Oracle数据库的可观测性和问题诊断实践

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-21

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

* 活动最终解释权归甲骨文公司所有

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：
400-699-8888

Oracle 数据库的可观测性和问题诊断实践

甲骨文技术公益课 - 数据库专场

2023 年 8 月 11 日 11:00

线上直播

赵靖宇（Alfred Zhao）

Oracle数据库的可观测性和问题诊断实践

遇到问题时如何快速找到具体原因和制定解决方案

Oracle数据库的可观测性和问题诊断实践

从实际问题出发，了解如何在合适场景下正确选用某功能来定位问题

目标：

- **服务专家视角：**
  - 想查什么就能查什么，为问题诊断提供真实数据和可靠信息支撑
- **甲方客户视角：**
  - 初步判断问题点，对疑难杂症了解大概方向，更好掌控服务商处理结果

常用工具：

- Alert、AWR、ASH、Hanganalyze、SSD、Wait Event、10046、10053、SQL Monitor Report、SQL Profile、SPM、Hints...
- 很熟悉这些关键字？但实际... 一言难尽，总之，稍有迟疑就会延误处理时机
- 极简方式介绍这些工具的使用思路，并举例说明
- 直观理解并用好这些工具，从而享受由Oracle数据库的可观测性带来的一系列好处

Oracle数据库的可观测性和问题诊断实践

从实际问题出发，了解如何在合适场景下正确选用某功能来定位问题

1.数据库层的可观测性

- Alert、ADRCI、AWR 、ASH、Hanganalyze & systemstate dumps(SSD)

2.会话层的可观测性

- Wait Event 、10046

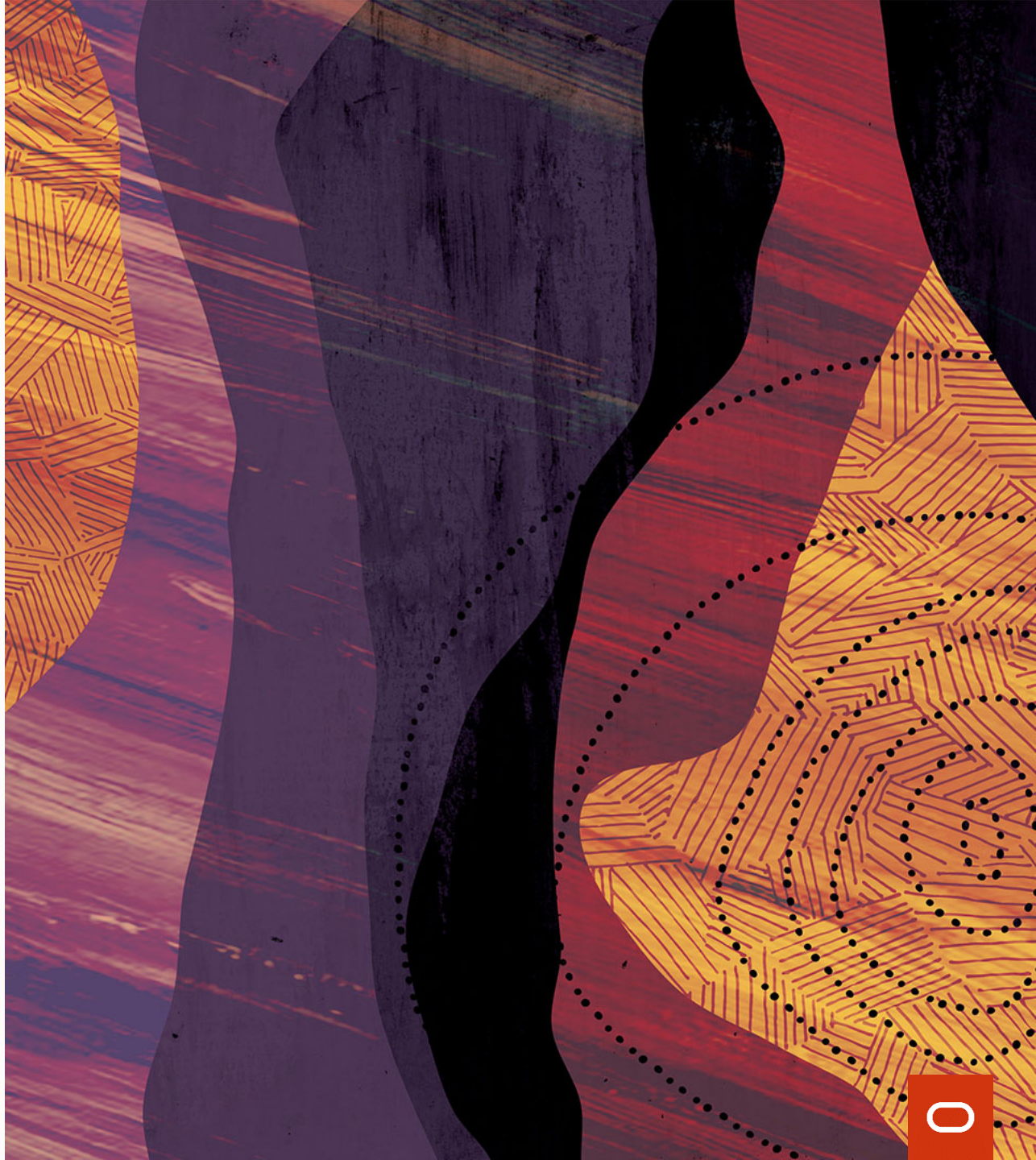
3.优化诊断实践

- DB & OS:
  - 数据库、系统层面配置：数据库各类参数，OS大页配置等符合设计/最佳实践
- SQL:
  - CBO， 统计信息，10053、SQL Monitor Report

4.应急场景处置

- SQL Profile，SPM，hints，快速诊断技巧?
- 总能有手段帮助你去解决问题，但是没其他办法时才用，不能依赖!

数据库层的可观测性



Alert日志

记录了数据库的运行情况、错误信息、警告以及其他重要事件

Alert日志非常重要！Tips：在你负责的环境，要设置一个别名，可以快速查到alert日志。

alias alert='tail -200f /u01/app/oracle/diag/rdbms/demo/demo/trace/alert_demo.log'

alert

```
[oracle@db01rac1 ~]$ alert
WARNING: A file of type FLASH BACK may exist in
db_recovery_file_dest that is not known to the database.
Use the RMAN command CATALOG RECOVERY AREA to re-catalog
any such files. If files cannot be cataloged, then manually
delete them using OS command. This is most likely the
result of a crash during file creation.
************************************************************
2023-08-01T14:18:59.179886+08:00
Errors in file /u01/app/oracle/diag/rdbms/demorac/jydb1/trace/jydb1_rvwr_6985.trc:
ORA-38701: Flashback database log 38 seq 26 thread 1: "+ARCHDG"
ORA-17502: ksfdcre:4 Failed to create file +ARCHDG
ORA-15041: diskgroup "ARCHDG" space exhausted
Warning: Oracle failed to create a flashback log even though
there appeared to be enough space for the log based on the
size of the fast recovery area.
Check if init.ora parameter DB_RECOVERY_FILE_DEST_SIZE has been
set to more than what the underlying storage system can support.
************************************************************
Unable to allocate flashback log of 6400 blocks from
current recovery area of size 16106127360 bytes.
Recovery Writer (RVWR) is stuck until more space
is available in the recovery area.
Unable to write Flashback database log data because the
recovery area is full, presence of a guaranteed
restore point and no reusable flashback logs.
```

ADRCI

使用自带工具ADRCI可以提高对数据库问题的快速响应和解决能力

如果不是自己负责的陌生环境，也可以使用adrci 快速查看对应日志：

adrci

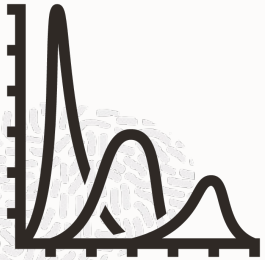
```
[oracle@db01rac1 ~]$ adrci

ADRCI: Release 19.0.0.0.0 - Production on Wed Aug 2 10:29:49 2023

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

ADR base = "/u01/app/oracle"
adrci> show alert

Choose the home from which to view the alert log:

1: diag/rdbms/demorac/jydb1
2: diag/clients/user_oracle/host_3923573244_110
3: diag/asmcmd/user_oracle/db01rac1
4: diag/kfod/db01rac1/kfod
Q: to quit

Please select option: 1
```

1

```
************************************************************
Errors in file /u01/app/oracle/diag/rdbms/demorac/jydb1/trace/jydb1_rvwr_6985.trc:
ORA-38701: Flashback database log 38 seq 26 thread 1: "+ARCHDG"
ORA-17502: ksfdcre:4 Failed to create file +ARCHDG
ORA-15041: diskgroup "ARCHDG" space exhausted
Warning: Oracle failed to create a flashback log even though
there appeared to be enough space for the log based on the
size of the fast recovery area.
Check if init.ora parameter DB_RECOVERY_FILE_DEST_SIZE has been
set to more than what the underlying storage system can support.
************************************************************
Unable to allocate flashback log of 6400 blocks from
current recovery area of size 16106127360 bytes.
Recovery Writer (RVWR) is stuck until more space
is available in the recovery area.
Unable to write Flashback database log data because the
recovery area is full, presence of a guaranteed
restore point and no reusable flashback logs.
```

AWR

快速定位是否为DB层问题



Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
[redacted]	Linux x86 64-bit	128	64	2	1007.06

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	1316	01-Aug-23 16:00:30	1212	5.2
End Snap:	1317	01-Aug-23 17:00:46	1211	5.2
Elapsed:		60.26 (mins)		
DB Time:		2,145.09 (mins)		

Report Summary

DB Time/s

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	35.6	0.2	0.00	0.00
DB CPU(s):	3.7	0.0	0.00	0.00
Background CPU(s):	0.2	0.0	0.00	0.00
Redo size (bytes):	4,866,181.1	28,704.6		
Logical read (blocks):	75,383.2	444.7		
Block changes:	15,371.1	90.7		
Physical read (blocks):	2,111.1	12.5		
Physical write (blocks):	0.0	0.0		
Read IO requests:	1,302.4	7.7		
Write IO requests:	0.0	0.0		
Read IO (MB):	16.5	0.1		
Write IO (MB):	0.0	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
Global Cache blocks received:	1,721.4	10.2		
Global Cache blocks served:	0.0	0.0		
User calls:	18,332.9	108.1		
Parses (SQL):	13,127.3	77.4		
Hard parses (SQL):	0.6	0.0		
SQL Work Area (MB):	2.0	0.0		
Logons:	0.1	0.0		
User logons:	0.0	0.0		
Executes (SQL):	17,575.6	103.7		
Rollbacks:	0.0	0.0		
Transactions:	169.5			

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
enq: RO - fast object reuse	4,992,811	46K	9.21ms	35.7	Application
reliable message	2,308,070	36.6K	15.86ms	28.5	Other
enq: TX - row lock contention	143,190	26.3K	183.74ms	20.4	Application
DB CPU		13.5K		10.5	
cell smart table scan	9,219,061	4061.8	440.59us	3.2	User I/O
enq: KO - fast object checkpoint	836,572	1383.8	1.65ms	1.1	Application
cell single block physical read: flash cache	2,308,137	553.2	239.68us	.4	User I/O
PGA memory operation	27,593,987	501.8	18.19us	.4	Other
gc current block 2-way	2,670,812	161.3	60.40us	.1	Cluster
gc current block 3-way	1,653,769	158.9	96.09us	.1	Cluster

Wait Classes by Total Wait Time

%DB Time

Wait Class	Waits	Total Wait Time (sec)	Avg Wait Time	% DB time	Avg Active Sessions
Application	12,340,315	74,184	6.01ms	57.6	20.5
Other	30,745,302	37,266	1.21ms	29.0	10.3
DB CPU		13,492		10.5	3.7
User I/O	11,868,815	4,705	396.39us	3.7	1.3
Cluster	6,913,234	635	91.79us	.5	0.2
Concurrency	888,586	89	100.18us	.1	0.0
Network	69,243,384	41	590.02ns	.0	0.0
Commit	336,153	24	72.49us	.0	0.0
Configuration	5,739	14	2.51ms	.0	0.0
Administrative	3	0	104.78ms	.0	0.0
System I/O	72	0	83.03us	.0	0.0

AWR

Top Events对应的SQL以及Top SQL

Top SQL with Top Events

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

Top SQL

SQL ID	Plan Hash	Executions	% Activity	Event	% Event	Top Row Source	% Row Source	SQL Text	Container Name
4xtx7wqhxx4r6	629372131	7935	62.95	enq: RO - fast object reuse	32.86	TABLE ACCESS - STORAGE FULL	32.8		
				reliable message	20.81	TABLE ACCESS - STORAGE FULL	20.81		
				CPU + Wait for CPU	6.07	TABLE ACCESS - STORAGE FULL	5.33		
4cvgsqynw0nxf	1066948210	2500	20.26	enq: TX - row lock contention	20.17	FOR UPDATE	20.1		
ann56h0mn7n5d	1845825214	762	6.03	reliable message	3.74	TABLE ACCESS - STORAGE FULL	3.74		
				enq: RO - fast object reuse	1.24	TABLE ACCESS - STORAGE FULL	1.24		
5ydybhyk5mw82	2105301041	674	5.35	reliable message	3.33	TABLE ACCESS - STORAGE FULL	3.33		

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 97.1% of Total DB Time (s): 128,705
- Captured PL/SQL account for 0.0% of Total DB Time (s): 128,705

Top SQL

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	PDB Name	SQL Text
80,539.13	1,691,285	0.05	62.58	7.54	3.97	4xtx7wqhxx4r6			
26,436.61	282,763	0.09	20.54	0.11	0.00	4cvgsqynw0nxf			
8,289.74	282,545	0.03	6.44	7.35	6.00	ann56h0mn7n5o			
7,278.71	327,481	0.02	5.66	8.48	7.27	5ydybhyk5mw82			
231.49	858,983	0.00	0.18	40.73	11.49	5n28a6abwcttv			
180.68	554,623	0.00	0.14	27.94	4.90	4pcjx3mqjk3zc			
144.08	1,224,579	0.00	0.11	30.14	0.04	8x5ubat3x87a2			
134.55	282,552	0.00	0.10	33.38	14.85	a8my1mxc7afq			
129.89	285,343	0.00	0.10	35.68	7.94	0rvg7ff601tgc			
125.52	327,239	0.00	0.10	42.97	2.60	8287pust9q88k			

ASH

ASH裸数据dba_hist_active_sess_history的分析

裸数据来源：

- v$active_session_history　　每1s采集一次
- dba_hist_active_sess_history　　每10s采集一次

ASH裸数据典型分析思路：

- 1.确定异常时刻的top n event
- 2.确定最终的top holder
- 3.找出每个采样点的最终top holder

其实很多基于这些数据写的SQL，并没有什么固定标准答案，无非是根据实际需求来对这些数据进行分析。

值得一提的是，因为这些数据记录了数据库的活动会话详细信息，因此可以对数据库运行状态进行细粒度的观测，所以很多监控软件也会使用这些信息进行分析和展示。

Hanganalyze & systemstate dumps

记录一则ASM实例阻塞，rbal进程异常的案例

- Hanganalyze

可以非常快速的诊断出会话间阻塞关系。

- Systemstate dump（SSD）

系统状态转储，是一种用于故障排除和问题分析的重要工具。

当数据库遇到严重的故障或性能问题时，可以生成Systemstate dump，它会捕获数据库实例的当前状态，包括会话信息、锁信息、等待事件、内存结构和其他关键信息。

Tips：对于一般的数据库管理和调优，通常不需要经常使用它们。但在出现疑难杂症时，它是一个非常有用的工具，可以帮助数据库管理员快速诊断和解决问题。

Hanganalyze & systemstate dumps

记录一则ASM实例阻塞，rbal进程异常的案例

故障现象：

数据库无法进行任何查询，alert告警ORA-00494：enqueue [CF] held for too long (more than 900 seconds) by ‘inst 1, osid 18875390’

登录到db实例，尝试查询select open_mode from v$database都会hang住。使用10046 event跟踪，发现最后的等待事件也是卡在'Disk file operations I/O'不再刷新。

该环境是standalone的单实例ASM环境，既然卡在I/O，自然就要去判断ASM是否正常。

发现ASM实例确实存在阻塞：其中417是rbal进程，等待事件是CSS operation：action。

SID	SQL_ID	EVENT	STATUS	ISLEAF	TREE	TREE_LEVEL
417		CSS operation: action	ACTIVE	1	<-705<-417	2
417		CSS operation: action	ACTIVE	1	<-1505<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-514<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-546<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-578<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-737<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-962<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-994<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1025<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1026<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1058<-705<-417	3

SID	SQL_ID	EVENT	STATUS	ISLEAF	TREE	TREE_LEVEL
417		CSS operation: action	ACTIVE	1	<-1249<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1281<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1313<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1409<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-1473<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-66<-705<-417	3
417		CSS operation: action	ACTIVE	1	<-98<-705<-417	3

18 rows selected.

Hanganalyze & systemstate dumps

记录一则ASM实例阻塞，rbal进程异常的案例

首先查找MOS时匹配到下面的文档： ASM Instance Hangs During The Diskgroup Mount Stage After AIX OS Patch Install (文档 ID 1633273.1)

For Standalone:
$> sqlplus /nolog
SQL> connect / as sysasm
SQL> oradebug setmypid
SQL> oradebug unlimit
REM : The next line should give something like Hang Analysis in $ORACLE_BASE/diag/.../trace/$ORACLE_SID_diag_<pid>.trc. Upload this
REM : Run the following two lines on one instance 2-3 times - 1 minute apart:
SQL> oradebug hanganalyze 3
SQL> oradebug dump systemstate 258
REM : The following line will print the location for the systemstate trace. Upload this
SQL> oradebug tracefile_name
REM : Also upload the instance alert log.

Hanganalyze & systemstate dumps

记录一则ASM实例阻塞，rbal进程异常的案例

The systemstate shows:

```
Resource                    Holder State
IPC 13                      13   Blocker                    <--------- RBAL
Enq DD-00000000-00000001    17   17: is waiting for IPC 13  <--------- ALTER DISKGROUP ALL MOUNT

Short stack dump:   <------------- RBAL
ksedsts ksdxfstk ksdxcb sspuser open64 ssOswOpenCio ss_osw_wopen ssOswOpen
skgfrdevstat skgfrdscdevs ksfddscdevs kfk_dscv_ufs_init kfk_discover_disks
kfdDiscoverString kfdDiscoverShallow kfgbDriver ksbabs ksbrdp opirip opidrv
sou2o opimai_real ssthrdmain main __start
...
    Session Wait History:
...
     0: waited for 'CSS operation: action'
...
     1: waited for 'GPnP Get Item'
...
     2: waited for 'CSS initialization'
...
     3: waited for 'rdbms ipc message'
```

根据收集到的trc文件和MOS描述的故障现象进行匹配，无论是ssd的等待事件历史，还是hanganalyze中显示的函数调用名称和顺序，结果都与MOS的描述一致。但是MOS描述的现象还明确提出是在安装了一个OS的patch后才出现的故障：

SYMPTOMS

non-clustered -- 11203 -- AIX 7.1

ASM instance hangs and will not mount diskgroups, after AIX OS patch was installed (AIX 7.1TL03-01-1341).
This is a platform specifc issue.

Hanganalyze & systemstate dumps

记录一则ASM实例阻塞，rbal进程异常的案例

- **确认原因之后，先解决当前问题：**

现在既然确认是ASM实例阻塞，自然就想到只需要将阻塞进程杀死或者干脆重启ASM实例甚至has集群即可暂时解决。

但实际上事违人愿，我在尝试杀死这个rbal进程时，发现即使使用kill -9也无济于事。并且即使将ASM实例成功abort后，这个rbal进程依然在，进一步尝试直接强制关闭crsctl stop has -f集群也无法成功。

看来目前的环境已经完全表现异常，最终还是重启了主机才恢复正常。

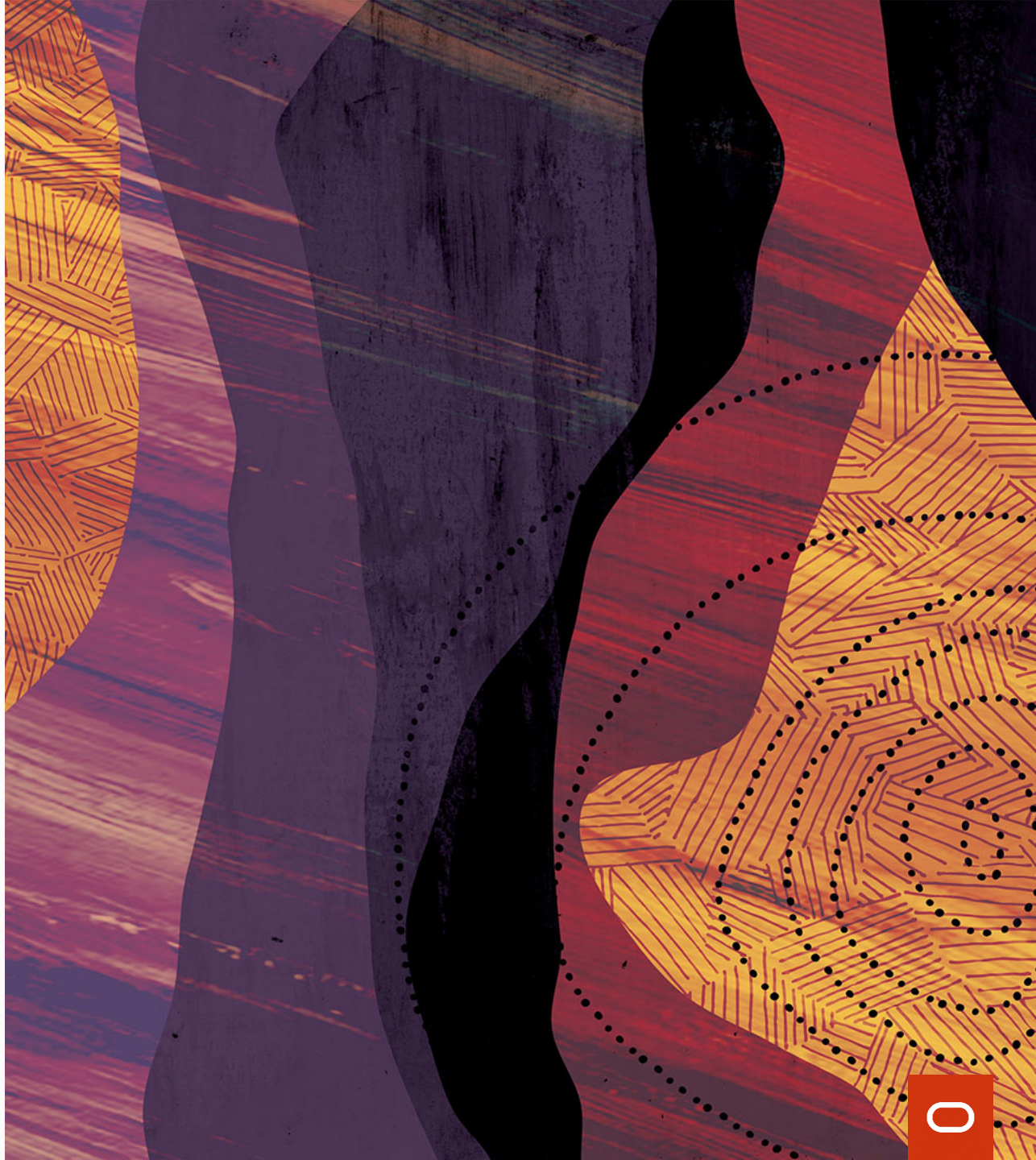
- **问题复盘：**

虽然最终用到了重启大法：

kill进程 –> abort –> crsctl stop has -f –> 重启主机…

但始终能做到心中有数，知道问题根因，也了解如何根本解决此问题，这就是可观测性带来的价值所在！

会话层的可观测性



Wait Event

每一个等待事件都是一个埋点

初级DBA：不好了！不好了！核心数据库卡住了！请专家快帮看看吧！

专家：**在等什么?**

初级DBA：啊？！等啥？等您赶紧帮分析处理啊！！！

Wait Event（等待事件）：等待事件是指在执行过程中数据库进程等待的特定操作或资源。通过监视等待事件，您可以确定系统中存在的瓶颈，了解数据库进程等待的原因，并针对性地进行性能优化。一些常见的等待事件包括I/O等待、锁等待和CPU等待。

```
SQL> select count(*) from v$event_name;

  COUNT(*)
----------
      1920
```

1920个具体等待事件...

```
SQL> select distinct WAIT_CLASS from v$event_name order by 1;

WAIT_CLASS
----------------------------------------------------------------
Administrative
Application
Cluster
Commit
Concurrency
Configuration
Idle
Network
Other
Queueing
Scheduler
System I/O
User I/O

13 rows selected.
```

13类等待事件...

10046 底层递归调用一清二楚

EXP导出aud$报错EXP-00008,ORA-00904 解决

10046是Oracle数据库的事件追踪功能，有些操作涉及递归操作，只有表面的报错并不能解决问题。
举例说明：exp导出报错ORA-00904现象：

```
[oracle@jyrac1 ~]$ exp system/oracle file=audit.dmp log=audit.log tables=sys.aud$

Export: Release 11.2.0.4.0 - Production on Wed Jan 17 17:16:30 2018

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Tes
Export done in ZHS16GBK character set and AL16UTF16 NCHAR character set

About to export specified tables vi
Current user changed to SYS
. . exporting table                            AUD$
EXP-00008: ORACLE error 904 encountered
ORA-00904: : invalid identifier
Export terminated successfully with warnings.
```

哪里无效？咋排查？

10046 底层递归调用一清二楚

EXP导出aud$报错EXP-00008,ORA-00904 解决

定位exp进程，开启10046，再次执行exp操作：

```
SQL> oradebug setospid 15552
SQL> oradebug unlimit
SQL> oradebug tracefile_name
SQL> oradebug Event 10046 trace name context forever, level 12
```

我搜索904这个错误，可以匹配到err=904的部分，为了更好观察，我使用tkprof美化生成的trc文件：

```
********************************************************************************

The following statement encountered a error during parse:

SELECT SYNNAM, DBMS_JAVA.LONGNAME(SYNNAM), DBMS_JAVA.LONGNAME(SYNTAB),            TABOWN, TABNODE, PUBLIC$, SYNOWN, SYNOWNID,
TABOWNID, SYNOBJNO      FROM   SYS.EXU9PTS      WHERE  SYNOBJNO IN (          SELECT SYNOBJNO          FROM SYS.EXU9TYPT
WHERE TABOBJNO = :1 )      ORDER  BY SYNTIME

Error encountered: ORA-00904
********************************************************************************
```

10046 底层递归调用一清二楚

EXP导出aud$报错EXP-00008,ORA-00904 解决

根据报错和trc文件的内容，匹配到MOS文档：Errors EXP-8 ORA-904 During Export (文档 ID 1091927.1)

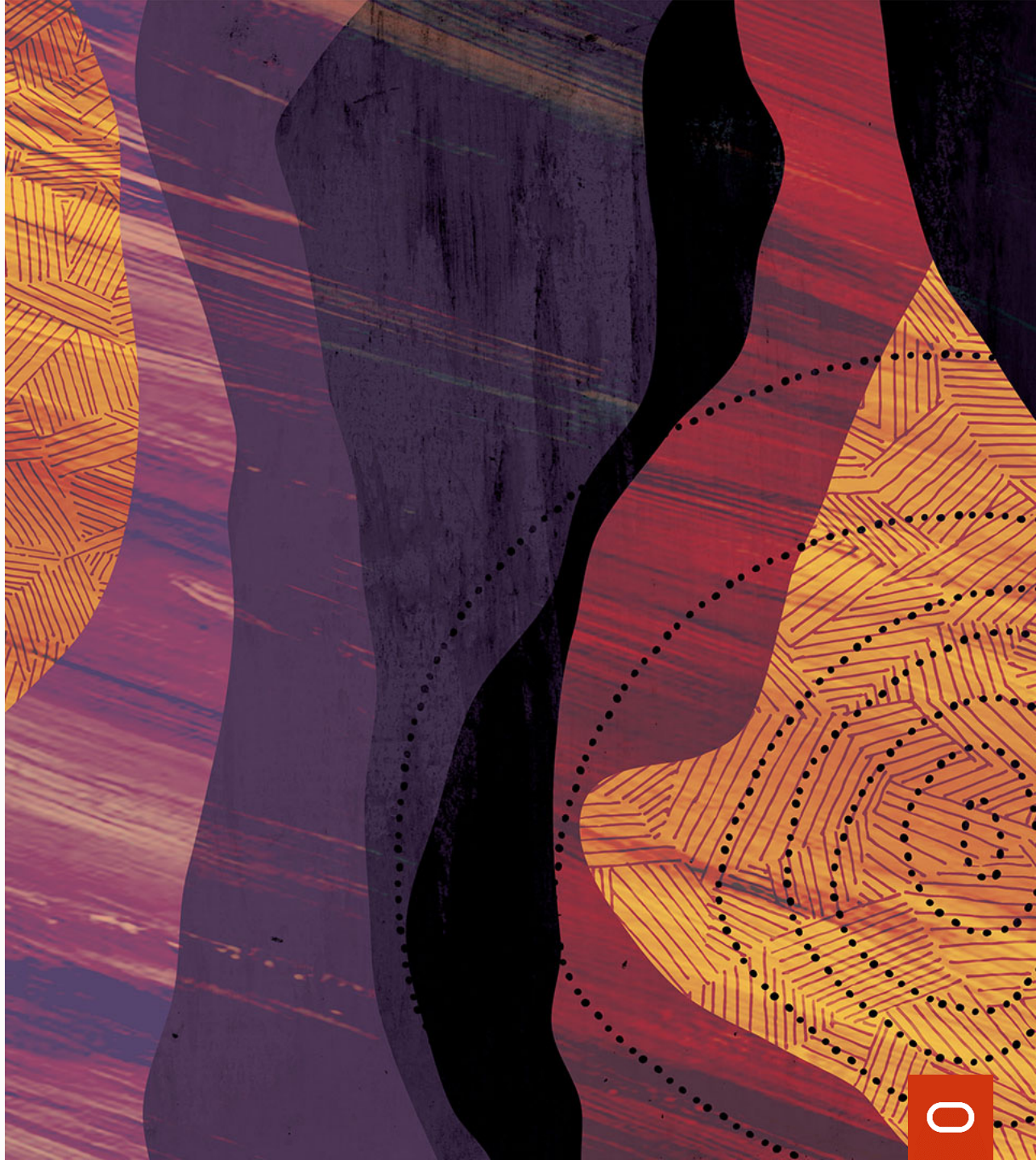
MOS解决方案赋权DBMS_JAVA给DBA角色：

```
SYS@jyzhao1 >grant execute on DBMS_JAVA to dba;

Grant succeeded.
```

再次尝试exp导出成功。

```
[oracle@jyrac1 ~]$ exp system/oracle file=audit.dmp log=audit.log tables=sys.aud$

Export: Release 11.2.0.4.0 - Production on Wed Jan 17 17:39:07 2018

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Tes
Export done in ZHS16GBK character set and AL16UTF16 NCHAR character set

About to export specified tables via Conventional Path ...
Current user changed to SYS
. . exporting table                           AUD$         25 rows exported
Export terminated successfully without warnings.
```

优化诊断实践



数据库、系统层面配置

数据库各类参数，OS大页配置等符合设计/最佳实践

为何在可观测性的主题上，单独强调这点呢？好像不太相关？

- 就是因为通常我们并不会去为这些基础项的配置问题而做过多支持。
- 这类基础问题也通常不会在我们讨论范围之内，所以先抛出来让大家尽量避免此类问题。

系统层面配置：

- 底层存储RAID配置、空间规划分配、系统内存大页配置…

数据库参数：

- SGA/PGA、Redo大小和组数、会话数、数据文件数、游标、隐藏参数（慎用，在售后专家指导下才可以选择使用）…
- 至少要符合你自己的稳定基线，不要出现低级配置不合理这类错误导致性能/故障问题。

10053 优化器为何如此选择?

10053是Oracle数据库的CBO（Cost-Based Optimizer）优化器的跟踪功能

启用10053跟踪可以生成SQL查询的优化器执行计划详细信息，包括代价估算和选择的访问路径。
这对于了解优化器是如何生成执行计划的，并调整查询以改进性能非常有帮助。

--SQL1:

```
SELECT * FROM t WHERE OBJECT_ID > 2;
```

--SQL2:

```
SELECT * FROM t WHERE OBJECT_ID > 3;
```

--10053使用方法：

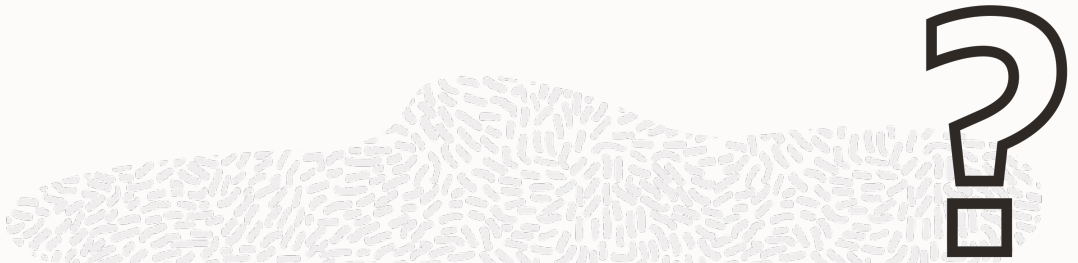
```
ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';
```

Your SQL!

```
ALTER SESSION SET EVENTS '10053 trace name context off';
```

10053 优化器为何如此选择?

两条SQL语句的执行计划差异?



```
SQL> SELECT * FROM t WHERE OBJECT_ID > 2;

76594 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1601196873

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      | 76594 |  9723K|   401   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| T    | 76594 |  9723K|   401   (1)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("OBJECT_ID">2)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       6474  consistent gets
          0  physical reads
          0  redo size
    4713503  bytes sent via SQL*Net to client
      56563  bytes received via SQL*Net from client
       5108  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      76594  rows processed
```

```
SQL> SELECT * FROM t WHERE OBJECT_ID > 3;

no rows selected


Execution Plan
----------------------------------------------------------
Plan hash value: 120220905

--------------------------------------------------------------------------------------------
| Id  | Operation                           | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |       |     1 |   130 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| T     |     1 |   130 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | IDX_T |     1 |       |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OBJECT_ID">3)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          2  consistent gets
          0  physical reads
          0  redo size
       2375  bytes sent via SQL*Net to client
        386  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

10053 优化器为何如此选择?

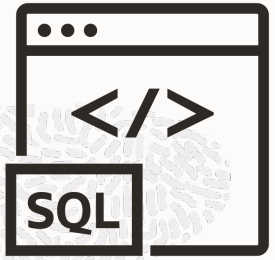
```
Access path analysis for T
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for T[T]
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = TABLE

 kkecdn: Single Table Predicate:"T"."OBJECT_ID">2
  Column (#4):
    NewDensity:0.000007, OldDensity:0.000007 BktCnt:76595.000000, PopBktCnt:76594.000000, PopValCnt:1, NDV:2
  Column (#4): OBJECT_ID(NUMBER)
    AvgLen: 3 NDV: 2 Nulls: 0 Density: 0.000007 Min: 2.000000 Max: 3.000000
    Histogram: Freq  #Bkts: 2  UncompBkts: 76595  EndPtVals: 2  ActualVal: yes
  Estimated selectivity: 0.999987 , endpoint value predicate, col: #4
  Table: T  Alias: T
    Card: Original: 76595.000000  Rounded: 76594  Computed: 76594.000000  Non Adjusted: 76594.000000
  Scan IO  Cost (Disk) =   399.000000
  Scan CPU Cost (Disk) =   60219219.600000
  Cost of predicates:
    io = NOCOST, cpu = 50.000000, sel = 0.999987 flag = 2048  ("T"."OBJECT_ID">2)
  Total Scan IO  Cost  =   399.000000 (scan (Disk))
                       + 0.000000 (io filter eval) (= 0.000000 (per row) * 76595.000000 (#rows))
                       =   399.000000
  Total Scan CPU  Cost =   60219219.600000 (scan (Disk))
                       + 3829750.000000 (cpu filter eval) (= 50.000000 (per row) * 76595.000000 (#rows))
                       =   64048969.600000
  Access Path: TableScan
    Cost:  400.806698  Resp: 400.806698  Degree: 0
      Cost_io: 399.000000  Cost_cpu: 64048970
      Resp_io: 399.000000  Resp_cpu: 64048970
 ****** Costing Index IDX_T
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_SCAN
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_FILTER
  Estimated selectivity: 0.999987 , endpoint value predicate, col: #4
  Access Path: index (RangeScan)
    Index: IDX_T
    resc_io: 1616.000000  resc_cpu: 76613147
    ix_sel: 0.999987  ix_sel_with_filters: 0.999987
    Cost: 1618.161109  Resp: 1618.161109  Degree: 1
  Best:: AccessPath: TableScan
         Cost: 400.806698  Degree: 1  Resp: 400.806698  Card: 76594.000000  Bytes: 0.000000

***************************************
```

10053 优化器为何如此选择?

```
Access path analysis for T
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for T[T]
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = TABLE

 kkecdn: Single Table Predicate:"T"."OBJECT_ID">3
  Column (#4):
    NewDensity:0.000007, OldDensity:0.000007 BktCnt:76595.000000, PopBktCnt:76594.000000, PopValCnt:1, NDV:2
  Column (#4): OBJECT_ID(NUMBER)
    AvgLen: 3 NDV: 2 Nulls: 0 Density: 0.000007 Min: 2.000000 Max: 3.000000
    Histogram: Freq  #Bkts: 2  UncompBkts: 76595  EndPtVals: 2  ActualVal: yes
  Using density: 6.5278e-06 of col #4 as selectivity of pred having unreasonably low value
  Table: T  Alias: T
    Card: Original: 76595.000000  Rounded: 1  Computed: 0.500000  Non Adjusted: 0.500000
  Scan IO  Cost (Disk) =   399.000000
  Scan CPU Cost (Disk) =   26518299.600000
  Cost of predicates:
    io = NOCOST, cpu = 50.000000, sel = 0.000007 flag = 2048  ("T"."OBJECT_ID">3)
  Total Scan IO  Cost  =   399.000000 (scan (Disk))
                       + 0.000000 (io filter eval) (= 0.000000 (per row) * 76595.000000 (#rows))
                       =   399.000000
  Total Scan CPU  Cost =   26518299.600000 (scan (Disk))
                       + 3829750.000000 (cpu filter eval) (= 50.000000 (per row) * 76595.000000 (#rows))
                       =   30348049.600000
  Access Path: TableScan
    Cost:  399.856060  Resp: 399.856060  Degree: 0
      Cost_io: 399.000000  Cost_cpu: 30348050
      Resp_io: 399.000000  Resp_cpu: 30348050
 ****** Costing Index IDX_T
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_SCAN
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = INDEX_FILTER
  Using density: 6.5278e-06 of col #4 as selectivity of pred having unreasonably low value
  Access Path: index (RangeScan)
    Index: IDX_T
    resc_io: 3.000000  resc_cpu: 22214
    ix_sel: 6.5278e-06  ix_sel_with_filters: 6.5278e-06
    Cost: 3.000627  Resp: 3.000627  Degree: 1
  Best:: AccessPath: IndexRange
  Index: IDX_T
         Cost: 3.000627  Degree: 1  Resp: 3.000627  Card: 0.500000  Bytes: 0.000000

***************************************
```


SQL Monitor Report

让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目



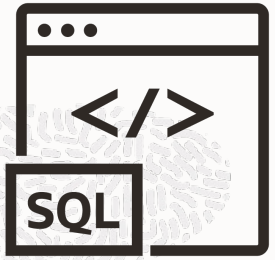
SQL Monitor Report（SQL监视报告）：SQL监视报告提供了关于特定SQL语句执行的详细信息，包括执行计划、统计信息、等待事件和资源使用情况等。通过分析SQL监视报告，您可以识别慢查询、高消耗或性能问题，并采取相应的措施进行优化。

强烈推荐大家使用SQL Monitor Report，且要使用ACTIVE的类型，这可以让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目。

比如我想评估In-Memory使用与否的效果。

SQL Monitor Report

让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目



- 1.准备通用的sqlmon.sql脚本

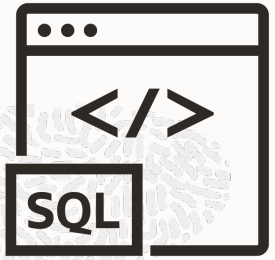
为了更具通用性，这里以SQL_ID为输入条件：

vi sqlmon.sql

```
set pagesize 0 echo off timing off linesize 1000 trimspool on trim on long 2000000 longchunksize 2000000 feedback off
spool sql_monitor_&sql_id\.htm
select dbms_sqltune.report_sql_monitor(type=>'ACTIVE', sql_id=>'&sql_id', report_level=>'ALL') monitor_report from dual;
spool off
```

SQL Monitor Report

让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目



➢ 2.执行两条SQL，并确认各自的sql_id

这次使用更明确的hints来区分是否使用In-Memory和确保都可以生成SQL Monitor Report：

--SQL1:

```
select /*+ monitor */ count(*) from L
```

sql_id: ahtu40vr8dbhu

--SQL2:

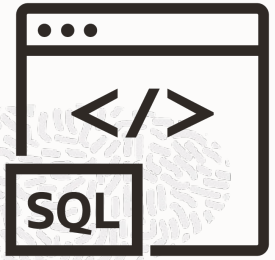
```
select /*+ monitor no_inmemory */ count(*) from L
```

sql_id: 7rzcsju067wr0

获取sql_id有多种方式，其实最方便的就是也可以通过传统看执行计划的方式来获取。

SQL Monitor Report

让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目



- 3.调用sqlmon脚本生成SQL Monitor Report

SQL> @sqlmon
Enter value for sql_id: ahtu40vr8dbhu
Enter value for sql_id: ahtu40vr8dbhu

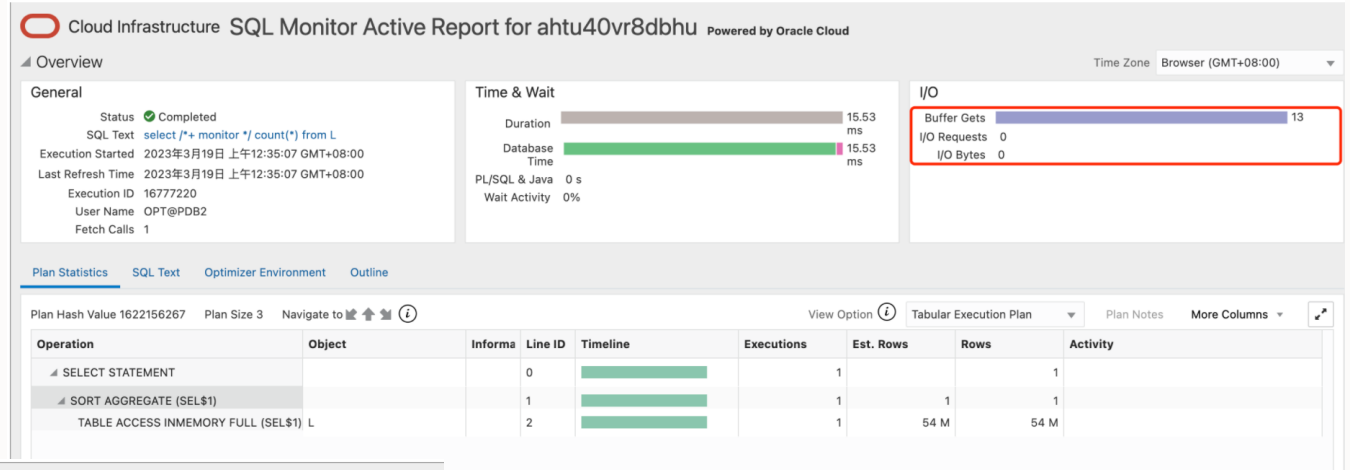
SQL> @sqlmon
Enter value for sql_id: 7rzcsju067wr0
Enter value for sql_id: 7rzcsju067wr0

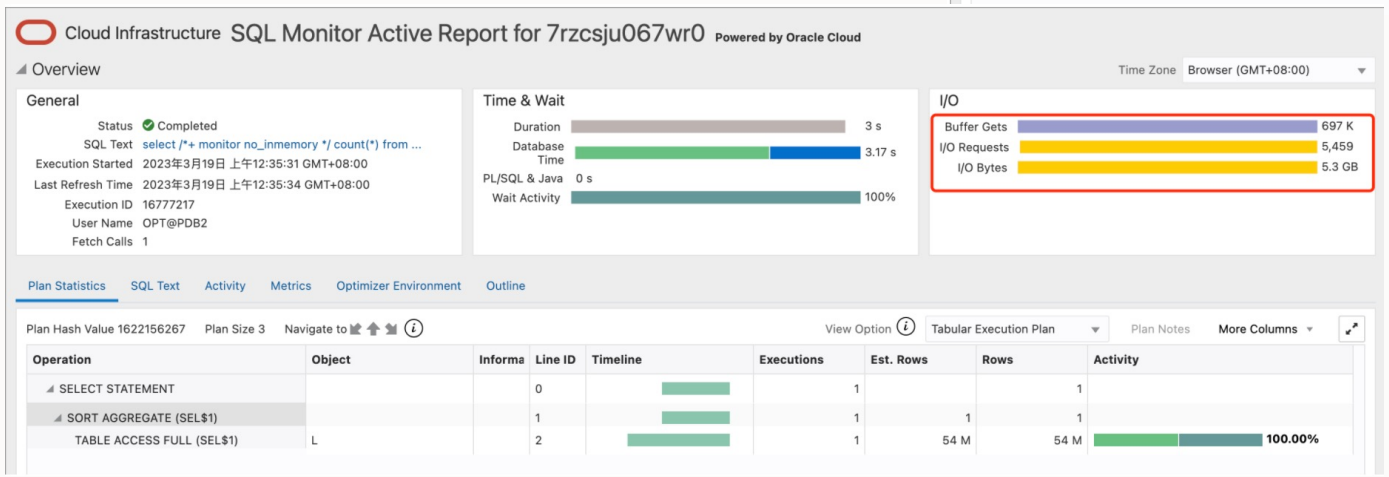
SQL Monitor Report

让看SQL执行计划这件枯燥无味的事情从此变得赏心悦目

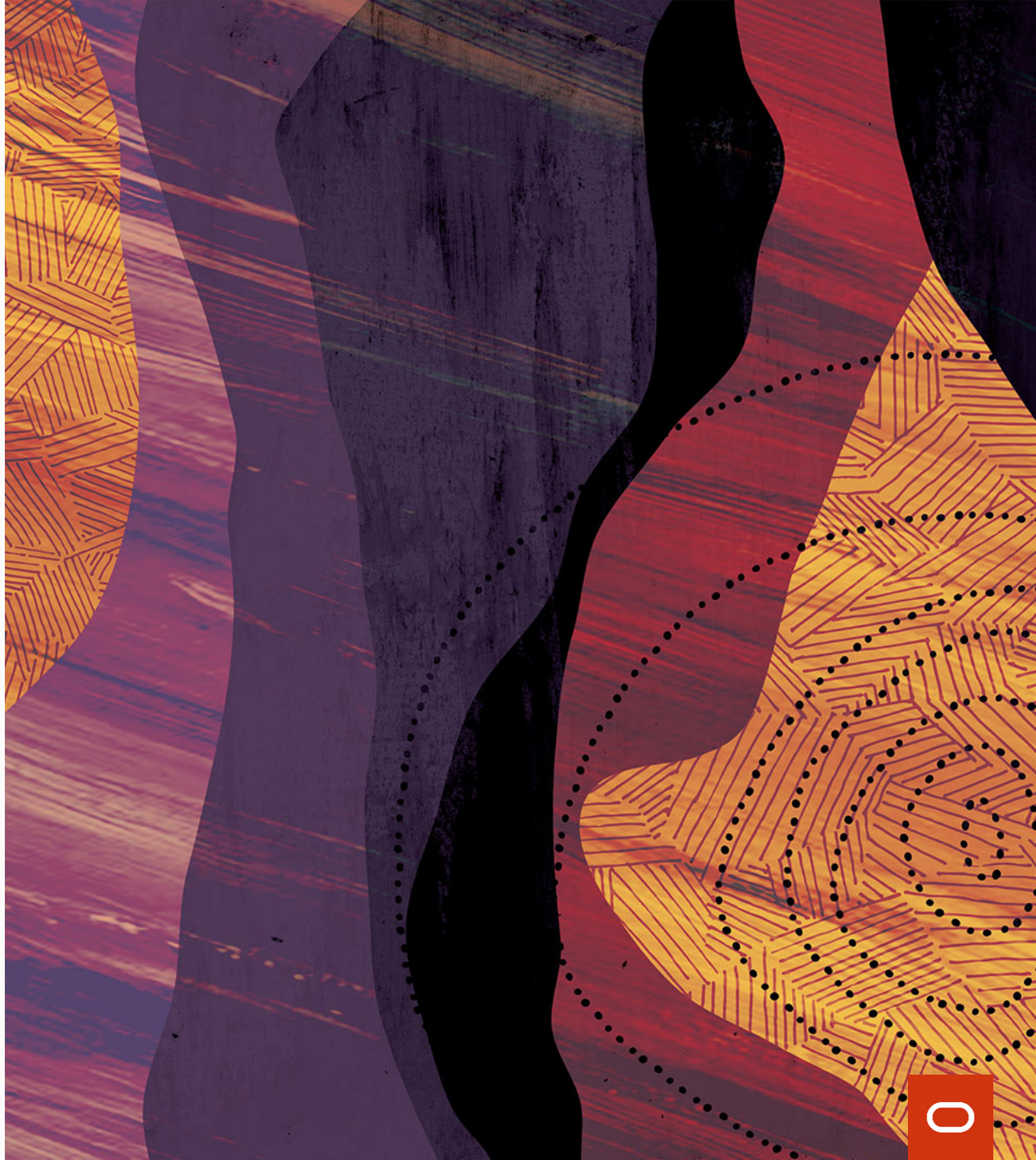
- 4.对比两个SQL Monitor Report

除了之前执行时就体验的执行时间差异，其对IO资源的实际消耗也可以非常直观的看到，二者有很大的差异。





应急场景处置



SQL Profile，SPM，hints

Oracle之SQL优化专题02-稳固SQL执行计划的方法

SQL Profile，SPM，hints

总能有手段帮助你去解决问题，但是没其他办法时才用，不能依赖！

这里以最常用的SQL Profile举例说明。

SQL Profile，SPM，hints

Oracle之SQL优化专题02-稳固SQL执行计划的方法


```
SQL> @coe_xfr_sql_profile.sql

Parameter 1:
SQL_ID (required)

Enter value for 1: dqd10y7wqrg7f


PLAN_HASH_VALUE AVG_ET_SECS
--------------- -----------
     1123238657         .095

Parameter 2:
PLAN_HASH_VALUE (required)

Enter value for 2: 1725450077

Values passed to coe_xfr_sql_profile:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SQL_ID          : "dqd10y7wqrg7f"
PLAN_HASH_VALUE: "1725450077"
...
Execute coe_xfr_sql_profile_dqd10y7wqrg7f_1725450077.sql
on TARGET system in order to create a custom SQL Profile
with plan 1725450077 linked to adjusted sql_text.


COE_XFR_SQL_PROFILE completed.
```

SQL_ID

Good PLAN_HASH_VALUE

在本次演示实验中，就是将sql_id='dqd10y7wqrg7f'的SQL绑定好的plan_hash_value=1725450077。

然后按照提示执行生成的coe_xfr_sql_profile_dqd10y7wqrg7f_1725450077.sql脚本即可；

需要特别注意的是：可以根据实际情况是否需要修改这个脚本中的force_match的值为true。

SQL Profile，SPM，hints

Oracle之SQL优化专题02-稳固SQL执行计划的方法

常用操作：

1)查询sql_profile
可以通过查询dba_sql_profiles来确认数据库中的sql_profile：
select * from dba_sql_profiles;

2)删除sql_profile
如果有一天不再需要这个sql_profile来稳固执行计划，可以这样删除sql_profile：
exec dbms_sqltune.drop_sql_profile('name');
eg: exec dbms_sqltune.drop_sql_profile('coe_dqd10y7wqrg7f_1725450077');

3)清除SQL执行计划
还可以清除共享池中指定SQL的执行计划：
exec sys.dbms_shared_pool.purge('address,hash_value','c');

SQL Profile，SPM，hints

Oracle之SQL优化专题02-稳固SQL执行计划的方法

清除SQL执行计划示例：
适用于生产系统，绑定正确的SQL Profile后，对应SQL还是没走正确执行计划的情形。

```
SQL> select sql_id, address, hash_value, plan_hash_value, sql_profile from v$sql where sql_id = 'dqd10y7wqrg7f';

SQL_ID        ADDRESS          HASH_VALUE PLAN_HASH_VALUE SQL_PROFILE
------------- ---------------- ---------- --------------- ------------------------------
dqd10y7wqrg7f 0000000076B909F8 4184587502      1123238657
dqd10y7wqrg7f 0000000076B909F8 4184587502      1123238657
dqd10y7wqrg7f 0000000076B909F8 4184587502      1725450077 coe_dqd10y7wqrg7f_1725450077

SQL> exec sys.dbms_shared_pool.purge('0000000076B909F8,4184587502','c');

PL/SQL procedure successfully completed.

SQL> select sql_id, address, hash_value, plan_hash_value, sql_profile from v$sql where sql_id = 'dqd10y7wqrg7f';

no rows selected
```

SQL_PROFILE

快速诊断的技巧?

Oracle数据库该如何着手优化一个SQL

1.首先，看SQL在等什么?

- 是否真的正在执行？还是被其他会话阻塞了?

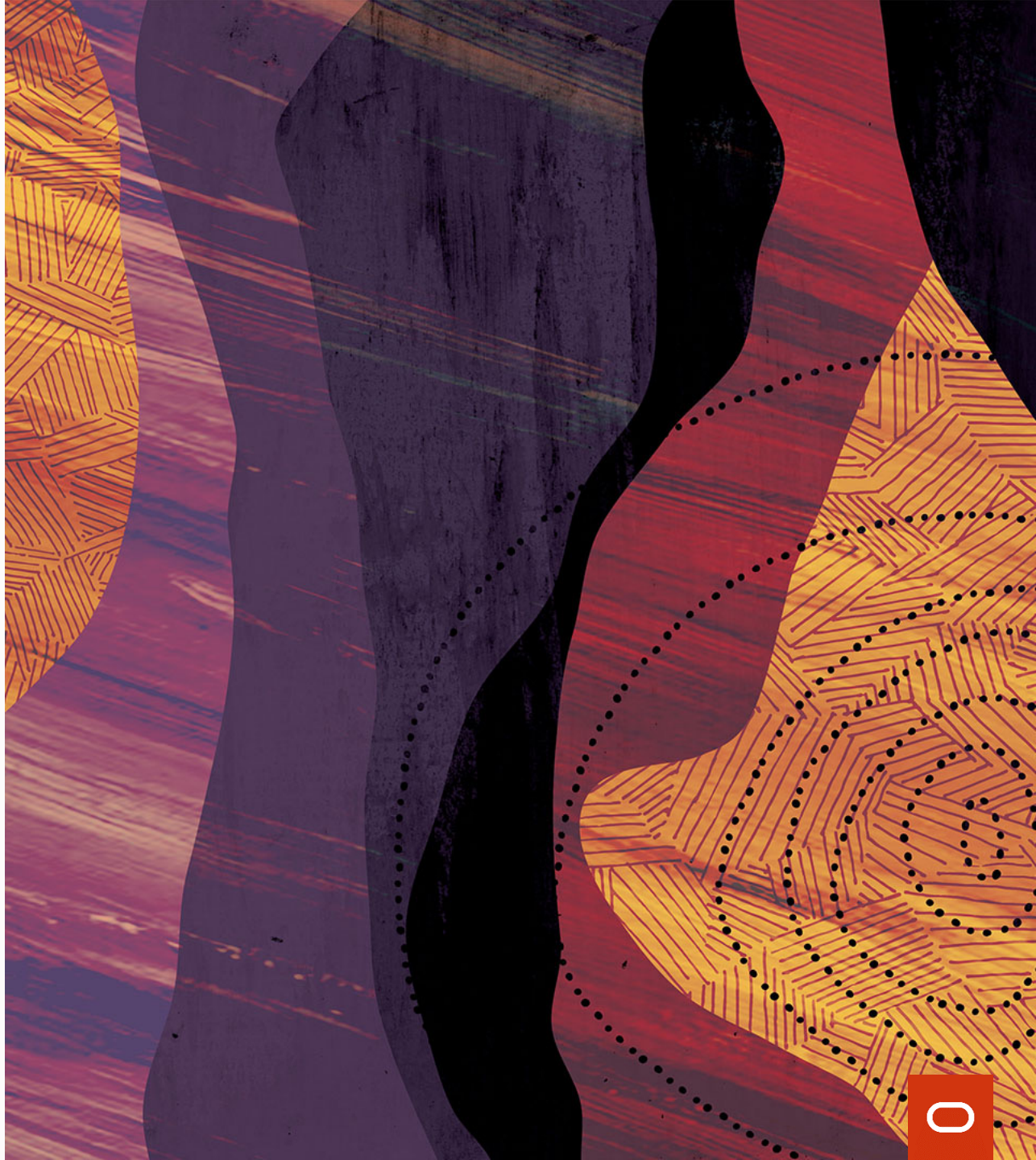
2.查看SQL执行计划

- SQL执行计划中看耗费时间占比最高的步骤？是否执行计划走错了？为何走错，好的执行计划应该是啥?
- 建议大家使用SQL Monitory Report，直接关注Activity比重大的步骤。

3.针对性优化

- 使用之前介绍的各类方法/手段来想办法优化此步骤。

总结



总结

Oracle数据库的可观测性和问题诊断实践

1.数据库层的可观测性

- Alert、ADRCI、AWR 、ASH、Hanganalyze & systemstate dumps(SSD)

2.会话层的可观测性

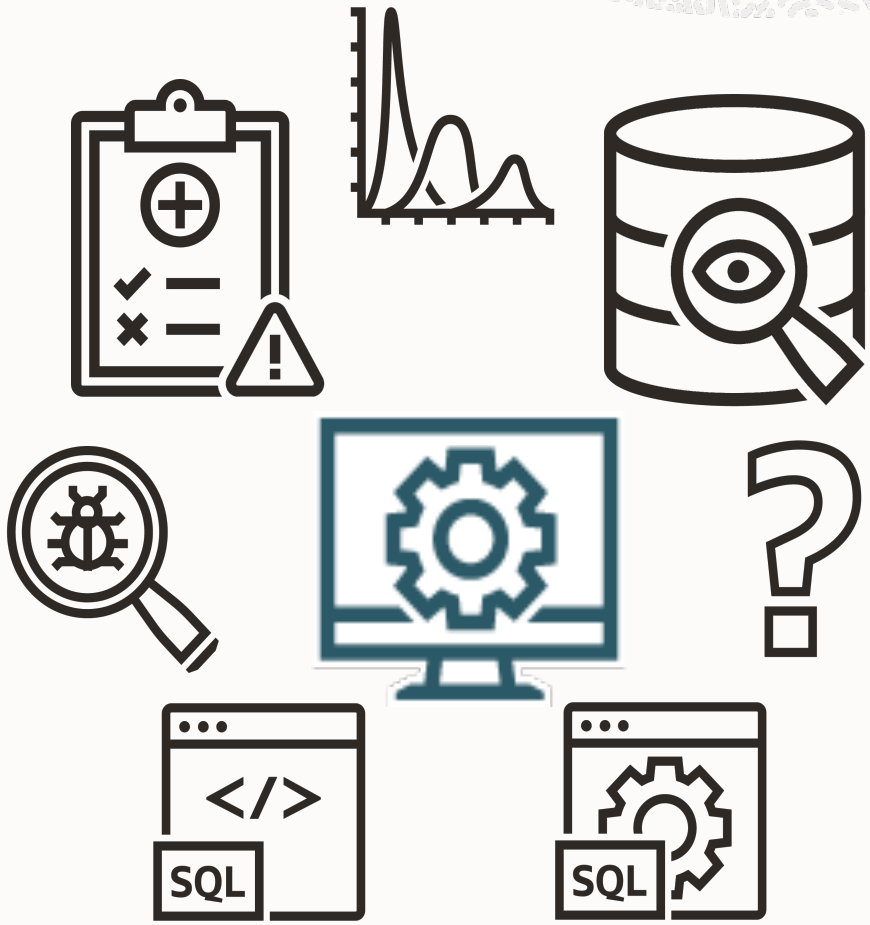
- Wait Event(接近2000) 、10046

3.优化诊断实践

- DB & OS：数据库、系统层面配置：数据库各类参数，OS大页配置等符合设计/最佳实践
- SQL：10053（了解CBO的访问路径选择等）、SQL Monitor Report

4.应急场景处置

- SQL Profile，SPM，hints，快速诊断技巧?
- 总能有手段帮助你去解决问题，但是没其他办法时才用，不能依赖!



基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

400-699-8888

* 活动最终解释权归甲骨文公司所有

数据库和云讲座群

20-21

甲骨文云技术公众号

技术专家1V1深入交流


ORACLE