE(waiters) */ waiters.cnt "blocked sessions cnt", ash.*
from gv$active_session_history ash join waiters on ash.session_id = waiters.blocking_Session
    and ash.session_serial#=waiters.blocking_session_Serial# and ash.sample_id=waiters.sample_id
order by waiters.cnt desc,sample_time;
```

Пример работы скрипта: ash_wait_chains.sql

```

ash_samples s
, ash_data d
WHERE
  s.sample_id = d.sample_id
AND d.sample_time BETWEEN &3 AND &4
CONNECT BY NOCYCLE
  ( PRIOR d.blocking_session = d.session_id
    AND PRIOR d.blocking_inst_id = d.inst_id
    AND PRIOR s.sample_id = d.sample_id
  )
START WITH &2
)
SELECT * FROM (
  SELECT
    LPAD(ROUND(RATIO_TO_REPORT(COUNT(*)) OVER () * 100)||'%',5,' ') "%This"
  , COUNT(*) seconds
  , ROUND(COUNT(*) / ((CAST(&4 AS DATE) - CAST(&3 AS DATE)) * 86400), 1) AAS
  , COUNT(DISTINCT sids) distinct_sids
  , path wait_chain
  TO_CHAR(MIN(sample_time), 'YYYY-MM-DD HH24:MI:SS') first_seen

```

%This	SECONDS	AAS	WAIT_CHAIN	FIRST_SEEN	LAST_SEEN
50%	838840	77.7	-> dm7gh9ix7dzc.eng: TM - contention	2022-05-23 02:34:06	2022-05-23 03:38:05
48%	802310	74.3	-> bqz4usbvgrtpt.eng: TM - contention	2022-05-23 02:18:25	2022-05-23 03:38:05
2%	38580	3.6	-> bqz4usbvgrtpt.eng: TM - contention -> 6utq5s9v8acw6.library cache lock	2022-05-23 02:19:19	2022-05-23 02:34:01
0%	880	.1	-> 0qwwapf5grvkz.eng: TM - contention	2022-05-23 02:19:06	2022-05-23 02:33:58
0%	830	.1	-> 8vixzk2s4ytr.eng: TM - contention	2022-05-23 02:04:06	2022-05-23 02:18:07
0%	710	.1	-> 42td704gbu5at.eng: TX - row lock contention	2022-05-23 02:00:38	2022-05-23 03:30:32
0%	50	0	-> 6utq5s9v8acw6.eng: TM - contention	2022-05-23 02:18:17	2022-05-23 02:18:58
0%	20	0	-> 42td704gbu5at.eng: TX - row lock contention -> 6bfqc3kb2s0z3.db file sequential read	2022-05-23 02:30:44	2022-05-23 02:30:54
0%	20	0	-> 42td704gbu5at.eng: TX - row lock contention -> 6bfqc3kb2s0z3.ON CPU	2022-05-23 02:30:34	2022-05-23 02:31:04
0%	10	0	-> 2qr52n47vt2aj.eng: RO - fast object reuse -> [idle blocker 1,504,54343]	2022-05-23 02:09:01	2022-05-23 02:09:01
0%	10	0	-> .eng: RO - fast object reuse	2022-05-23 02:48:48	2022-05-23 02:48:48

Проблема Hard parse (литералы в тексте SQL)

Проблема: разработчики приложения используют динамический SQL вместо bind-переменных:

Литералы	Bind-переменные
SELECT WHERE CUSTOMER_ID=1001; SELECT WHERE CUSTOMER_ID=1002;	SELECT WHERE CUSTOMER_ID=:cus;
Hard parse при каждом выполнении. Топ-запросы не попадают в AWR.	

Диагностика с помощью ASH:

SELECT COUNT (DISTINCT SQL_ID), FORCE_MATCHING_SIGNATURE ...

SELECT COUNT (DISTINCT SQL_ID), SQL_PLAN_HASH_VALUE ...

FORCE_MATCHING_SIGNATURE: совпадает текст SQL без учета литералов

SQL_PLAN_HASH_VALUE: совпадает план выполнения (у разных запросов)

CURSOR_SHARING=FORCE (нежелательно)

Найти проблемный шаг плана выполнения

На выполнение какого шага плана выполнения база тратит время?

```
select count(*), SQL_PLAN_LINE_ID  
From v$active_session_history  
where sql_id='dskpgfor23dj3'  
Group by SQL_PLAN_LINE_ID  
order by count(*) desc;
```

```
@ashtop.sql SQL_PLAN_LINE_ID sql_id='dskpgfor23dj3' sysdate-10/1440 sysdate
```

Что еще можно узнать из ASH?

- Количество выполнений SQL-запроса за период времени (от 1 сек).
- Какие запросы вызвали нехватку TEMP (ORA-1652) или PGA (ORA-4036)
- Сколько дисковых операций (IOPS и MB/sec) делает база данных
- Задержки LGWR менее 500 ms (которые не попадают в трейс LGWR)
- В какой момент изменился план выполнения запроса
- Из какой PL/SQL-процедуры вызывается проблемный SQL-запрос

Чего нет в ASH?

- Достоверной информации о блокирующей сессии
- Текста SQL-запросов
- Call stacks
- Значений bind-переменных
- Статистики OS
- Количества транзакций в секунду, logons в сек и т.п.
 - см. v\$sysmetric_history (усреднение за 1 минуту)

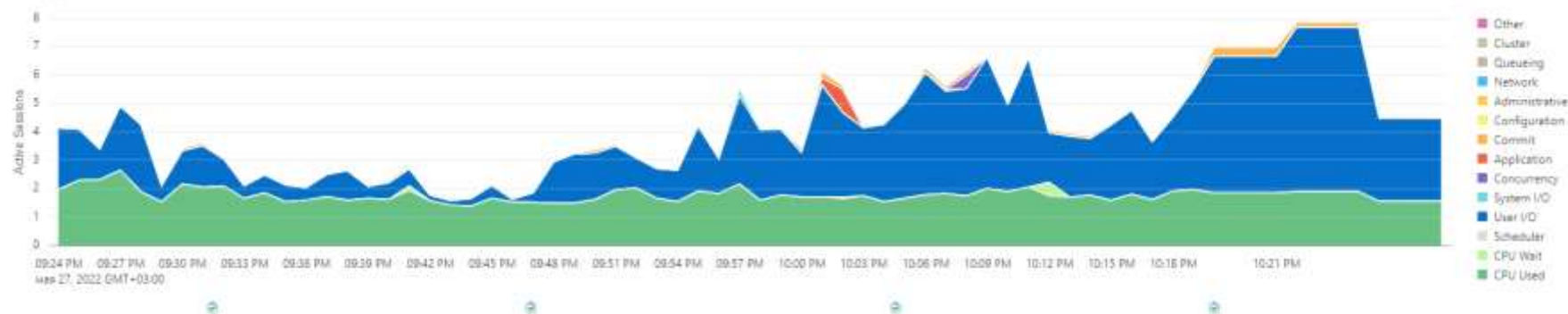


ASH-аналитика в Oracle Cloud Control

Performance -> Performance Home

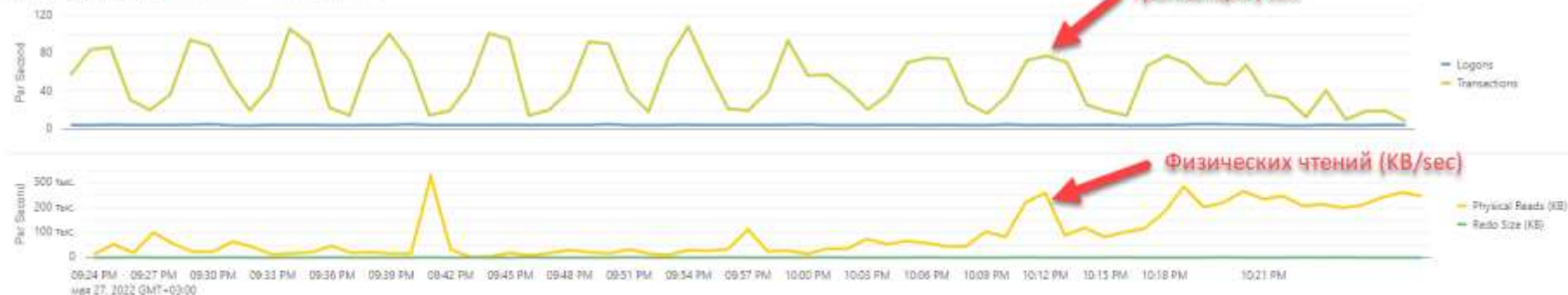
- Этот экран в основном из `v$sysmetric_history` (1 час), тут ASH с усреднением за 1 мин.
- Есть много других метрик: транзакции / сек, ввод-вывод и т.д.

Average Active Sessions

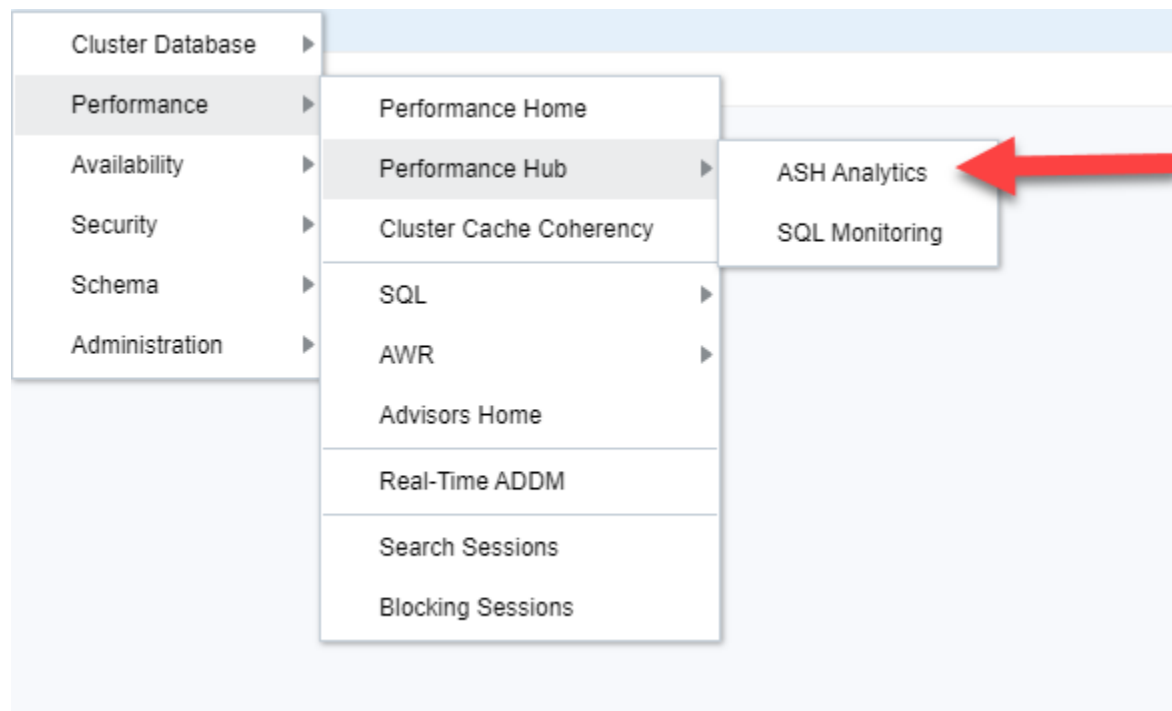


Throughput I/O Parallel Execution Services

Instance Throughput Rate ☒ Per Second ☐ Per Transaction



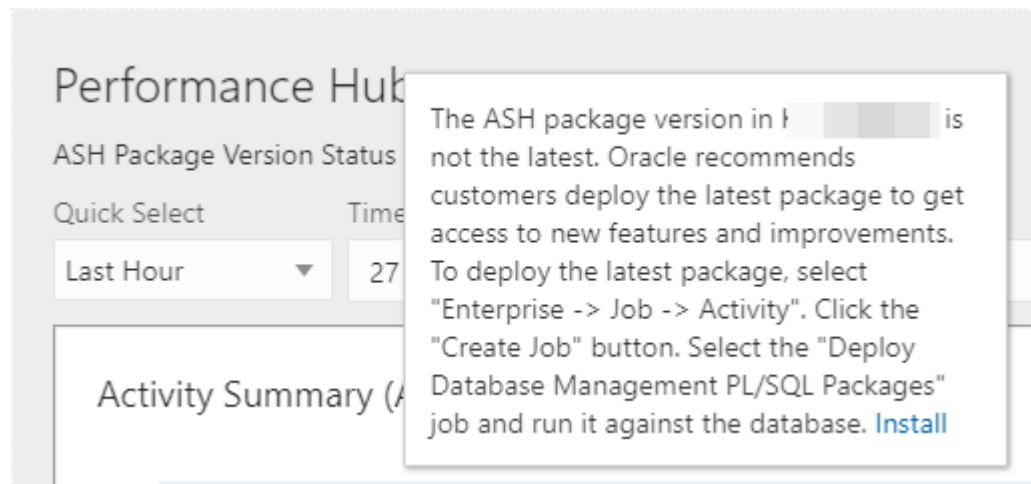
Performance -> Performance Hub -> ASH Analytics



- В старых версиях OEM - Top Activity

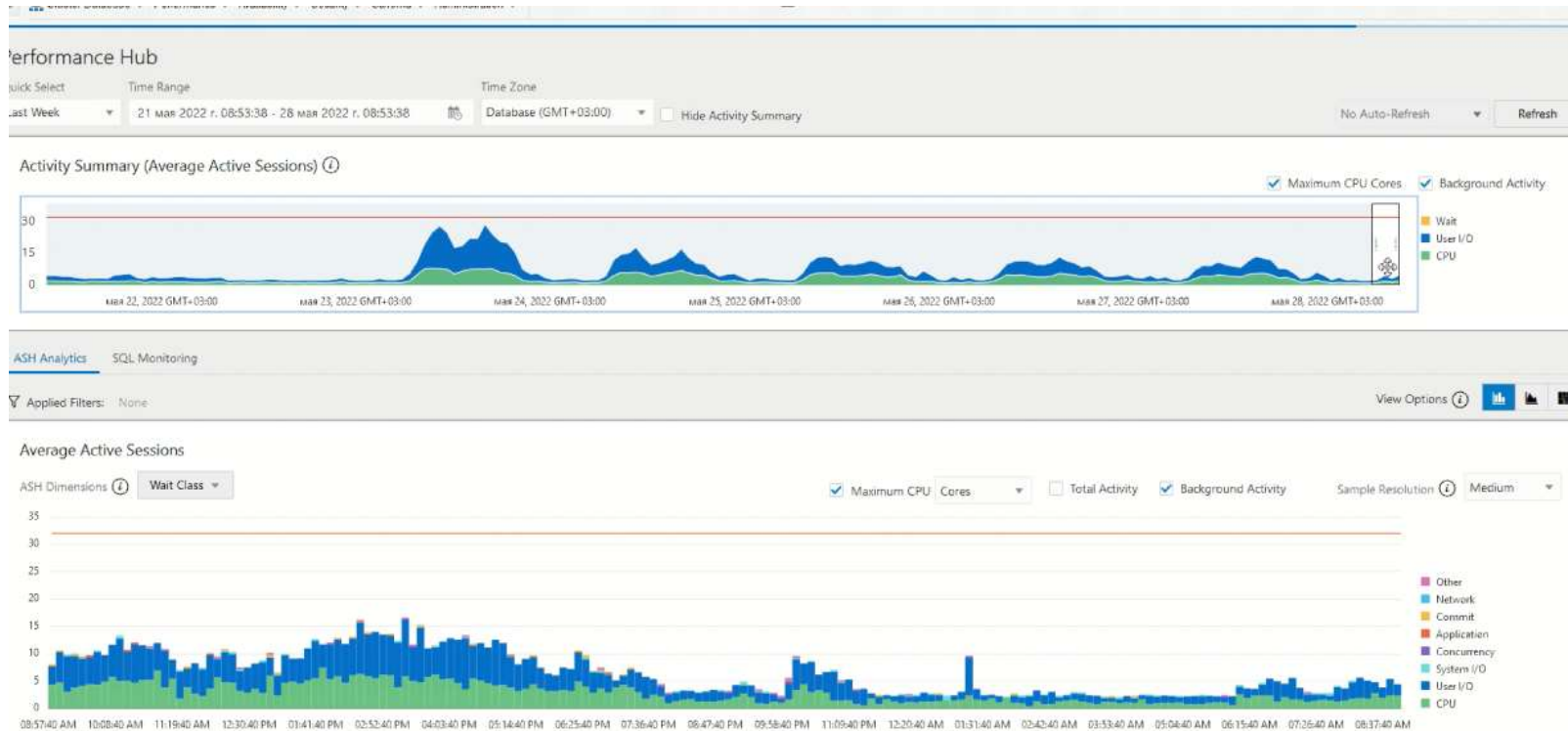
ASH Analytics: нужен пакет в схеме DBSNMP

- Если версия пакета неактуальная => предупреждение.
- Если пакета нет (11.2, 12.1) => требование установить.
 - Можно установить свежую версию на несколько баз джобом OEM.



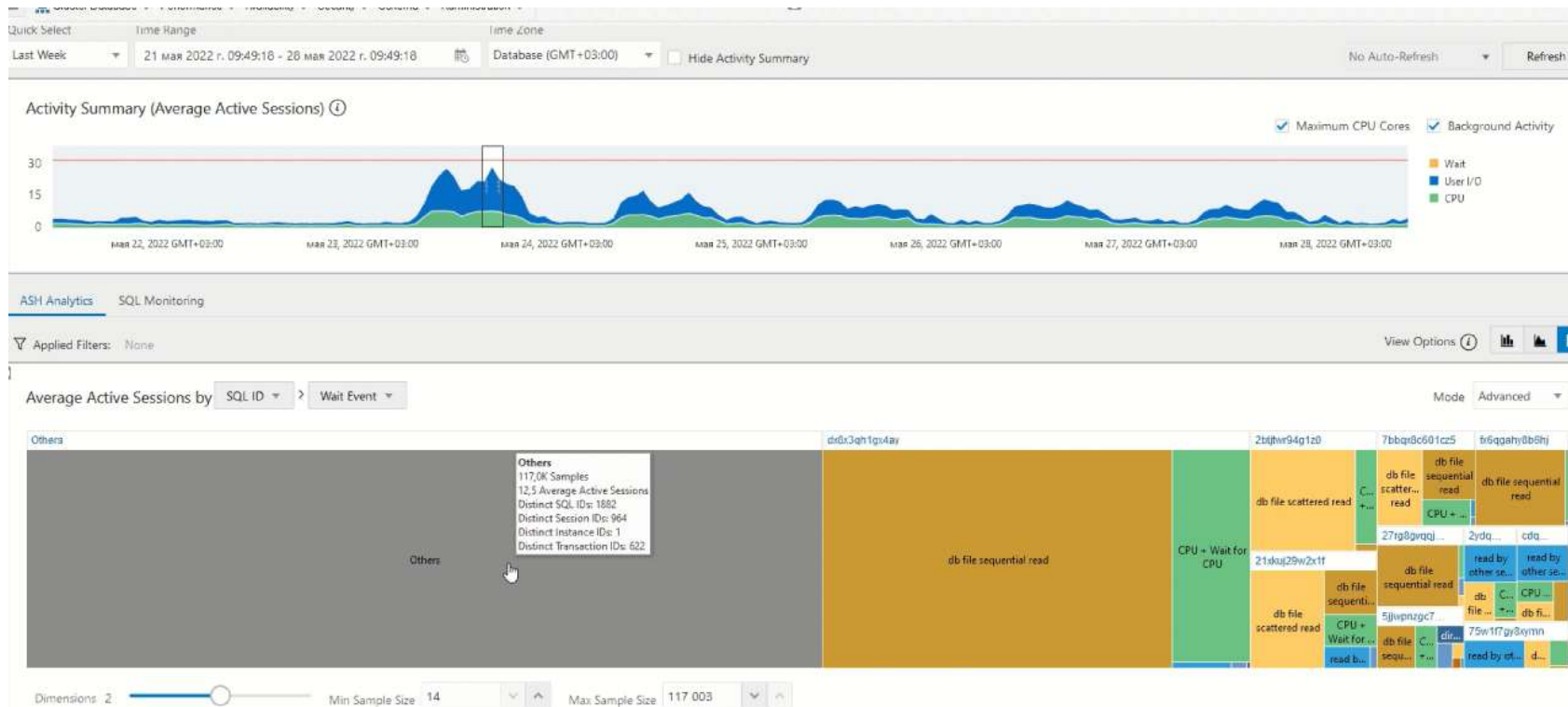
ASH Analytics -> Начало

■ видео-демонстрация



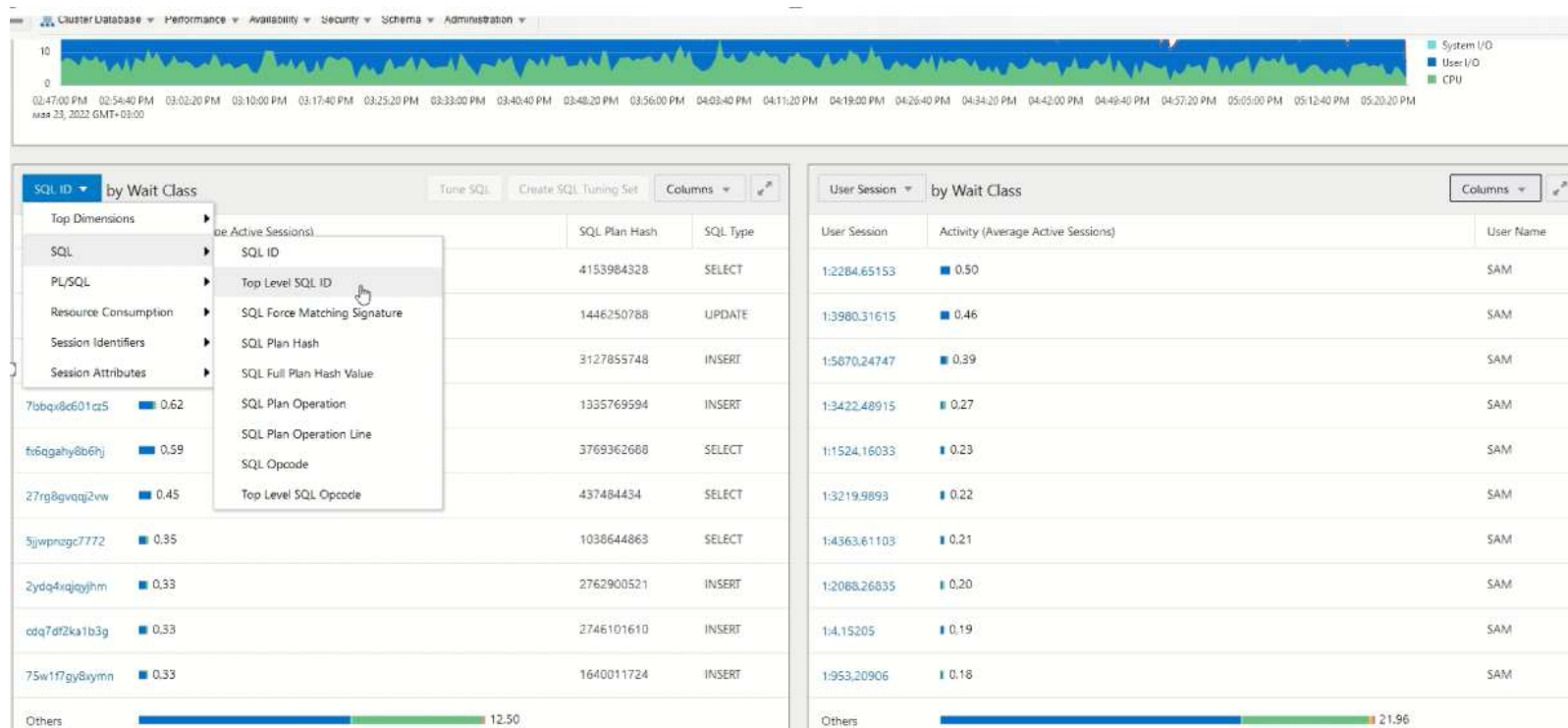
ASH Analytics -> Load Map

■ видео-демонстрация



ASH Analytics -> Top SQL

■ видео-демонстрация



ASH Analytics. Обзор функционала.

- Выбор интервала времени для анализа
- Клик по элементу в легенде добавляет фильтр по нему
- 2 вида отображения нагрузки: временная шкала или Load map.
- Два настраиваемых списка Топ-элементов в "подвале" страницы (Top SQL, Top Sessions и т.д.)
- Отдельная закладка "SQL Monitoring" для просмотра исторических отчетов SQL Monitoring'a

ASH Analytics. Ограничения.

- Фильтр нельзя задать вручную (например, для конкретного SQL_ID или с более сложной логикой).
- Если у вас есть сохраненный в таблице ASH, сюда его не загрузить.
- На графиках всегда COUNT(*) from ASH, хотя в ASH есть много других числовых колонок, пригодных для анализа (DELTA_READ_IO_REQUESTS и т.п).
- Нет анализа по Y/N-колонкам вроде "IN_PARSE".
- Баги есть. Желательно ставить свежие RU. В них также и новые возможности.
- В предыдущих версиях можно было сохранить страницу в файл в виде динамического HTML и отправить по почте. Сейчас нет такого функционала.
 - `@?/rdbms/admin/perfhubrpt.sql`

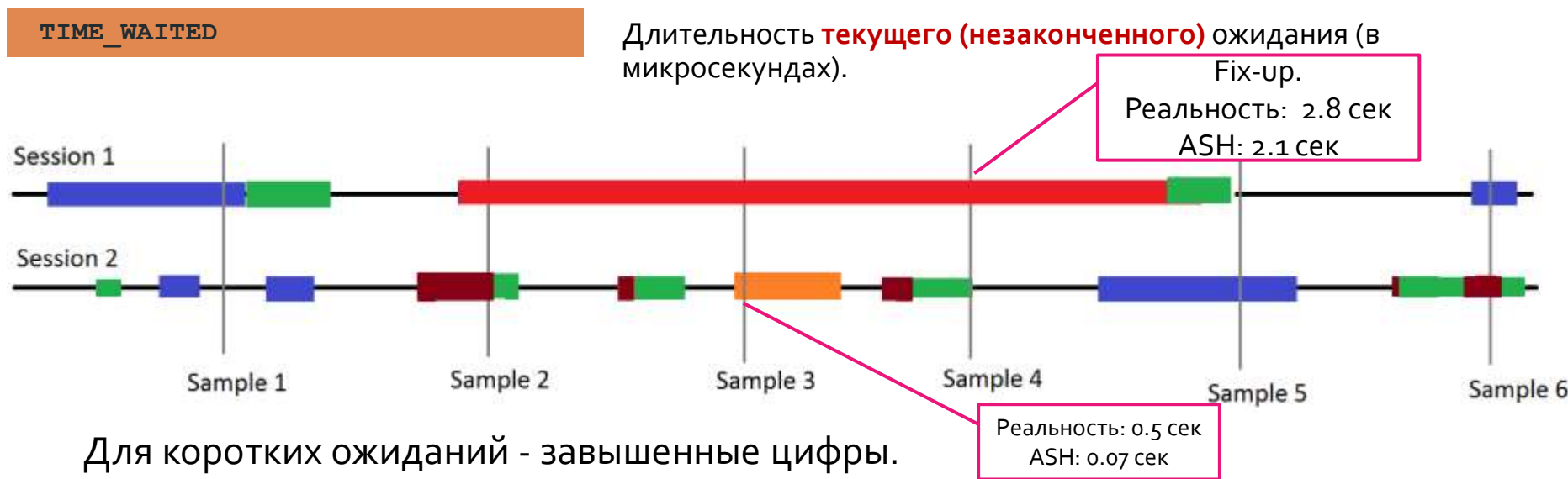


**Почему в ASH
встречаются неверные
данные? Как их обойти
при анализе**

Сэмплирование

- Концепция ASH - это сэмплирование, т.е. данные неполные by design
 - В ASH может попадать 1 из 10000 выполнений Топ-запроса
- При COUNT(*) > 20 данные приближаются к правде.

TIME_WAITED - заведомо искаженные данные



Для коротких ожиданий - завышенные цифры.

Для длинных - заниженные.

SUM(TIME_WAITED) или AVG(TIME_WAITED) - далеки от истины.

Более точные суммарное и среднее время ожидания - в AWR.

TIME_WAITED полезно сравнивать с "хорошим" периодом.

Неверные данные в колонках

- P1/P2/P3 - содержат бессмысленные значения при SESSION_STATE=ON CPU
- CURRENT_OBJ# - почти всегда неверное значение при ON CPU
- BLOCKING_SESSION - сомнительное значение при TIME_WAITED<1 сек

Блокирующая сессия в ASH - Правильная?

	Waiter	Blocker
SAMPLE_TIME	11.16.15.841 AM	11.16.15.841 AM
SESSION_ID	475	20683
SESSION_SERIAL#	10983	5821
SQL_OPNAME	INSERT	SELECT
SQL_PLAN_OPERATION	INSERT STATEMENT	TABLE ACCESS
SQL_PLAN_OPTIONS		BY LOCAL INDEX ROWID
SQL_EXEC_START		22.11.2017 11:16:11
EVENT	library cache: mutex X	db file sequential read
WAIT_CLASS	Concurrency	User I/O
SESSION_STATE	WAITING	WAITING
TIME_WAITED	10193	6361
BLOCKING_SESSION_STATUS	VALID	NO HOLDER
BLOCKING_SESSION	20683	
BLOCKING_SESSION_SERIAL#	5821	

Расследуем INSERT'ы, ожидающие на "library cache: mutex X".

Конкуренция либо за курсор INSERT SQL, либо за триггер (before_insert).

Мы определили корневую причину: частое выполнение INSERT в конкурентных сессиях.

Блокирующая сессия есть в ASH. Она 4 сек назад начала SELECT. В текущем сэмпле ждет на "db file sequential read".

Как этот SELECT, читающий таблицу, может блокировать наш INSERT на library cache?

Ответ: неверное значение в **BLOCKING_SESSION** для небольших TIME_WAITED.

Рекомендация: не верить на 100% значениям в отдельных строках ASH, только на COUNT(*)>n.

CURRENT_OBJ# / FILE# / BLOCK# / ROW#

CURRENT_OBJ#

Object ID объекта, с которым работает сессия.

Заполняется для **application (row lock, table lock)**, **cluster (on data blocks)**, **concurrency (buffer busy)**, and **user I/O on data blocks**.

Джойним с DBA_OBJECTS.DATA_OBJECT_ID.

CURRENT_FILE#

CURRENT_BLOCK#

CURRENT_ROW#

текущий файл БД, с которым работает сессия, блок в файле, строка в блоке.
V\$SESSION.ROW_WAIT_OBJ#.

Для CPU и других ожиданий эти колонки иногда содержат неверные данные.

Oracle не признает это багом.

Вместо NULL может быть 0 или -1.

CURRENT_OBJ# - when it is correct?


Customer case Ожидание enq: tx - index contention (Concurrency)

current_obj#: NULL или -1. Но он есть для блокирующей сессии, если сделать join v\$ASH с самой собой с текущим или предыдущим sample_id:

```
select decode(a2.current_obj#,-1,a1.current_obj#,null,a1.current_obj#,a2.current_obj#) current_obj#
      , a1.sample_time
      , count(*) cnt_samples
      , avg(decode(a1.time_waited,0,null,a1.time_waited)) avg_not_zero_time_waited
from v$active_session_history a1
left join v$active_session_history a2 on a1.blocking_session=a2.session_id and
a1.blocking_session_serial#=a2.session_serial# and decode(a1.time_waited,0,a1.sample_id-1,
a1.sample_id) =a2.sample_id
where a1.sample_time > sysdate-1/24/60*5 ----!!!! last 5 minutes
and a1.event = 'enq: TX - index contention'
group by decode(a2.current_obj#,-1,a1.current_obj#,null,a1.current_obj#,a2.current_obj#),
a1.sample_time
```

Выводы

- ASH позволяет проанализировать нагрузку базы, сессий, запросов с МНОЖЕСТВА точек зрения.
- Желательно иметь графический инструмент (Cloud Control).
- ASH содержит неполные, сэмплированные данные. Поэтому они верны лишь статистически.
 - при $COUNT(*) < 5$ велик шанс искажения ситуации
 - отдельные значения имеют право быть неверными.
- DBA_HIST_ACTIVE_SESS_HISTORY - сохраняется на диске каждые 10 сек, может пригодиться при расследовании сбоя инстанса.
- Дополнительная диагностика в AWR, SQL Monitoring, trace, errorstack.



129272, Москва,
Трифоновский тупик, 3

+7 (495) 747-7040

develop@fors.ru

www.fors.ru

