ASH – Active Session History

Feel the Power

Kyle Hailey
Embarcadero Technologies
Kyle.hailey@embarcadero.com
http://Oraperf.sourceforge.net

ASH
Whole New Paradigm in Technology

The Power of ASH lies in
Simplifying Performance Tuning

- Totally new and exciting methodology
- Cheaper, Quicker, Richer and better tasting too

Why should you care?

Because ASH can Change your life …

- 10g immediately Accessible
  - For Geeks: Via scripts in SQL
  - For non-geeks (ie Marketing) :
    Graphical EM
- For those of you stuck on 7,8,9 , my apologies because 10g rocks

  But no worries, the data is accessible for you too
  - via PL/SQL data collection scripts
  - Then the power of ASH is accessible via SQL

Here’s a piece of Personal history

This is the story of how
ASH
changed my Life

Once upon a time I was stuck in dull tech support …

Finally I was promoted to head DBA, thinking my life was getting better…

It was then that I learned I needed to know what the databases were doing ALL the time

Because lives (and my job) depended on it

What could I do?!

Clearly a job for Stats Pack

Because stats pack showed me every hour

- All the Stats!
- All the top Bottlenecks
- Almost all the top SQL

So why wasn’t I feeling better?

Hmmm … but … wait

- What if my database hits a bottleneck 15 minutes after my last stats pack?!
- What about that rouge coder who writes bad SQL … can I find his module gone mad?
- Ok I found the top SQL but … now what?

If lives (and my job) depended on my databases

- Was once an hour really enough ?
- Could Stats Pack find that rogue coder?
- Could Stats Pack tell me why an SQL bottlenecked?

- Don’t tell anyone but … I couldn’t even decipher Stats Pack half the time

Oh no … it’s 3am my manager calls “why is the database hanging?!”

- Who did it?
- Where did the SQL get blocked?
- What if the bottlenecked started 5 minutes after my last Stats Pack?

In that moment of crises Stats Pack couldn’t tell me… but ASH can

The crises of my day have left the “ASHes” of today.

I didn’t have ASH but You do

Introducing ASH

A technology capable of saving lives …

(And your job)

ASH
A Revolution in Monitoring

Active Session History

- New 10g
- Every Second it collects data
- 1 hour of history in Memory for immediate access at your fingertips

This an hour of data could change your life

It’s a Revolution and it’s an Evolution

- Oracle 6 … ie the dark ages … there was once the Cache Buffer Hit Ratio
- Oracle 7 … turned the lights on … Wait Events … hallelujah I can see the light
- Oracle 10g … ASH has landed

ASH – Intelligence for the new Millennium

- Intelligently Collects Data
  - It self adjusts for your needs
  - More activity, more data collected
  - Less activity, less data collected
- Those old methods collected everything
  - Obfuscated the problem, too many statistics too late
  - Costly
  - Too Granular – once an hour ?! Give me a break

ASH – In Memory

- Collects active session data only
- History v$session_wait + v$session + extras
- Circular Buffer - 1M to 128M (~2% of SGA)
- Flushed every hour to disk or when buffer 2/3 full (it protects itself so you can relax)

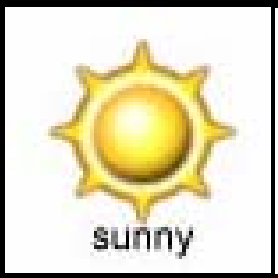
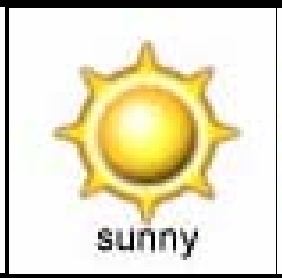
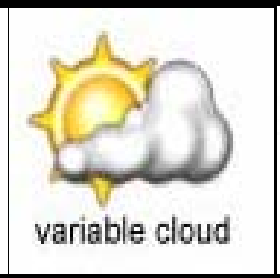
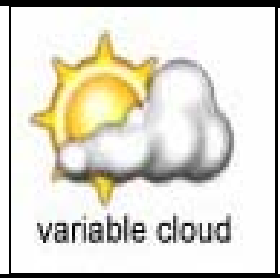
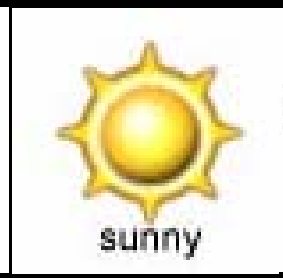
ASH Sizing …

- Avg row around 150bytes
- 3600 secs in an hour
- ~ ½ Meg per Active Session per hour
- That’s generally over an hour of ASH

ASH Samples

- What is Sampling?

Sampling Weather

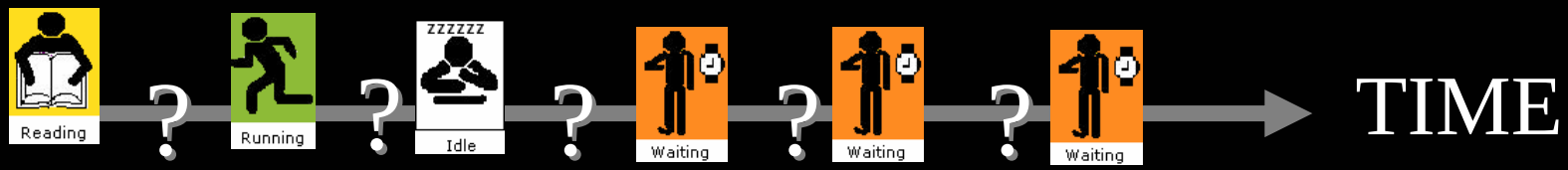
Mon	Tue	Wed	Thu	Fri	Sat	Sun
						

Weather Changes but we want the main picture

ASH Samples Session State

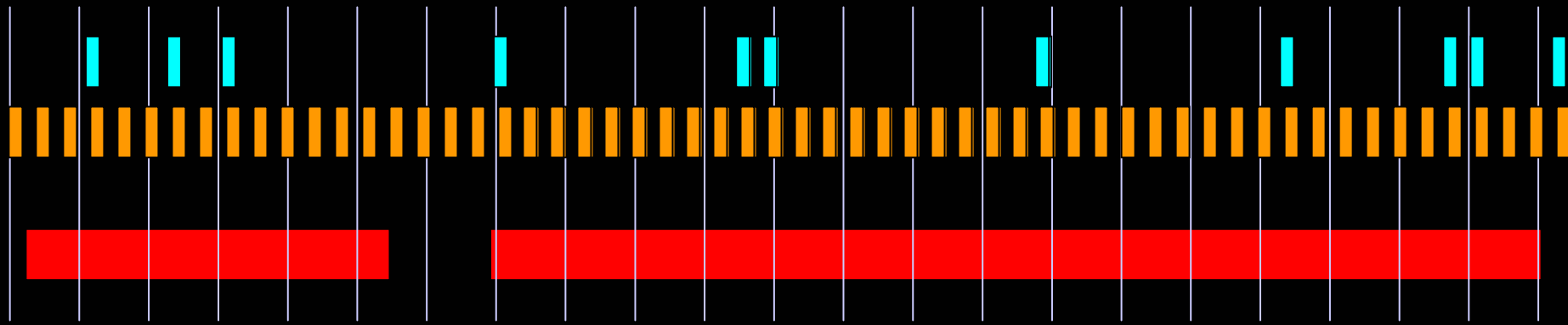
Sessions change but we want the main picture

ASH Samples Session State

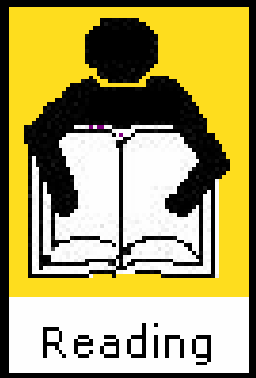


Sessions change a lot quicker but can get the main picture via sampling by sampling faster

If happens a lot or for long …
we'll catch it, guaranteed

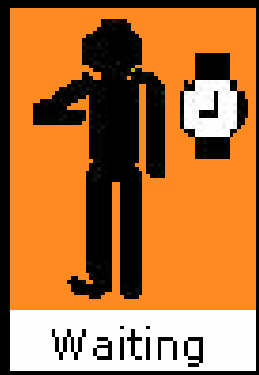


Session States

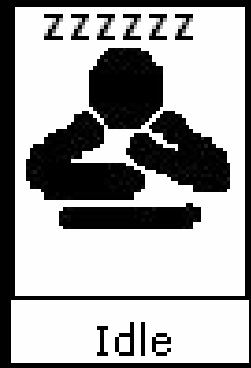


IO

CPU



Wait



Idle

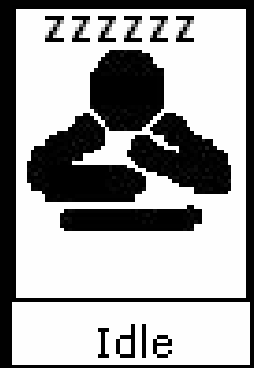
IDLE

- Ex : SQL*Net Message from Client
- All Idle Events:

  select name from v$event_name where

  wait_class='Idle‘;

  58 Rows

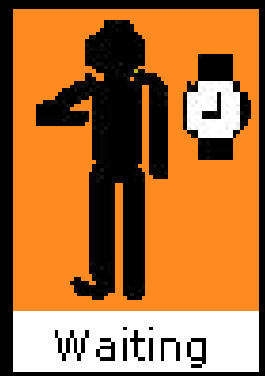


CPU

- ASH: SESSION_STATE = “ON CPU”
- ASH: wait_time > 0

WAITING

- ASH: SESSION_STATE='WAITING'
- ASH: WAIT_TIME=0
- WAIT_CLASS
  1. Administrative
  2. Application
  3. Cluster
  4. Commit
  5. Concurrency
  6. Configuration
  7. Network
  8. Other
  9. Scheduler
  10. System I/O
- 800+ WAIT



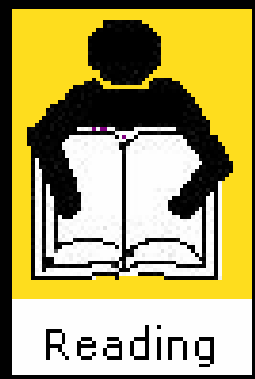
IO

- ASH:

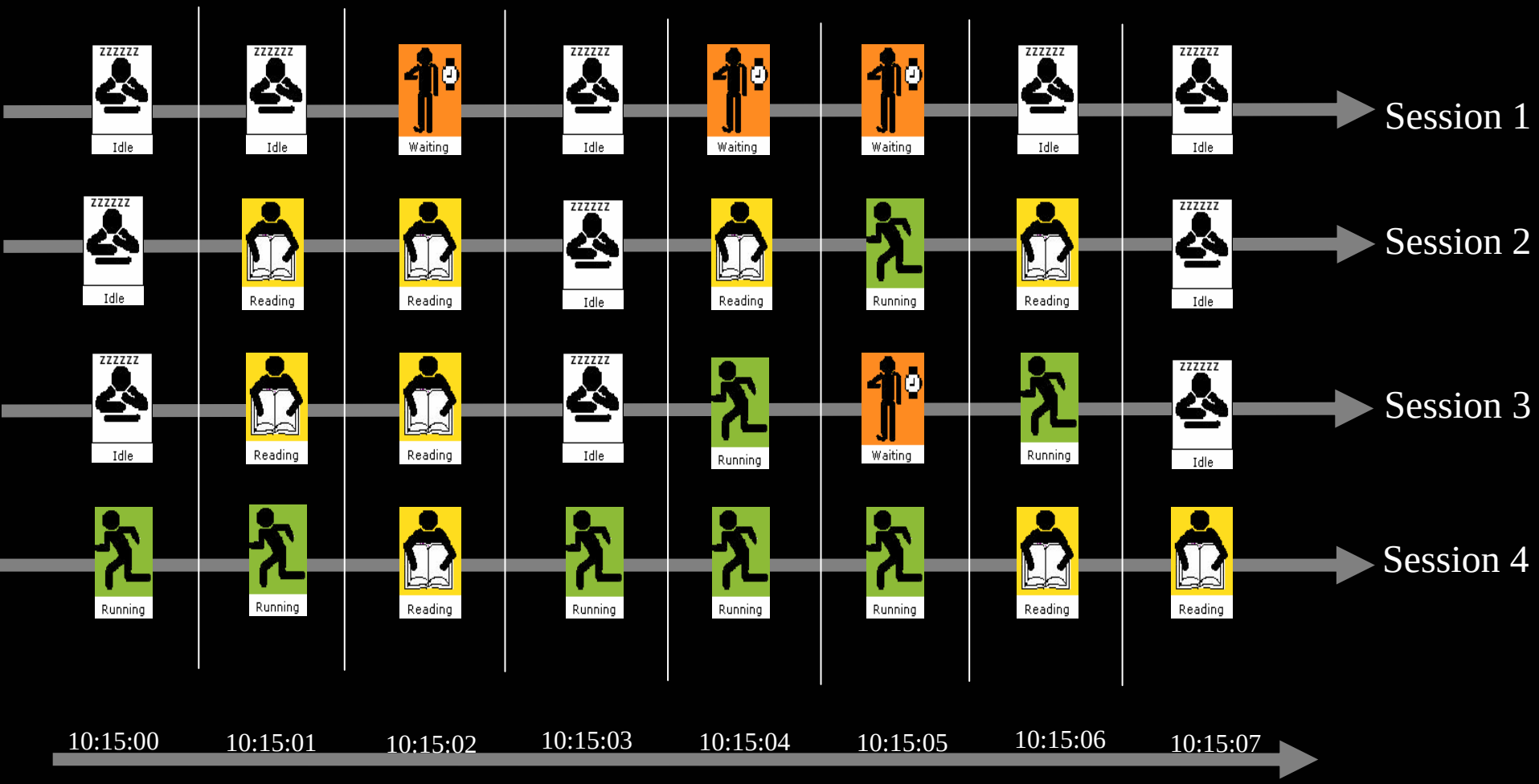
  SESSION_STATE=‘WAITING’

  and

  WAIT_CLASS=‘User I/O’



Samples for all users



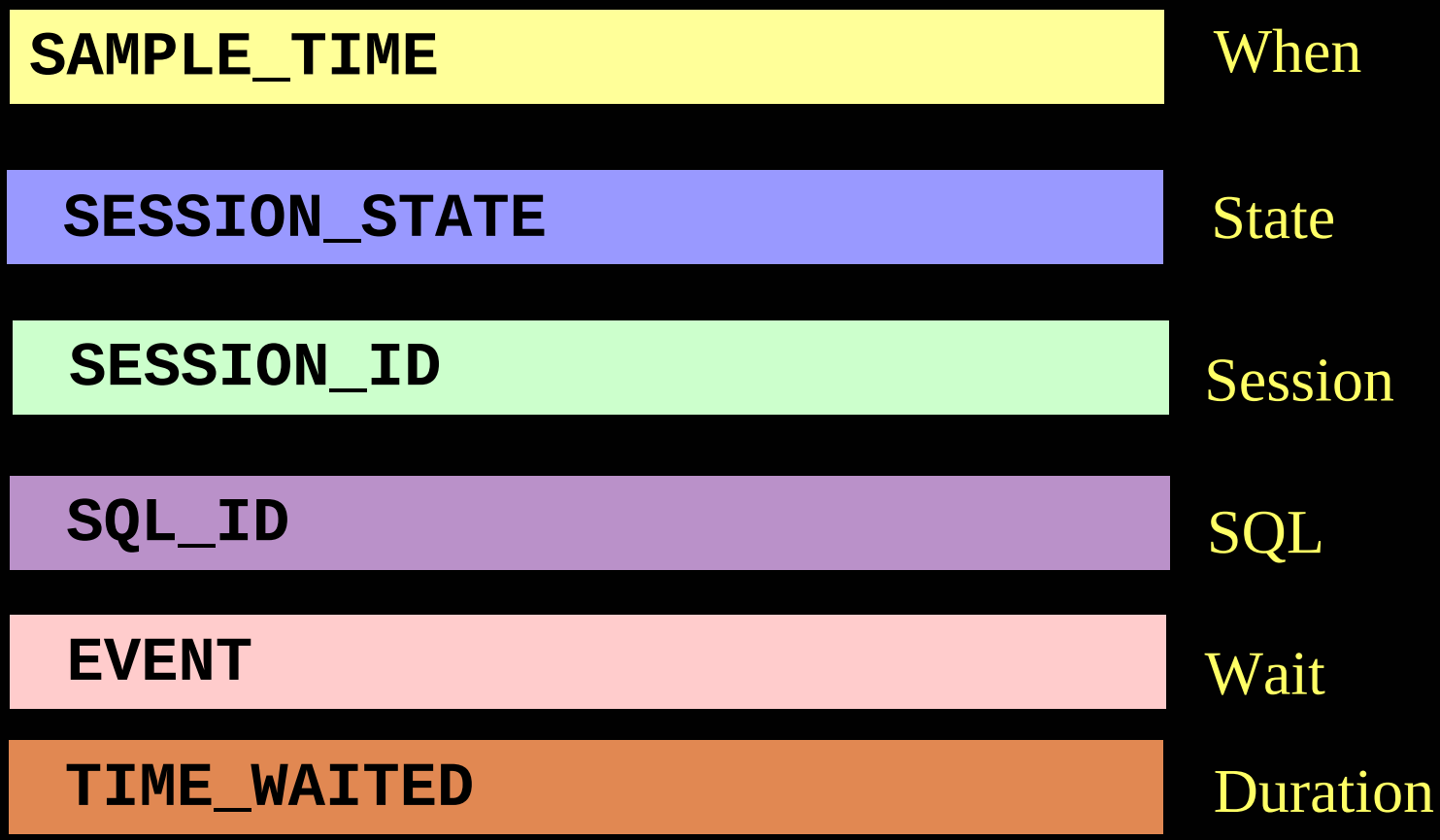
ASH Fields

```
SQL>  v$active_session_history
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SAMPLE_ID                                          NUMBER
 SAMPLE_TIME                                        TIMESTAMP(3)
 SESSION_ID                                         NUMBER
 SESSION_SERIAL#                                    NUMBER
 USER_ID                                            NUMBER
 SQL_ID                                             VARCHAR2(13)
 SQL_CHILD_NUMBER                                   NUMBER
 SQL_PLAN_HASH_VALUE                                NUMBER
 SQL_OPCODE                                         NUMBER
 SERVICE_HASH                                       NUMBER
 SESSION_TYPE                                       VARCHAR2(10)
 SESSION_STATE                                      VARCHAR2(7)
 QC_SESSION_ID                                      NUMBER
 QC_INSTANCE_ID                                     NUMBER
 EVENT                                              VARCHAR2(64)
 EVENT_ID                                           NUMBER
 EVENT#                                             NUMBER
 SEQ#                                               NUMBER
 P1                                                 NUMBER
 P2                                                 NUMBER
 P3                                                 NUMBER
 WAIT_TIME                                          NUMBER
 TIME_WAITED                                        NUMBER
 CURRENT_OBJ#                                       NUMBER
 CURRENT_FILE#                                      NUMBER
 CURRENT_BLOCK#                                     NUMBER
 PROGRAM                                            VARCHAR2(48)
 MODULE                                             VARCHAR2(48)
 ACTION                                             VARCHAR2(32)
 CLIENT_ID                                          VARCHAR2(64)
```

v$active_session_history

SAMPLE_ID	NUMBER	When
SAMPLE_TIME	TIMESTAMP(3)	
SESSION_ID	NUMBER	Session
SESSION_SERIAL#	NUMBER	
USER_ID	NUMBER	
SERVICE_HASH	NUMBER	
SESSION_TYPE	VARCHAR2(10)	
PROGRAM	VARCHAR2(64)	
MODULE	VARCHAR2(48)	
ACTION	VARCHAR2(32)	
CLIENT_ID	VARCHAR2(64)	
SESSION_STATE	VARCHAR2(7)	State
WAIT_TIME	NUMBER	
EVENT	VARCHAR2(64)	Wait
EVENT_ID	NUMBER	
EVENT#	NUMBER	
SEQ#	NUMBER	
P1	NUMBER	
P2	NUMBER	
P3	NUMBER	
WAIT_TIME	NUMBER	
TIME_WAITED	NUMBER	
CURRENT_OBJ#	NUMBER	
CURRENT_FILE#	NUMBER	
CURRENT_BLOCK#	NUMBER0	
SQL_ID	VARCHAR2(13)	SQL
SQL_CHILD_NUMBER	NUMBER	
SQL_PLAN_HASH_VALUE	NUMBER	
SQL_OPCODE	NUMBER	
QC_SESSION_ID	NUMBER	
QC_INSTANCE_ID	NUMBER	
TIME_WAITED	NUMBER	Duration

Primary Fields of ASH



Amazing things YOU can do with ASH

Resource Consumers

Consumers

- Top Session
- Top User
- Top SQL
- Top Object
- Top Module.Action
- Top Program
- Top Service
- Top Client
- Top Wait

(32 columns in ASH)

X

Resources

- CPU
- Waits
  - Event (800*)
- I/O
  - File
  - Block
- Time

Consumer Columns

	SAMPLE_ID	NUMBER
	SAMPLE_TIME	TIMESTAMP(3)
*	SESSION_ID	NUMBER
	SESSION_SERIAL#	NUMBER
*	USER_ID	NUMBER
*	SERVICE_HASH	NUMBER
	SESSION_TYPE	VARCHAR2(10)
*	PROGRAM	VARCHAR2(64)
*	MODULE	VARCHAR2(48)
*	ACTION	VARCHAR2(32)
*	CLIENT_ID	VARCHAR2(64)
	SESSION_STATE	VARCHAR2(7)
	WAIT_TIME	NUMBER
*	EVENT	VARCHAR2(64)
	EVENT_ID	NUMBER
	EVENT#	NUMBER
	SEQ#	NUMBER
	P1	NUMBER
	P2	NUMBER
	P3	NUMBER
	WAIT_TIME	NUMBER
	TIME_WAITED	NUMBER
	CURRENT_OBJ#	NUMBER
	CURRENT_FILE#	NUMBER
	CURRENT_BLOCK#	NUMBER0
*	SQL_ID	VARCHAR2(13)
	SQL_CHILD_NUMBER	NUMBER
	SQL_PLAN_HASH_VALUE	NUMBER
	SQL_OPCODE	NUMBER
	QC_SESSION_ID	NUMBER
	QC_INSTANCE_ID	NUMBER
	TIME_WAITED	NUMBER

When

Session

State

Wait

SQL

Duration

Groupings – Top Consumer

```
SESSION_ID
SESSION_SERIAL# (signal SID reuse)
SESSION_TYPE    (FOREGROUND,BACKGROUND)
```

```
USER_ID      (SYS, SYSTEM, SCOTT etc)
```

```
SERVICE_HASH (OE,GL,HR)
```

```
MODULE.ACTION(PLSQL tagging)
```

```
CLIENT_ID    (customers connecting via session pool)
```

```
PROGRAM      (SQL, JDBC, Forms etc)
```

```
SQL_ID
QC_SESSION_ID  - Query Coordinator
QC_INSTANCE_ID
```

```
EVENT + P1, P2, P3
```

```
CURRENT_OBJ#
CURRENT_FILE#
CURRENT_BLOCK#
```

Top CPU Session

Who is the rogue session ?

```
Select
   session_id,
   count(*)
from
   v$active_session_history
where
   session_state= ‘ON CPU‘ and
   SAMPLE_TIME > sysdate – (5/(24*60))
group by
   session_id
order by
   count(*) desc;
```

Results Top CPU Session

SESSION_ID	COUNT(*)
265	256
264	15
257	12
271	12
276	1

CPU with Bars

SESSION_ID	COUNT(*)	%	Bar
257	75	25	***
263	62	21	**
256	32	11	*
264	9	3	
277	3	1	
258	1	0	
280	1	0	

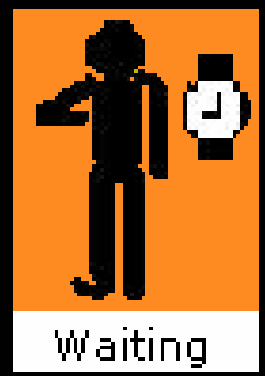
Top Waiting Session

- Top Waiting Session in last 5 minutes

```
Select
    session_id,
    count(*)
from
    v$active_session_history
where
    session_state='WAITING'  and
    SAMPLE_TIME >  SYSDATE - (5/(24*60))
group by
    session_id
order by
    count(*) desc;
```

Top Waiting Session Results

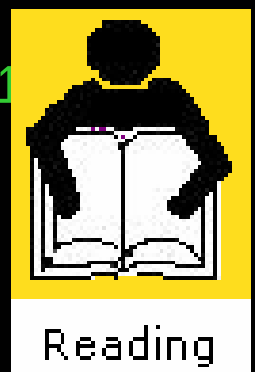
SESSION_ID	COUNT(*)
272	224
254	8
249	5
276	5
277	4
270	1

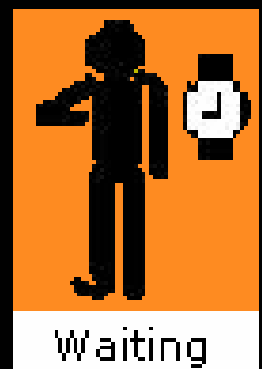


Top SQL from ASH

Top Categories of Resource usage – IO, CPU, WAIT

```
select
    ash.SQL_ID ,
    sum(decode(ash.session_state,'ON CPU',1,0))     "CPU",
    sum(decode(ash.session_state,'WAITING',1,0))    -
    sum(decode(ash.session_state,'WAITING', decode(en.wait_class, 'User I/O',1,0),0))    "WAIT" ,
    sum(decode(ash.session_state,'WAITING', decode(en.wait_class, 'User I/O',1,0),0))    "IO" ,
    sum(decode(ash.session_state,'ON CPU',1,1))     "TOTAL"
from v$active_session_history ash,
        v$event_name en
where SQL_ID is not NULL  and en.event#=ash.event#
group by sql_id
order by sum(decode(session_state,'ON CPU',1,1
```





Top SQL from ASH Results

SQL_ID	CPU	WAITING	IO	TOTAL
4c1xvq9ufwcjc	**23386**	**0**	**0**	**23386**
6wjw6rz5uvbp3	99	0	23	122
968dm8hr9qd03	97	0	22	119
938jp5gasmrah	90	0	25	115
cv8xnv81kf582	42	0	9	51
6p9bzu19v965k	21	0	0	21
5zu8pxnun66bu	15	0	0	15
db2jr13nup72v	9	0	0	9
7ks5gnj38hghv	8	0	0	8
38gfa1vpmwvx6	5	0	0	5
6zw21jfzjsunv	5	0	0	5
78s8yj36j2w1t	4	1	0	5
6769wyy3yf66f	4	0	0	4
aptc882suuy74	4	0	0	4

Top Session

```
select
    ash.session_id,
    ash.session_serial#,
    ash.user_id,
    ash.program,
    sum(decode(ash.session_state,'ON CPU',1,0))     "CPU",
    sum(decode(ash.session_state,'WAITING',1,0))    -
    sum(decode(ash.session_state,'WAITING',
      decode(en.wait_class,'User I/O',1, 0 ), 0))    "WAITING" ,
    sum(decode(ash.session_state,'WAITING',
      decode(en.wait_class,'User I/O',1, 0 ), 0))    "IO" ,
    sum(decode(session_state,'ON CPU',1,1))     "TOTAL"
from v$active_session_history ash,
      v$event_name en
where en.event# = ash.event#
group by session_id,user_id,session_serial#,program
order by sum(decode(session_state,'ON CPU',1,1))
```

Top Session Results

SESSION_ID	SERIAL#	USER_ID	PROGRAM	CPU	WAITING	IO
247	**61970**	**1**	**sqlplus**	**11698**	**0**	**0**
277	1	0	oracle@labsfrh903 (LGWR)	14	21	0
276	1	0	oracle@labsfrh903 (CKPT)	19	10	0
278	1	0	oracle@labsfrh903 (DBW0)	29	0	0
280	1	0	oracle@labsfrh903 (PMON)	19	0	0
254	22617	5	Executor.exe	13	0	3
255	12877	5	Executor.exe	11	0	5
257	33729	5	Executor.exe	15	0	1
255	13417	5	Executor.exe	14	0	2

Top Session w/ Username

```
select
      /* if  sid not found in v$session then  disconnected */
      decode(nvl(to_char(s.sid),-1),-1,'DISCONNECTED','CONNECTED')
                                        "STATUS",
      topsession.session_id             "SESSION_ID",
      u.name  "NAME",
      topsession.program                 "PROGRAM",
      max(topsession.CPU)               "CPU",
      max(topsession.WAITING)           "WAITING",
      max(topsession.IO)                "IO",
      max(topsession.TOTAL)              "TOTAL"
      from (   previous query   )       topsession,
                                        v$session s,
                                        user$ u
  where
               u.user# =topsession.user_id and
              /* outer join to v$session because the session might be disconnected */
              topsession.session_id         = s.sid         (+) and
              topsession.session_serial# = s.serial#    (+)
  group by  topsession.session_id, topsession.session_serial#, topsession.user_id,
              topsession.program, s.username,s.sid,s.paddr,u.name
  order by max(topsession.TOTAL) desc
```

Top Session
Finding a Rogue User

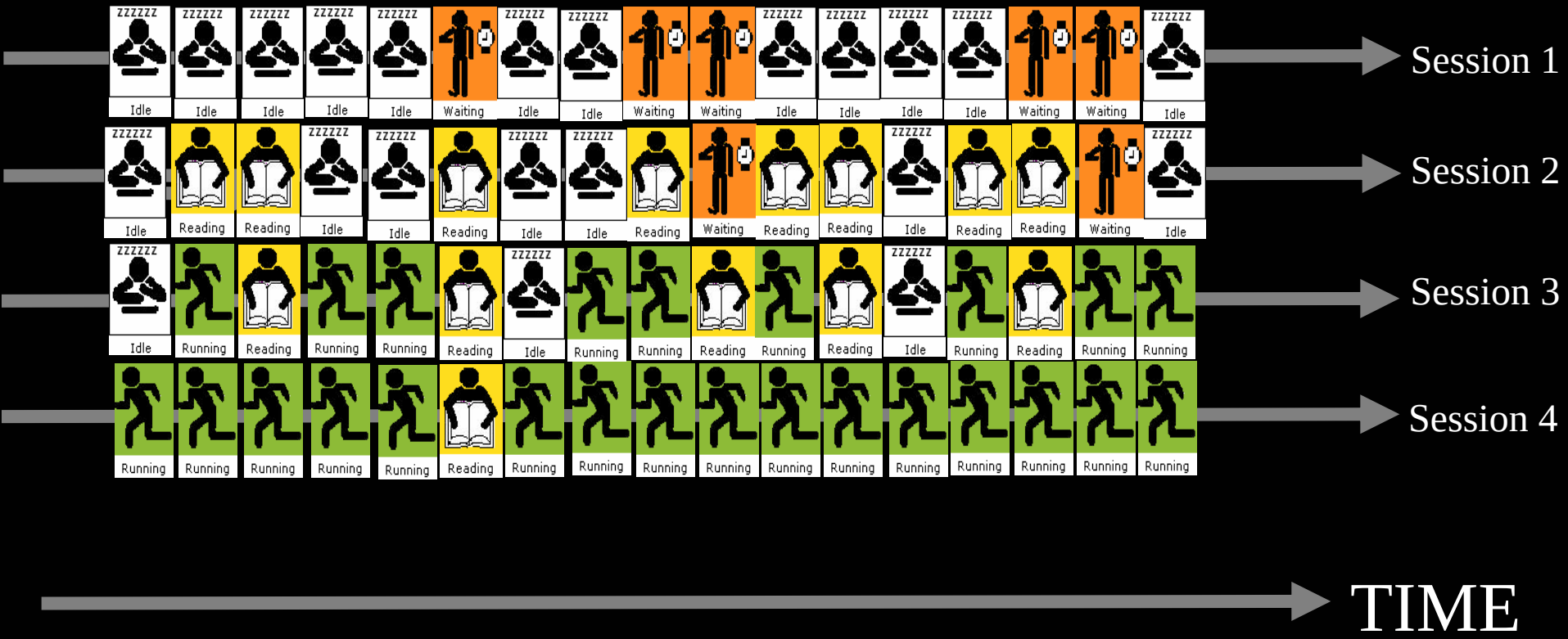
STATUS	SESSION_ID	NAME	PROGRAM	CPU	WAITING	IO
CONNECTED	**247**	**CPU_Monger**	**ChMgr304.exe**	**11704**	**0**	**0**
CONNECTED	277	SYS	oracle@labsfrh903 (LGWR)	14	19	0
CONNECTED	278	SYS	oracle@labsfrh903 (DBW0)	29	0	0
CONNECTED	276	SYS	oracle@labsfrh903 (CKPT)	18	9	0
CONNECTED	280	SYS	oracle@labsfrh903 (PMON)	20	0	0
DISCONNECTED	255	SYSTEM	Executor.exe	11	4	5
DISCONNECTED	257	SYSTEM	Executor.exe	13	0	3
DISCONNECTED	255	SYSTEM	Executor.exe	14	0	2
DISCONNECTED	257	SYSTEM	Executor.exe	13	0	3
DISCONNECTED	254	SYSTEM	Executor.exe	12	0	3
DISCONNECTED	254	SYSTEM	Executor.exe	13	0	2
DISCONNECTED	256	SYSTEM	Executor.exe	11	0	4
DISCONNECTED	256	SYSTEM	Executor.exe	12	0	3
DISCONNECTED	256	SYSTEM	Executor.exe	13	0	2
DISCONNECTED	255	SYSTEM	Executor.exe	14	0	1
DISCONNECTED	254	SYSTEM	Executor.exe	11	0	4
DISCONNECTED	246	SYSTEM	Executor.exe	11	0	3
DISCONNECTED	246	SYSTEM	Executor.exe	13	0	1

Top Wait

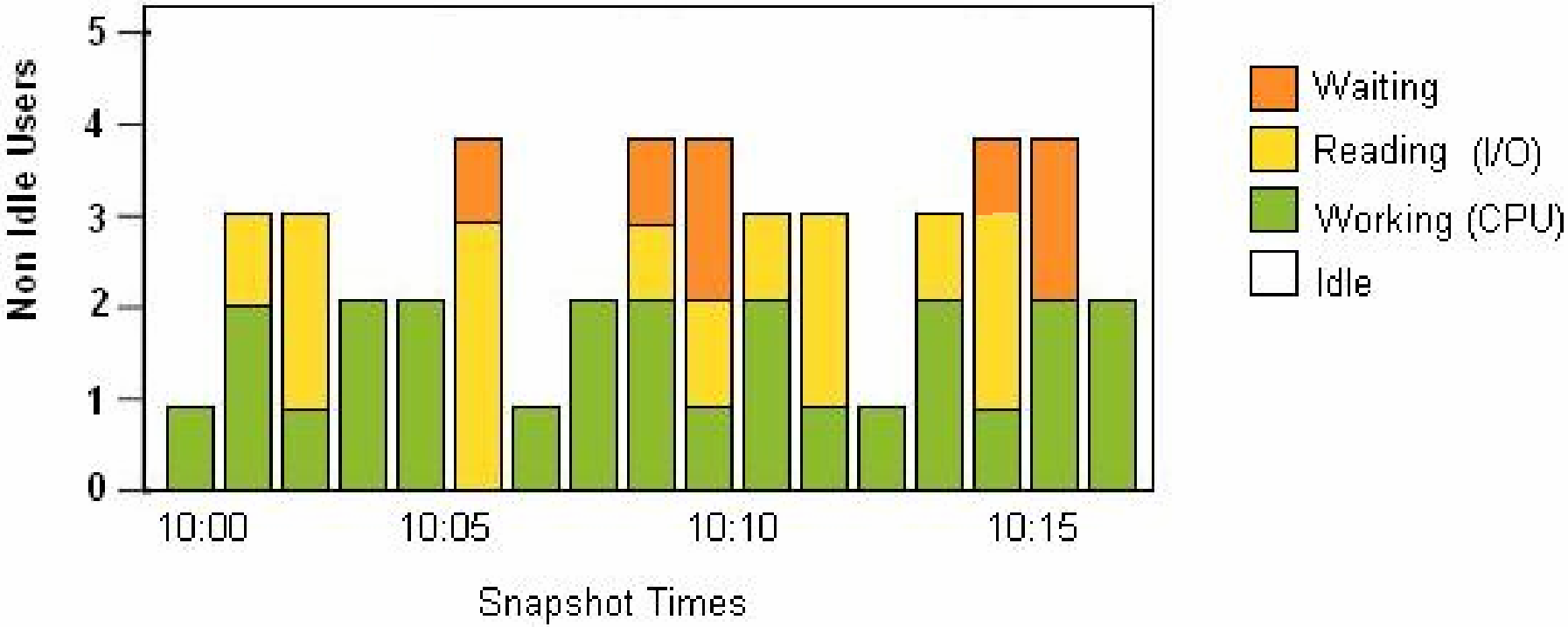
- How to Attack the problem?
- Top SQL?
  - Top wait for that SQL?
- Top Waiting Session ?
  - Top Waits for that Session
- Top Waits for Database?
  - Top Session waiting for that wait
  - Top SQL for that wait

With Ash you can attack the problem all these ways

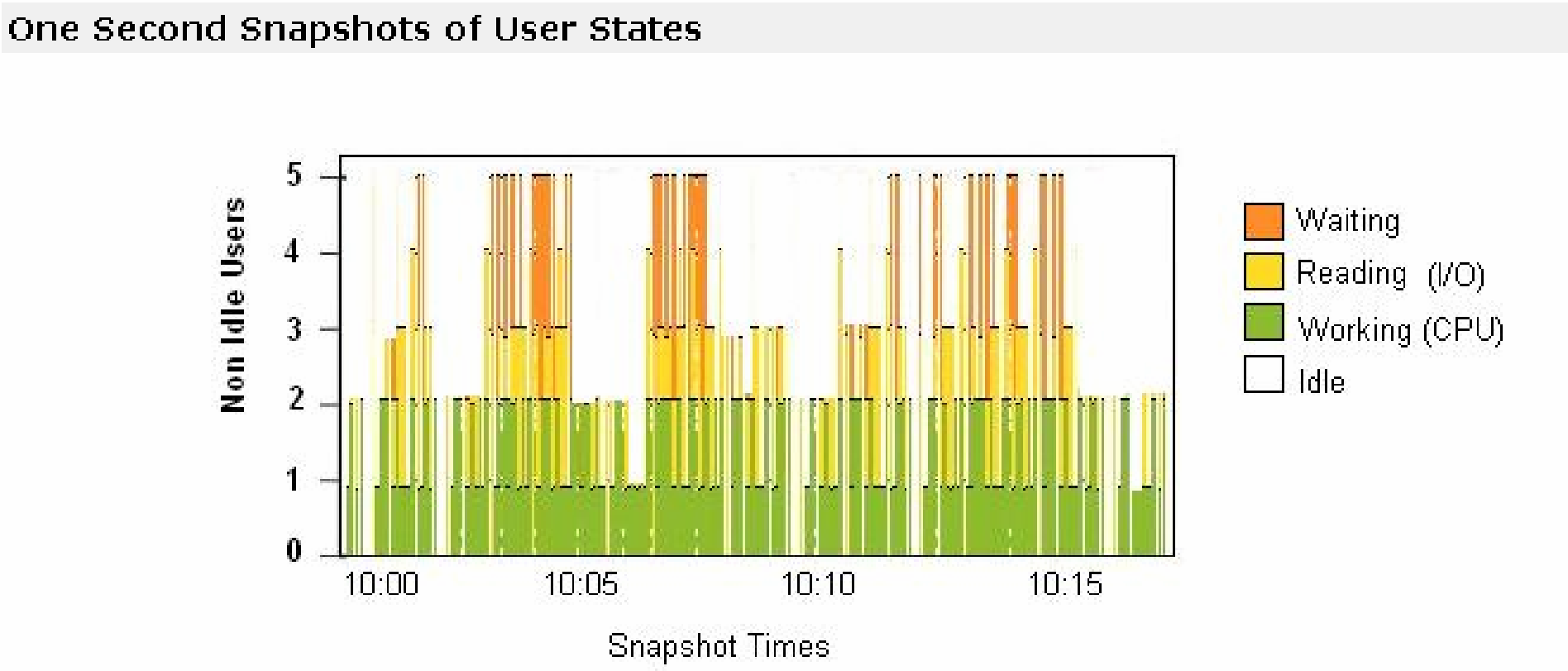
Graphical ASH



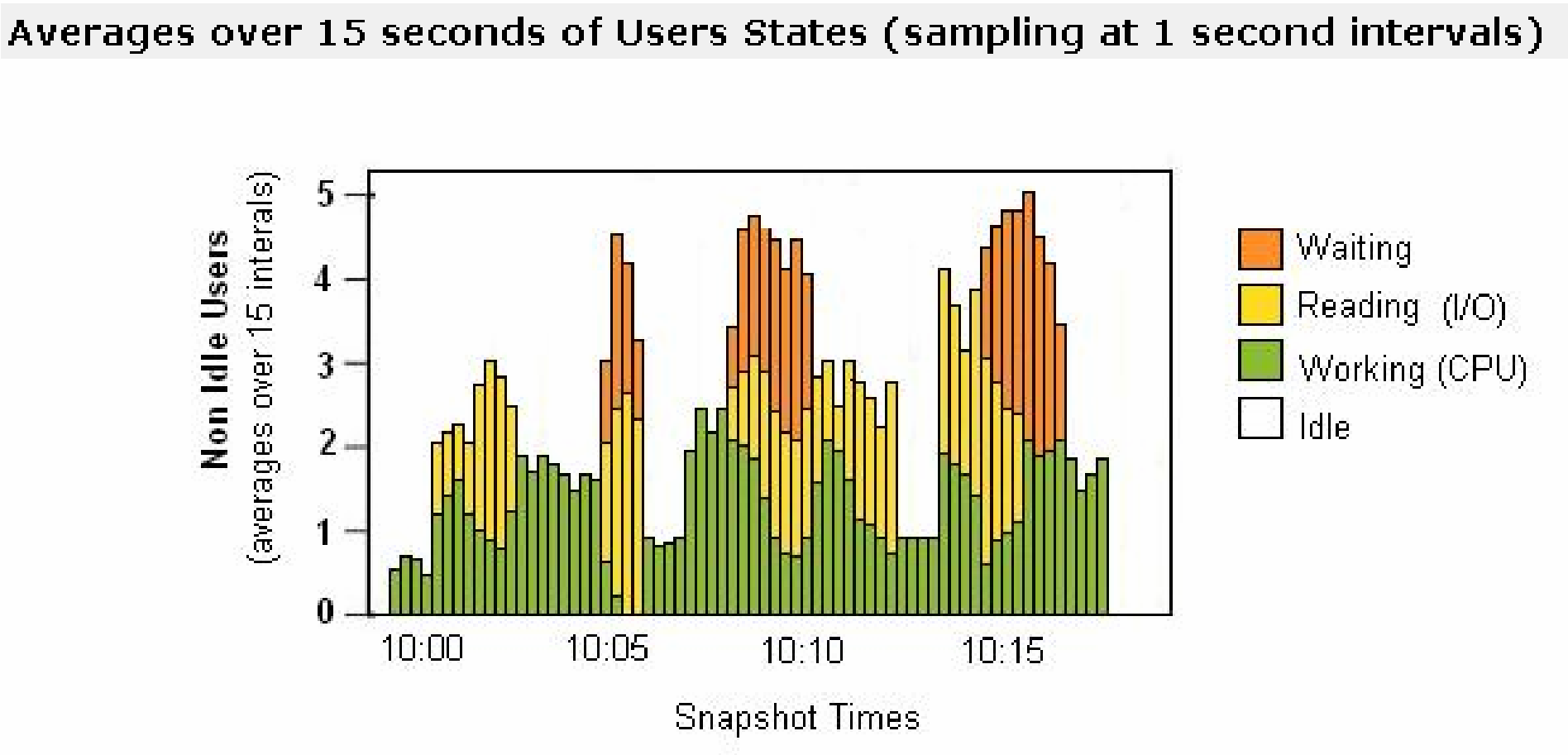
Graph of User States



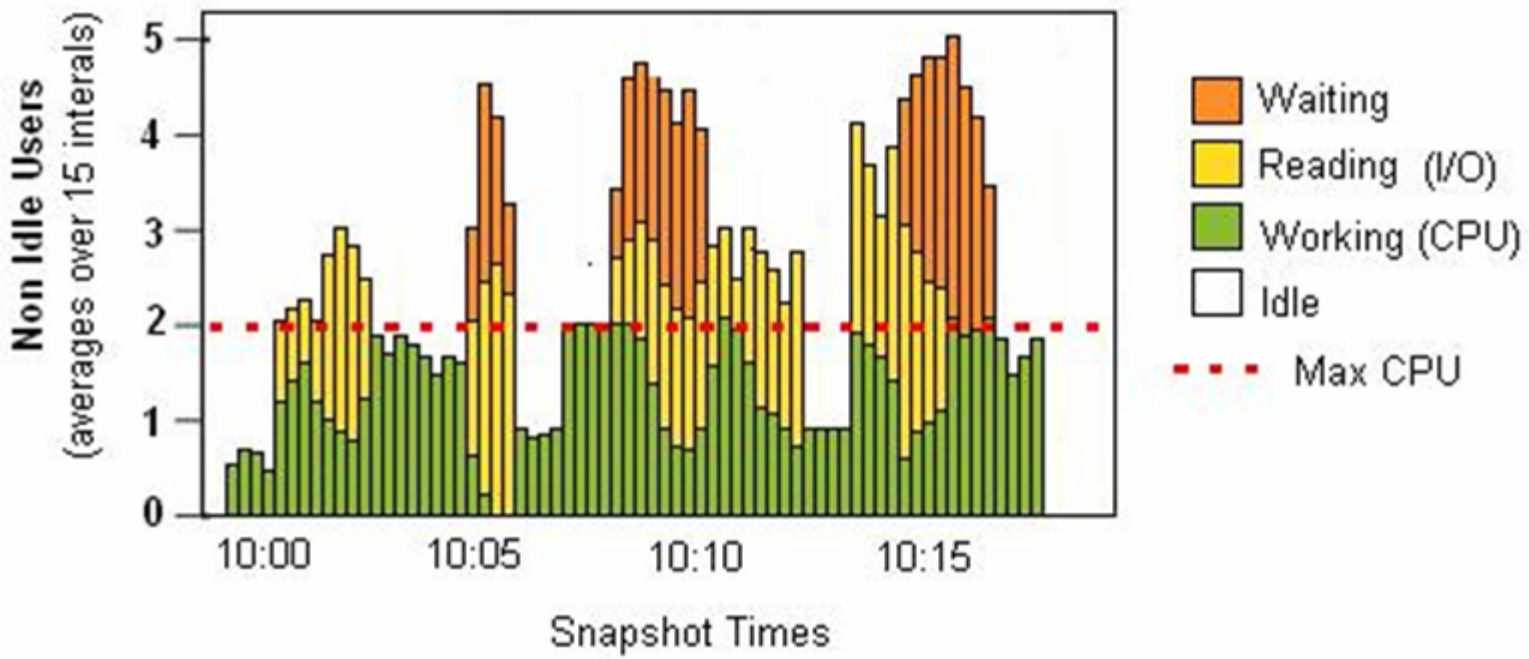
One Second Graph



15 Second Averages

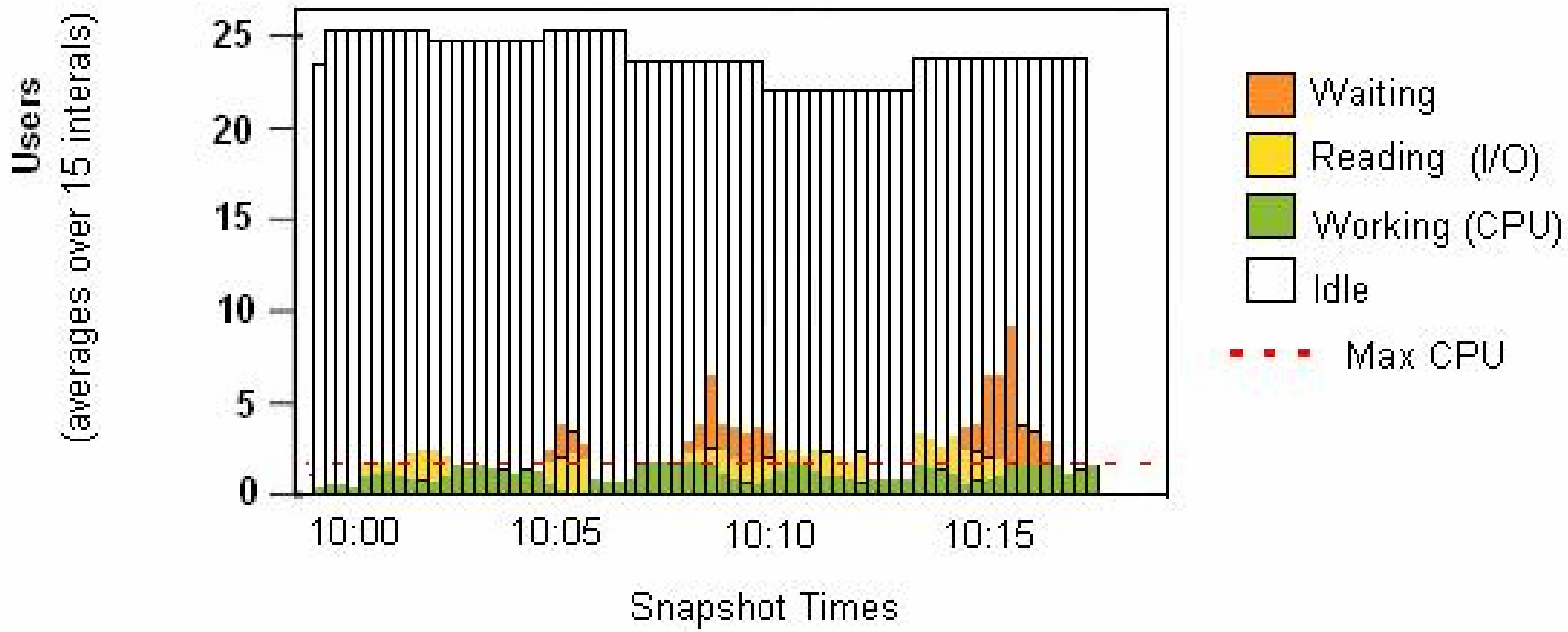


Maximum CPU Line

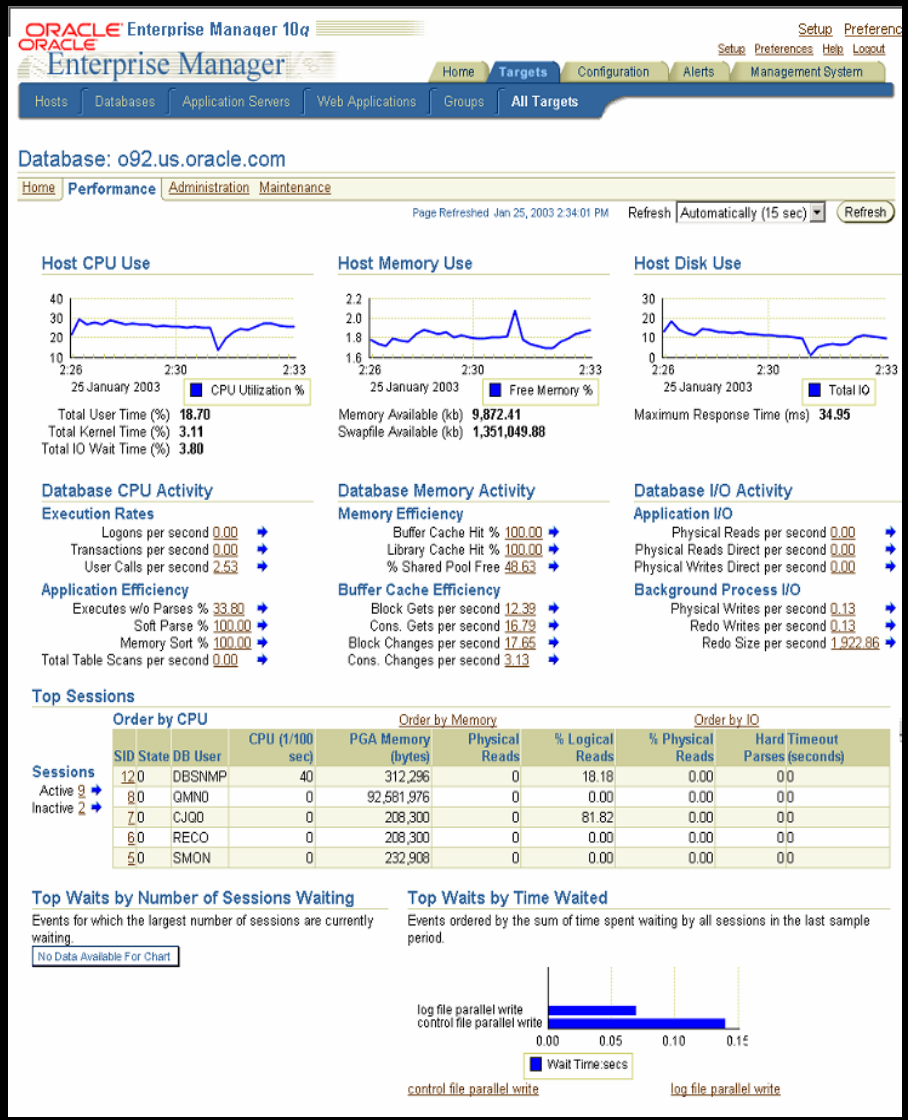


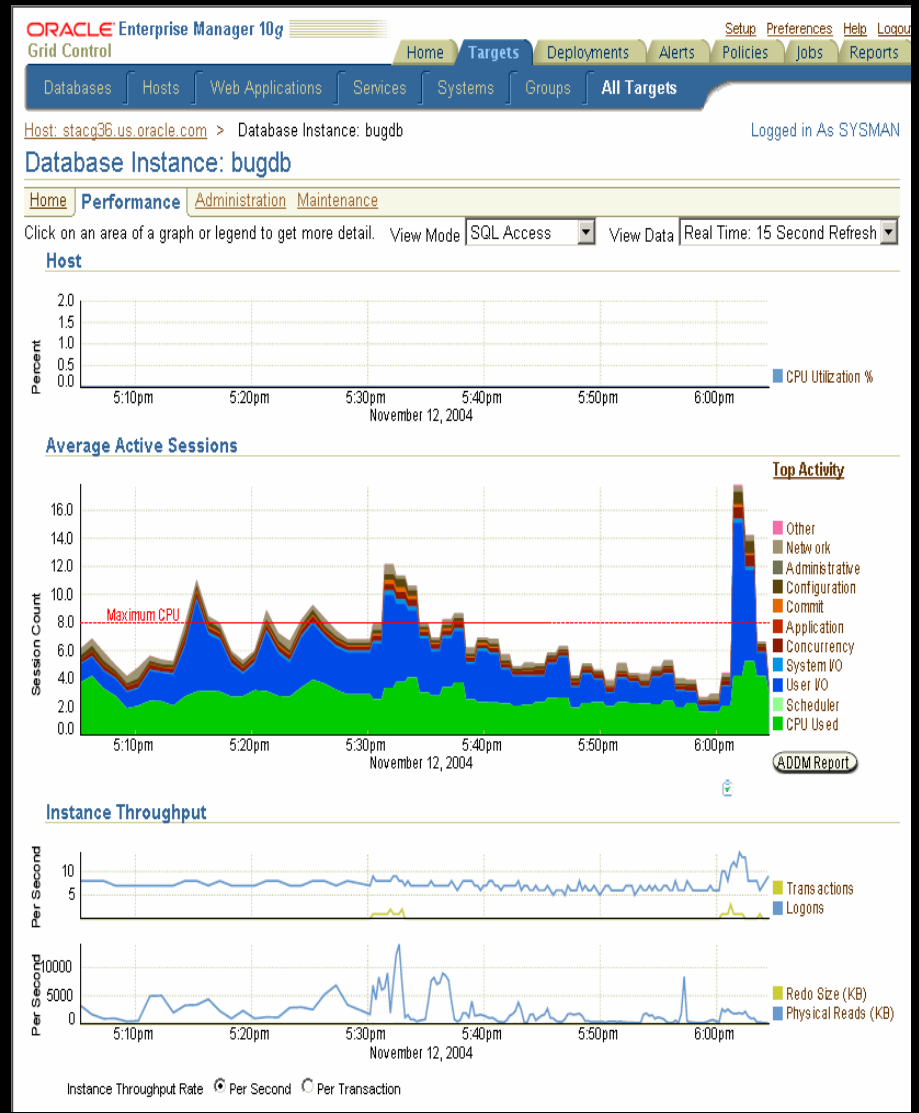
Idle Users

Why Don't We Display Idle Users in the Chart?

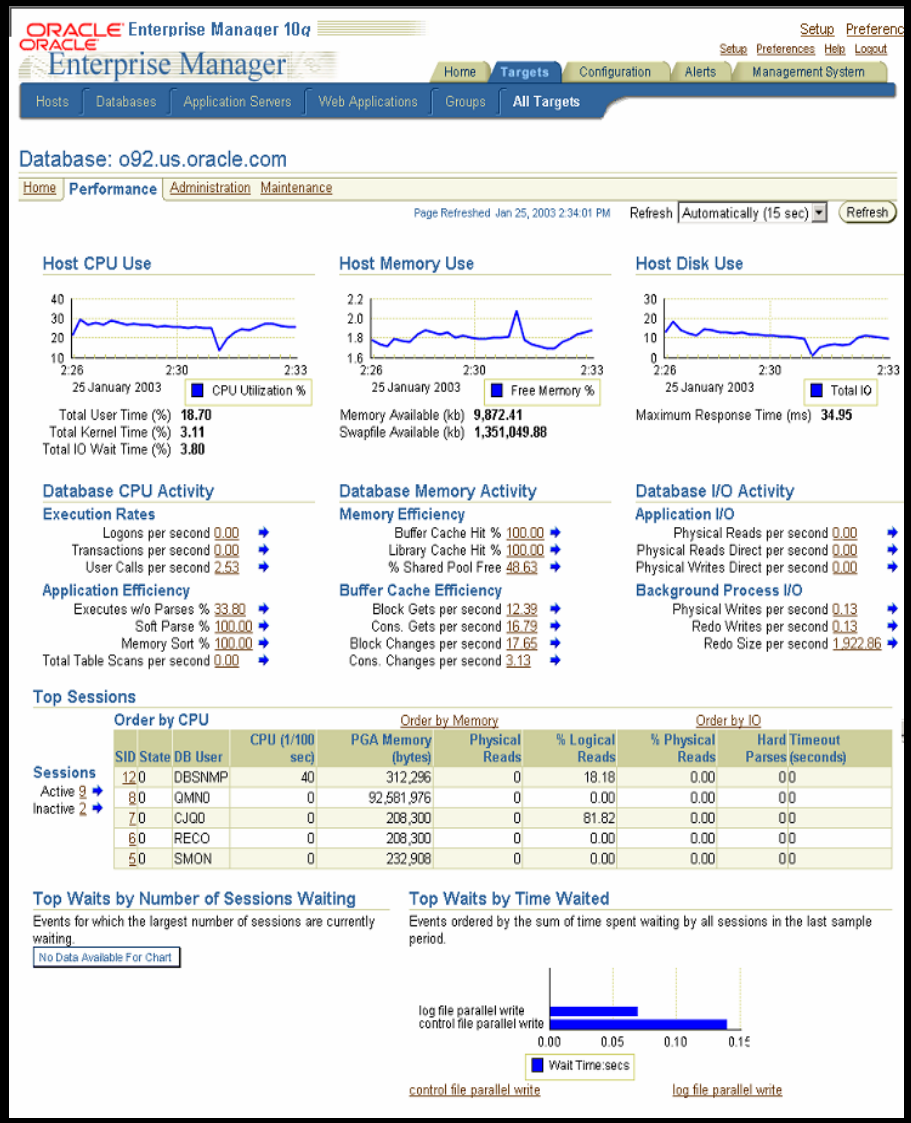


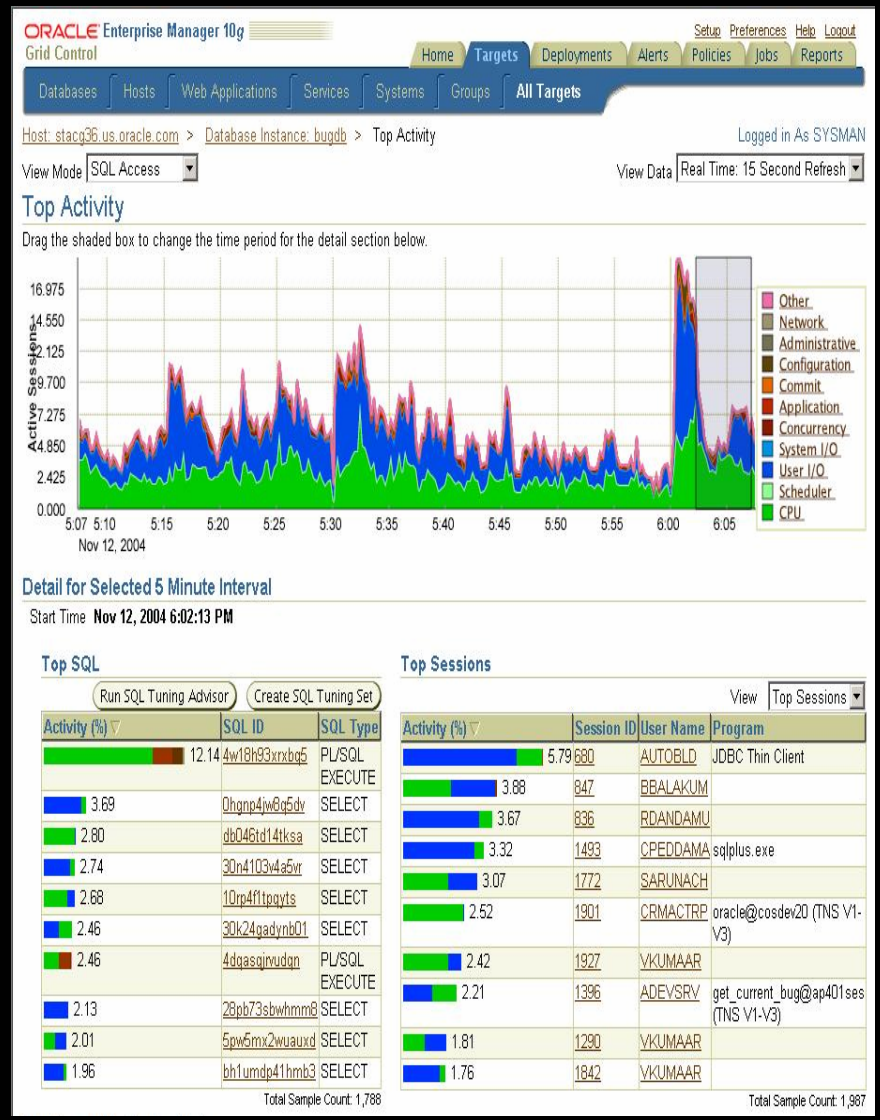
OEM Perf Page



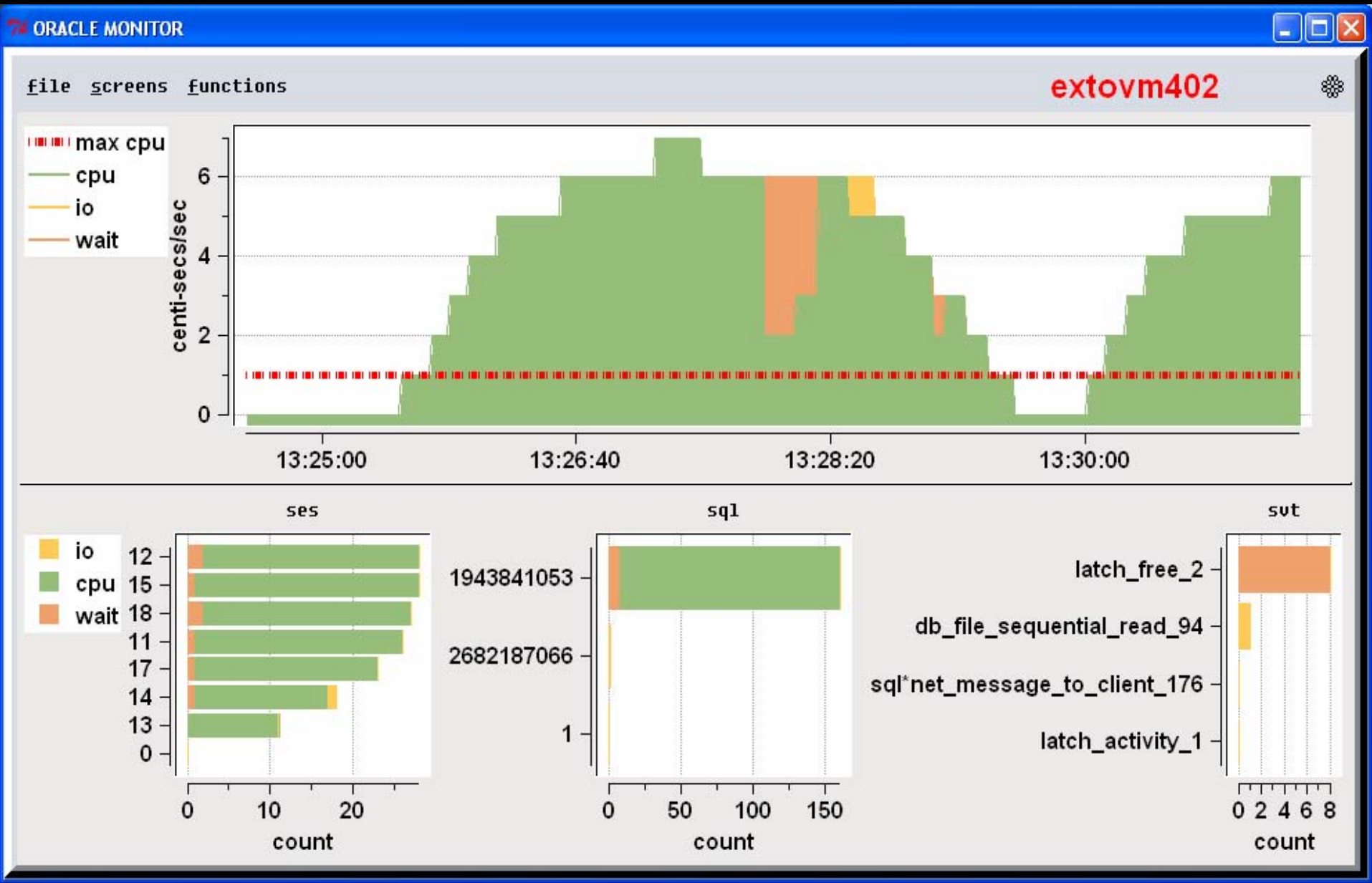


OEM Perf Page

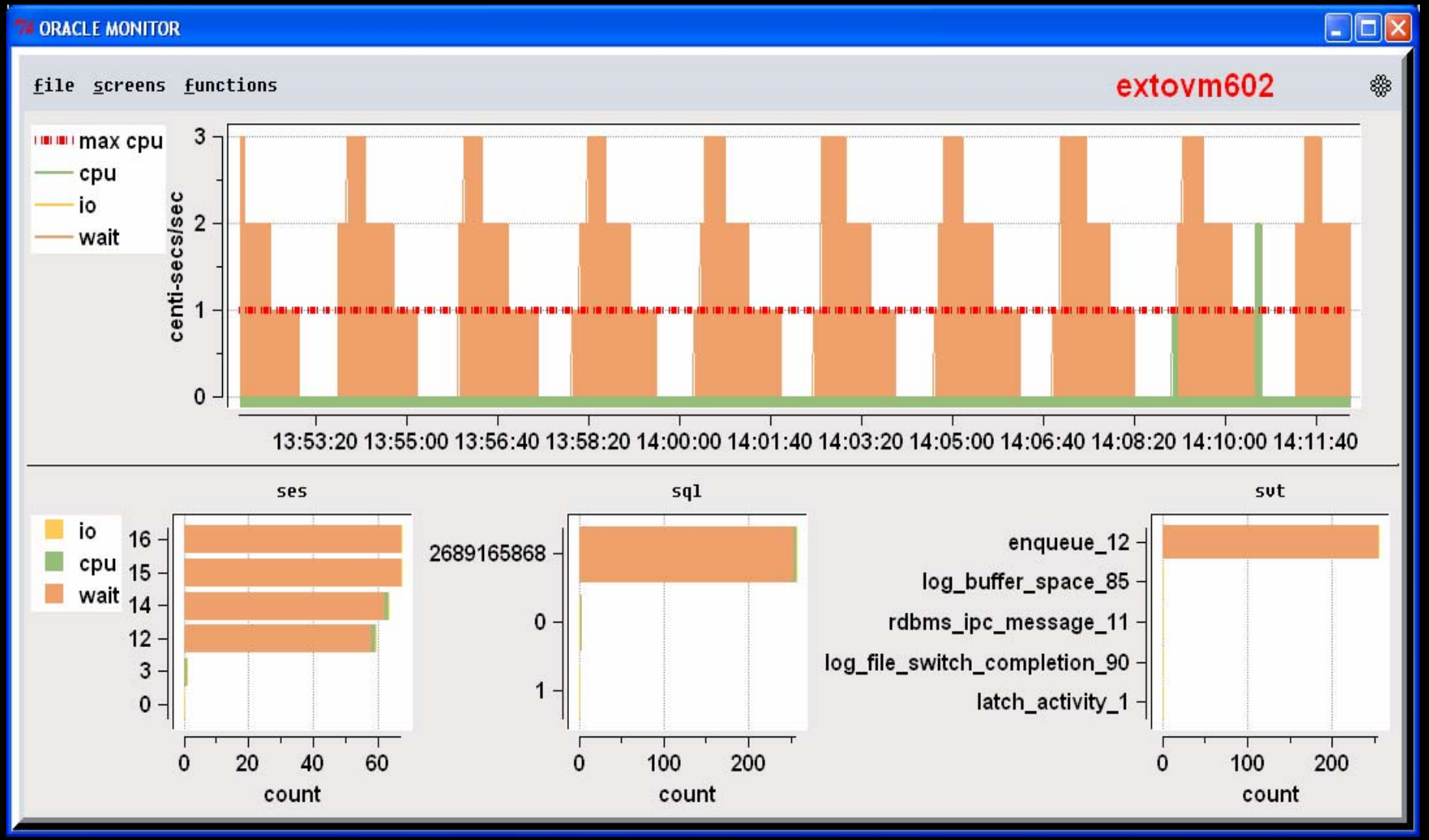




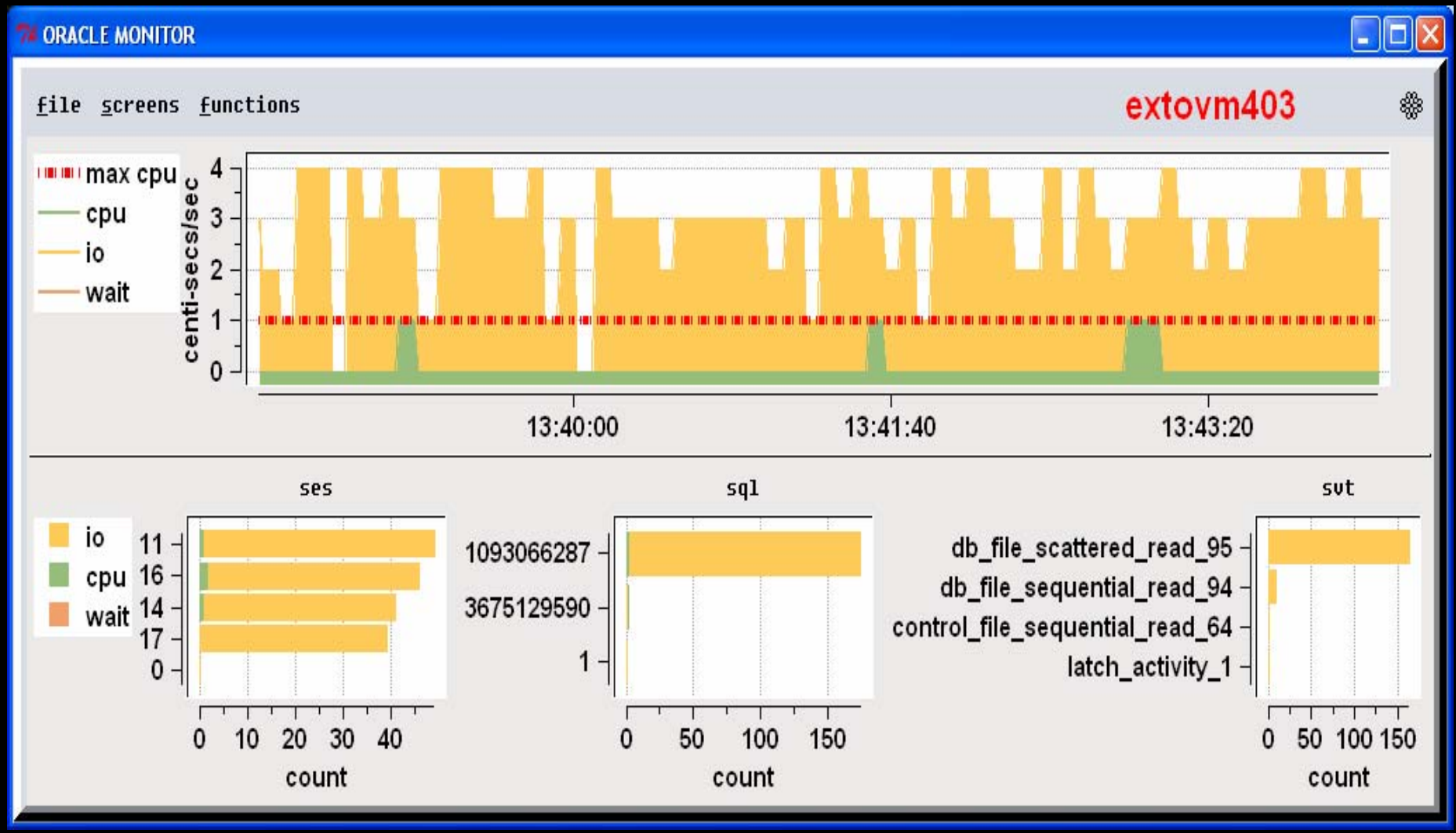
TCL CPU


ORACLE MONITOR
file screens functions
extovm402
max cpu
cpu
io
wait
centi-secs/sec
0
2
4
6
13:25:00
13:26:40
13:28:20
13:30:00
ses
sql
svt
io
cpu
wait
12
15
18
11
17
14
13
0
0
10
20
count
1943841053
2682187066
1
0
50
100
150
count
latch_free_2
db_file_sequential_read_94
sql*net_message_to_client_176
latch_activity_1
0 2 4 6 8
count

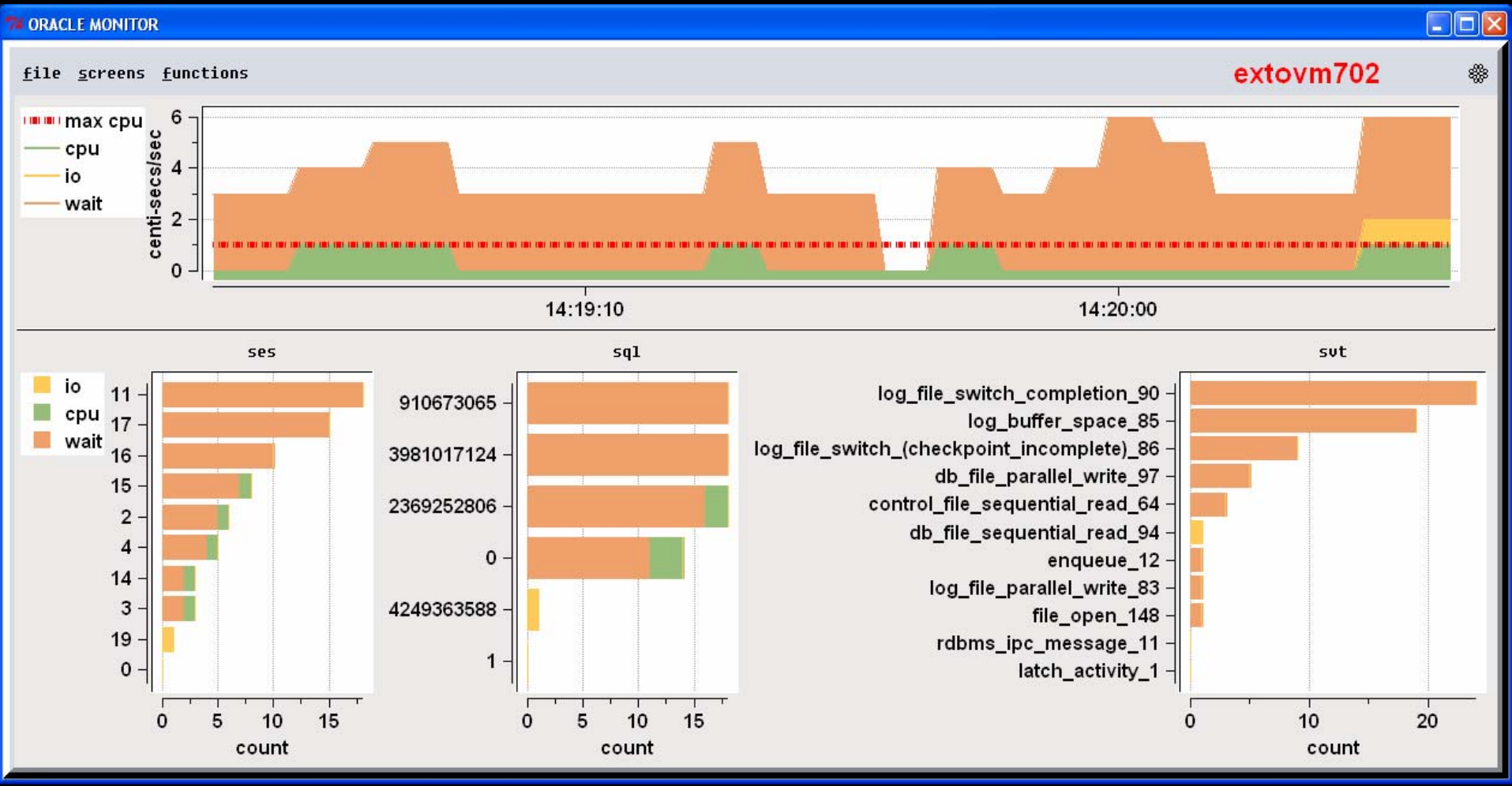
TCL Blocked Users (lock)



IO


ORACLE MONITOR
file screens functions
extovm403
max cpu
cpu
io
wait
centi-secs/sec
0
1
2
3
4
13:40:00
13:41:40
13:43:20
ses
io
cpu
wait
11
16
14
17
0
0 10 20 30 40
count
sql
1093066287
3675129590
1
0 50 100 150
count
svt
db_file_scattered_read_95
db_file_sequential_read_94
control_file_sequential_read_64
latch_activity_1
0 50 100 150
count

Redo Log Problems



Statistics vs Waits

- Cache Buffer Hit vs IO Waits

Why group around in the dark – just turn the lights on

CPU problem

- CPU is only updated at the end of the call.
- Long calls look deceiving like no CPU is being used

CPU in ASH vs Stats

Monitoring Waits can be Expensive

- Rows queried =

  # of sessions x # of waits

  In 10g there are over 800 waits.

  For example 100 user x 800 waits = 80,000 rows

Sampling Cheap

- With PL/SQL it’s less that 1 % CPU with 10 average active sessions
- ASH in 10g even cheaper
- ASH via Memory Scraping , 200x cheaper

ASH is Rich Data

- On top of being Cheap the data is multi-dimensional – you can cut it up in many different ways
  - Top Session
  - Top SQL
  - Top Module
  - Top Action
  - Top Program
  - Top Client
  - Top Service
- All ranked by CPU, IO, or any of 800 waits or time spent in wait, or by file accessed, or object accessed

Alert on Blocked Sessions

- Any session that is on a wait 15/15 samples can be called blocked whether or not they are on a row lock

ASH easily enables Drilldown Investigation

- See a spike in System Load (avg active sessions)
- Find out which SQL
- Find out what waits that SQL blocked on

Simulating ASH

- ASH is new in 10g
- Need Diagnostic Pack License ☹
- ASH data exist since V7
- Simulate it yourself ☺
  - Join v$session_wait & v$session
  - Save in a table

Consumes < 1% CPU for 10 active sessions (a lot)

Description of v$session_wait

SID	NUMBER	Session
SEQ#	NUMBER	
WAIT_TIME	NUMBER	State
EVENT	VARCHAR2(64)	Wait
P1TEXT	VARCHAR2(64)	
P1	NUMBER	
P1RAW	RAW(4)	
P2TEXT	VARCHAR2(64)	
P2	NUMBER	
P2RAW	RAW(4)	
P3TEXT	VARCHAR2(64)	
P3	NUMBER	
P3RAW	RAW(4)	
WAIT_TIME	NUMBER	
SECONDS_IN_WAIT	NUMBER	
STATE	VARCHAR2(19)	

Description of v$active_session_history

SAMPLE_ID	Calculated	NUMBER	Time
SAMPLE_TIME		TIMESTAMP(3)	
SESSION_ID	v$session	NUMBER	Session
SESSION_SERIAL#		NUMBER	
USER_ID		NUMBER	
SERVICE_HASH		NUMBER	
SESSION_TYPE		VARCHAR2(10)	
QC_SESSION_ID		NUMBER	
QC_INSTANCE_ID		NUMBER	
PROGRAM		VARCHAR2(64)	
MODULE		VARCHAR2(48)	
ACTION		VARCHAR2(32)	
CLIENT_ID		VARCHAR2(64)	
SESSION_STATE		VARCHAR2(7)	State
EVENT		VARCHAR2(64)	Wait
EVENT_ID		NUMBER	
EVENT#		NUMBER	
SEQ#		NUMBER	
P1		NUMBER	
P2		NUMBER	
P3		NUMBER	
WAIT_TIME		NUMBER	
TIME_WAITED		NUMBER	
CURRENT_OBJ#		NUMBER	
CURRENT_FILE#		NUMBER	
CURRENT_BLOCK#		NUMBER	
SQL_ID	v$session	VARCHAR2(13)	SQL
SQL_CHILD_NUMBER		NUMBER	
SQL_PLAN_HASH_VALUE		NUMBER	
SQL_OPCODE		NUMBER	

Create a Package and Insert

- Create Package
  - Query v$session & v$session_wait joined
  - Sample 1 a second
  - Collect into a GTT
  - Insert into local or remote every 15 seconds
  - voila

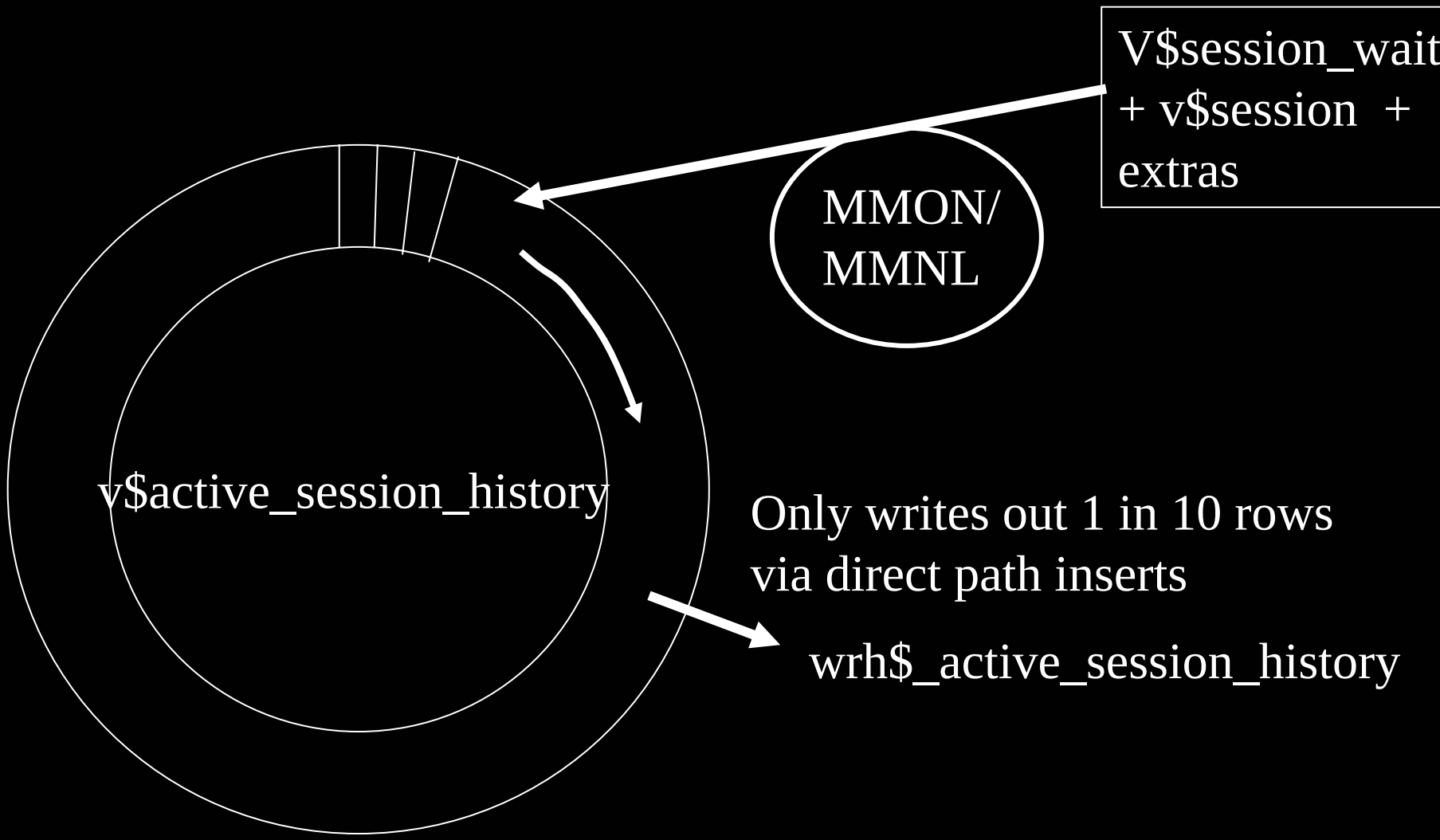
Create ASH Storage Table

```
drop table ash_data;
create table ash_data (
          target_id       number,
          sample_id       number,
          sample_time     date,
          sid             number,
          state           varchar2(20),
          serial#         number,
          user#           number,
          sql_address#    varchar2(20),
          sql_hash        number,
          command         number,
          session_type    number,
          event#          number,
          seq#            number,
          p1              number,
          p2              number,
          p3              number,
          wait_time       number,
          row_wait_obj#   number,
          row_wait_file#  number,
          row_wait_block# number,
          program         varchar2(64),
          module_hash     number,
          action_hash     number
);
```

Simulation Optimizations

- Partition ASH_DATA for efficient deletion of old data
- Run compaction routines to save history
  - Load
  - Top Session
  - Top sql
  - Top waits

How ASH works



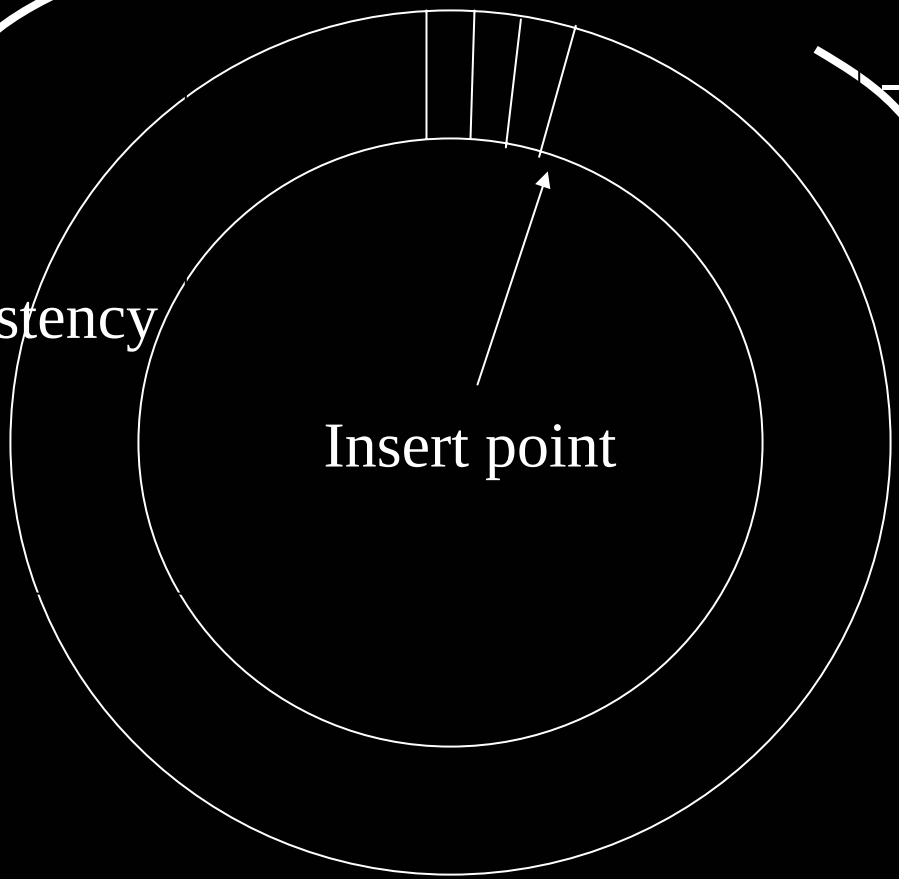
ASH buffer

Select reads backwards

-No latching

-No read consistency

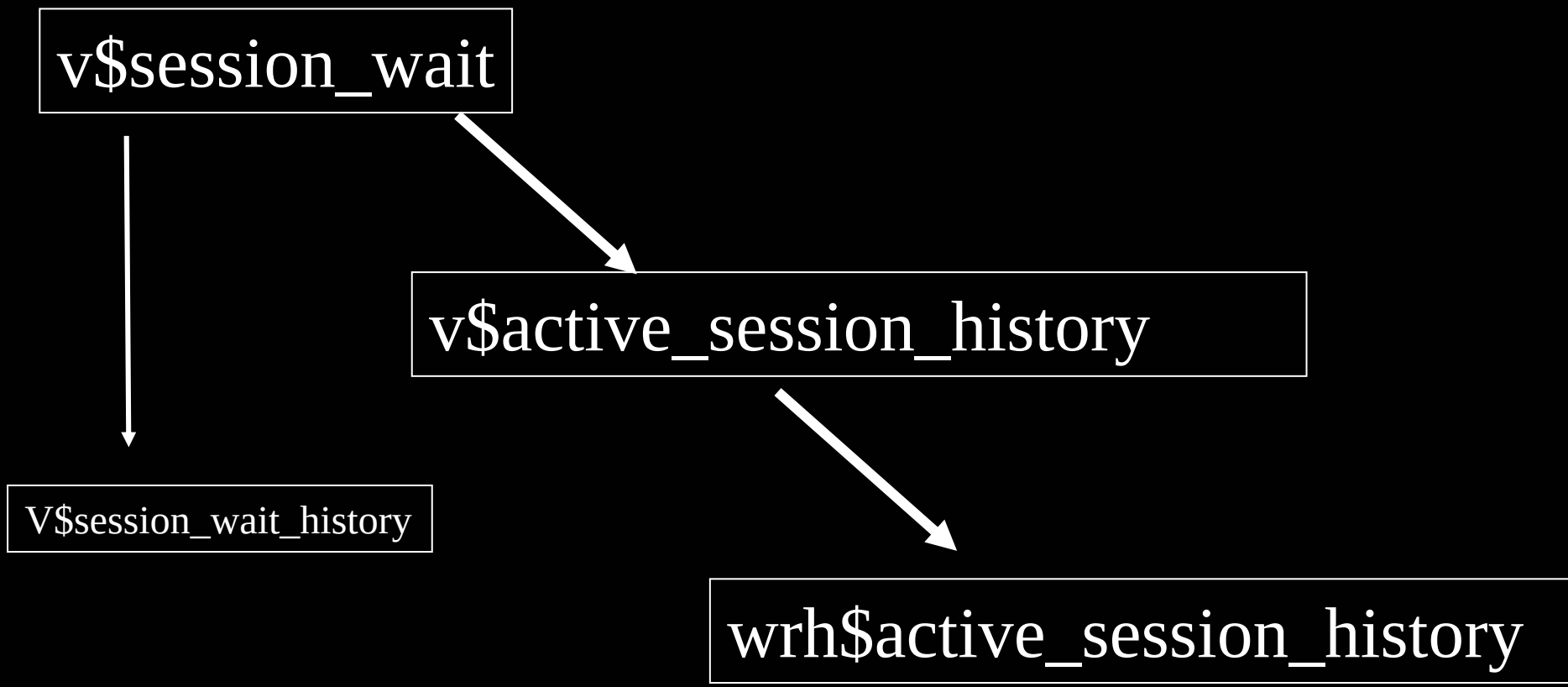
-Index on time



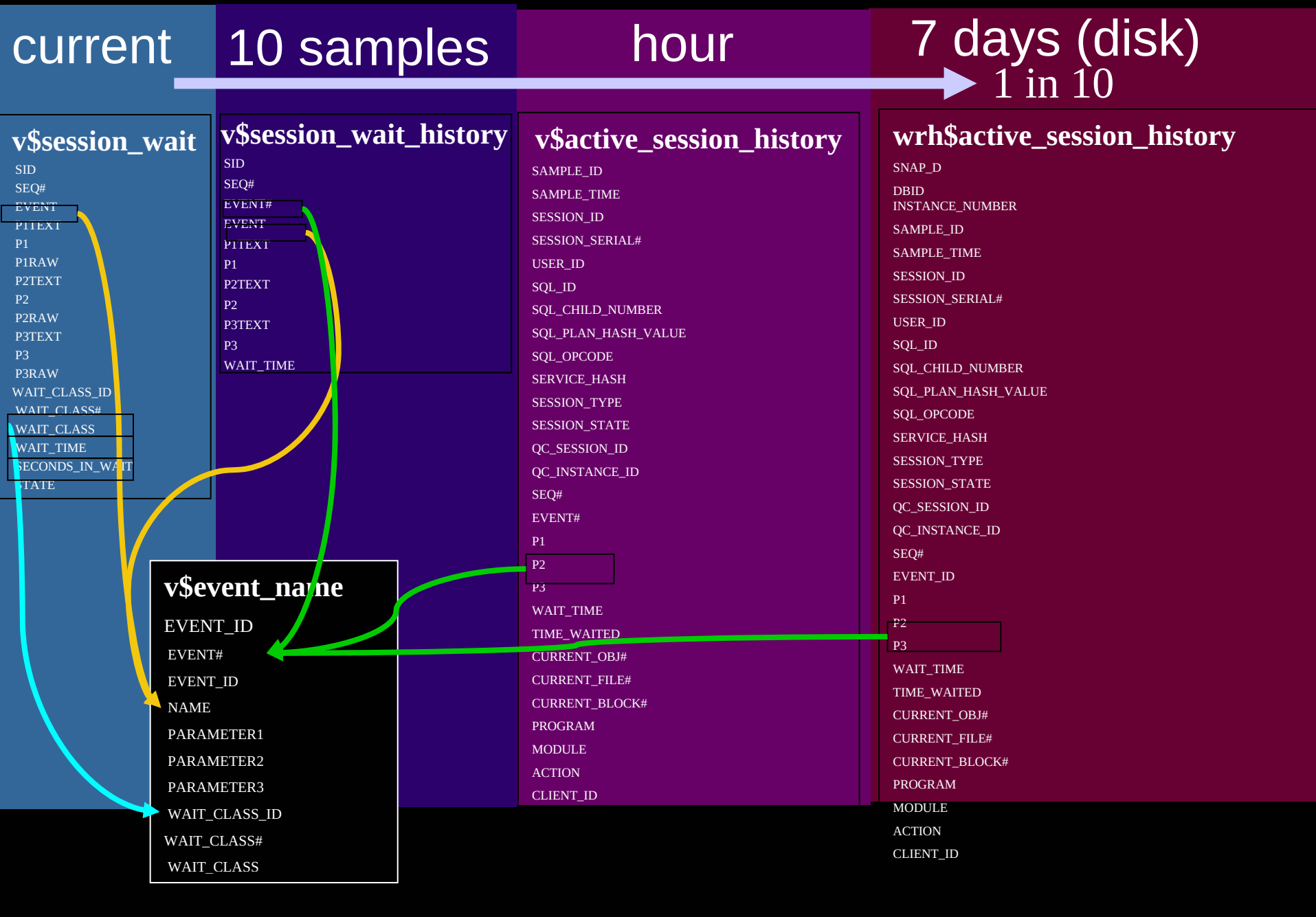
Insert one direction

- Touch up wait times

Family of ASH Tables



ASH Tables


current
10 samples
hour
7 days (disk)
1 in 10
v$session_wait
SID
SEQ#
EVENT
P1TEXT
P1
P1RAW
P2TEXT
P2
P2RAW
P3TEXT
P3
P3RAW
WAIT_CLASS_ID
WAIT_CLASS#
WAIT_CLASS
WAIT_TIME
SECONDS_IN_WAIT
STATE
v$session_wait_history
SID
SEQ#
EVENT#
EVENT
P1TEXT
P1
P2TEXT
P2
P3TEXT
P3
WAIT_TIME
v$active_session_history
SAMPLE_ID
SAMPLE_TIME
SESSION_ID
SESSION_SERIAL#
USER_ID
SQL_ID
SQL_CHILD_NUMBER
SQL_PLAN_HASH_VALUE
SQL_OPCODE
SERVICE_HASH
SESSION_TYPE
SESSION_STATE
QC_SESSION_ID
QC_INSTANCE_ID
SEQ#
EVENT#
P1
P2
P3
WAIT_TIME
TIME_WAITED
CURRENT_OBJ#
CURRENT_FILE#
CURRENT_BLOCK#
PROGRAM
MODULE
ACTION
CLIENT_ID
wrh$active_session_history
SNAP_D
DBID
INSTANCE_NUMBER
SAMPLE_ID
SAMPLE_TIME
SESSION_ID
SESSION_SERIAL#
USER_ID
SQL_ID
SQL_CHILD_NUMBER
SQL_PLAN_HASH_VALUE
SQL_OPCODE
SERVICE_HASH
SESSION_TYPE
SESSION_STATE
QC_SESSION_ID
QC_INSTANCE_ID
SEQ#
EVENT_ID
P1
P2
P3
WAIT_TIME
TIME_WAITED
CURRENT_OBJ#
CURRENT_FILE#
CURRENT_BLOCK#
PROGRAM
MODULE
ACTION
CLIENT_ID
v$event_name
EVENT_ID
EVENT#
EVENT_ID
NAME
PARAMETER1
PARAMETER2
PARAMETER3
WAIT_CLASS_ID
WAIT_CLASS#
WAIT_CLASS

Wait Time vs Time Waited

- SESSION_STATE
  - Waiting, on CPU
  - Based on WAIT_TIME
- WAIT_TIME
  - 0 => waiting
  - >0 => CPU (value is time of last wait)
- TIME_WAITED
  - Actual time waited for event
  - 0 until wait finishes
  - Fix up values (no one else can do this)

Oradebug

- Dump to trace file

```
SQL> oradebug dump ash 5

SQL> Alter session set events ‘immediate
     tracename ashdump level 5’;
```

level 5 = # of minutes

loader file rdbms/demo/ashldr.ctl

INIT.ORA

ASH

statistics_level = Typical (default)

_active_session_history =TRUE (default)

_ash_sampling_interval = 1000 (default, milliseconds)

_ash_enable = false; (A dynamic parameter will turn off ASH sampling, flushing and the V$ views on ASH)

Client Id

- Setting Client ID

```
dbms_session.set_identifier
            (client_id);
```

- Enabling trace for a client ID

```
dbms_monitor.client_id_trace_enable
            (client_id, TRUE, FALSE);
```

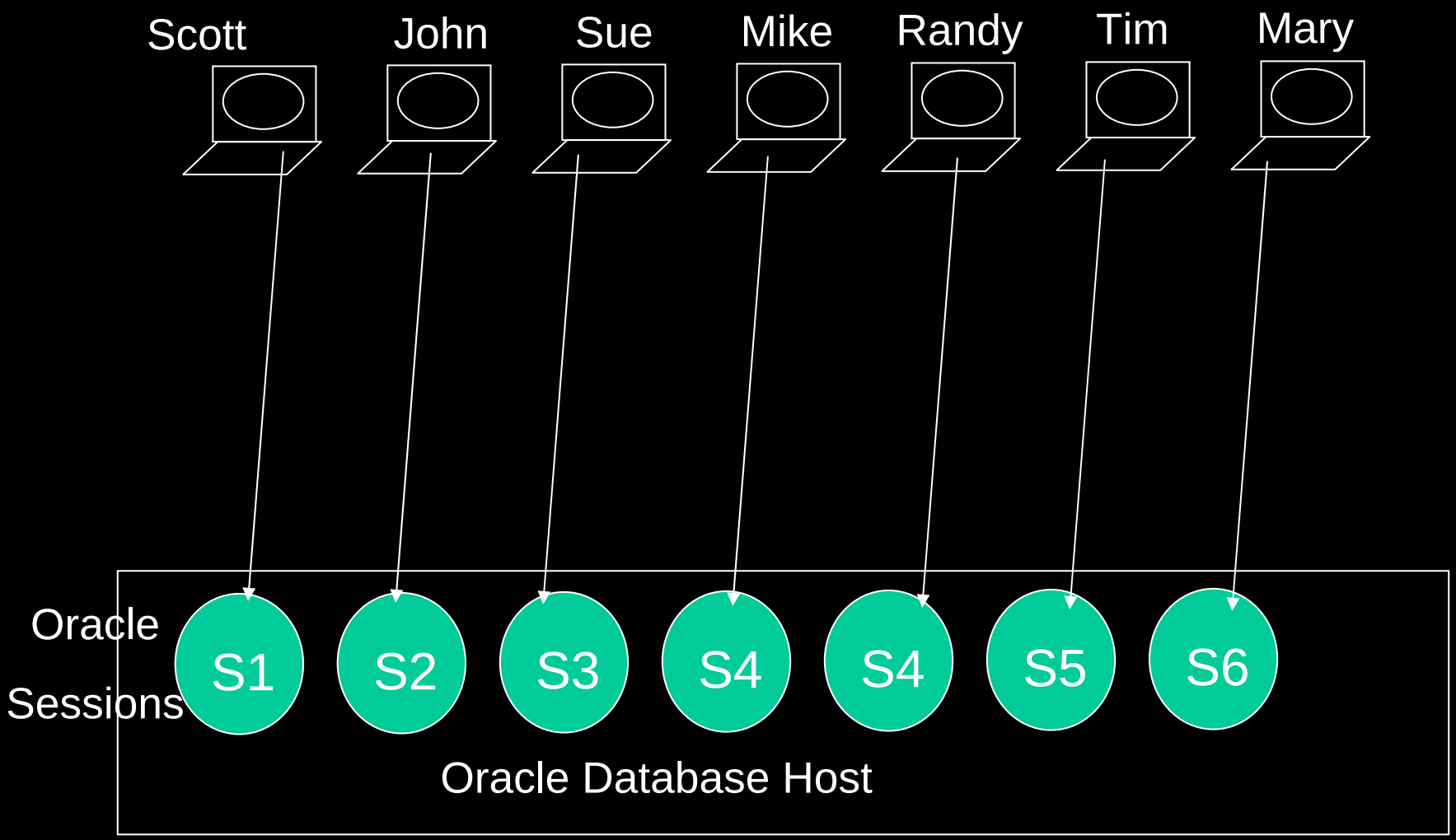
- Enabling statistics aggregation by client id

```
dbms_monitor.client_id_stat_enable
            (client_id);
```

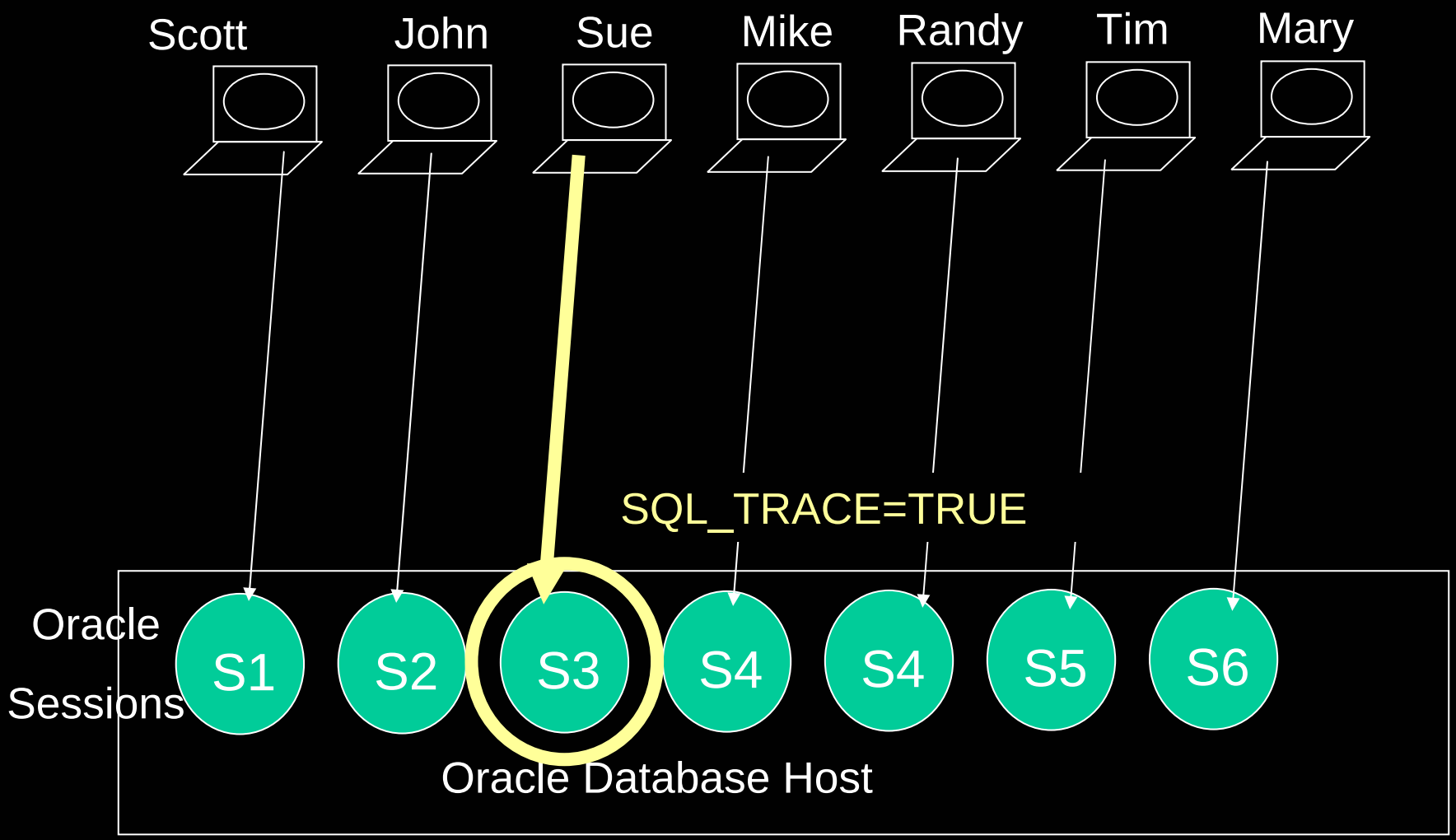
- Script to Extract Client Trace

```
trcsess
```

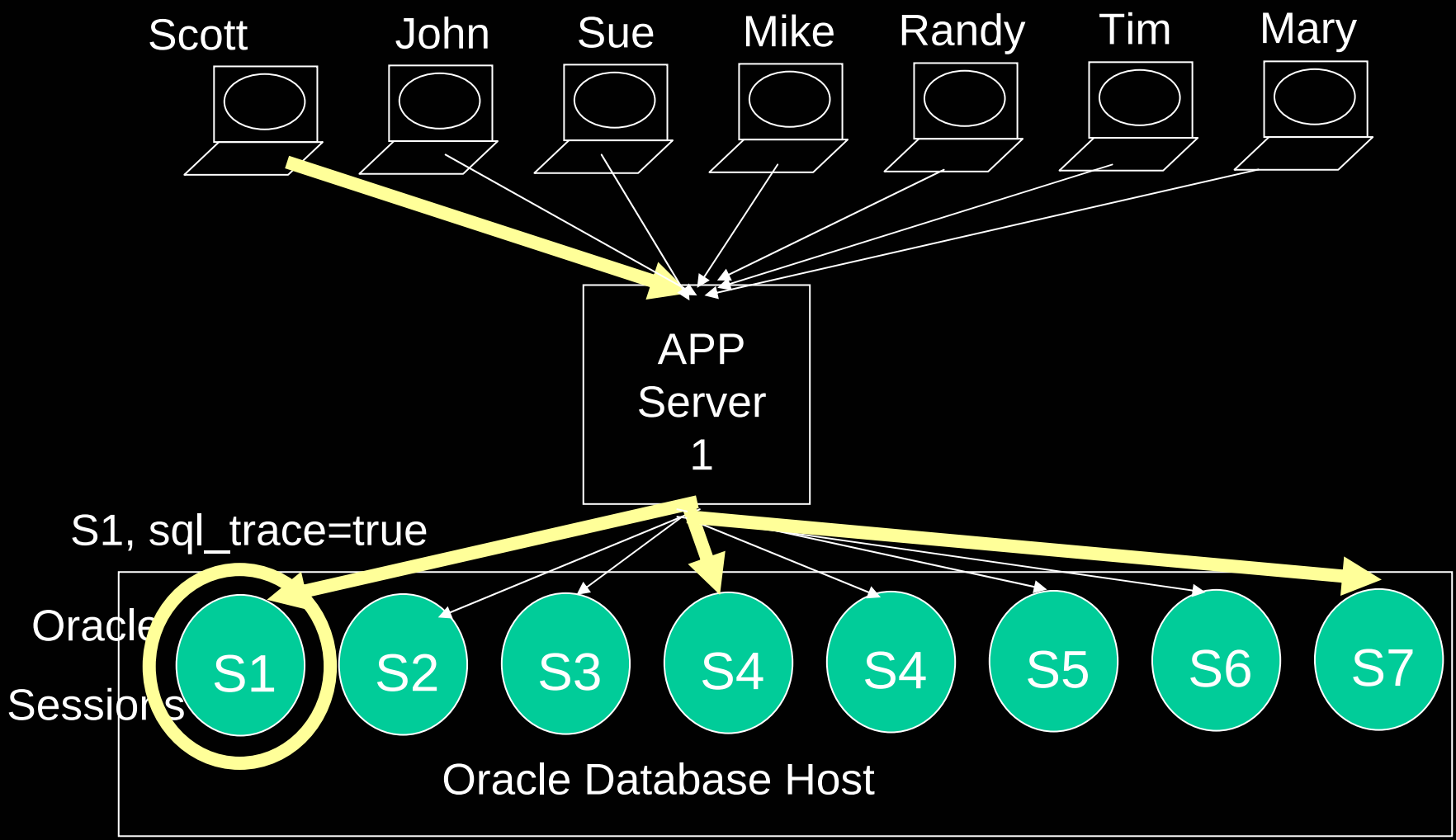
Session Dedicated



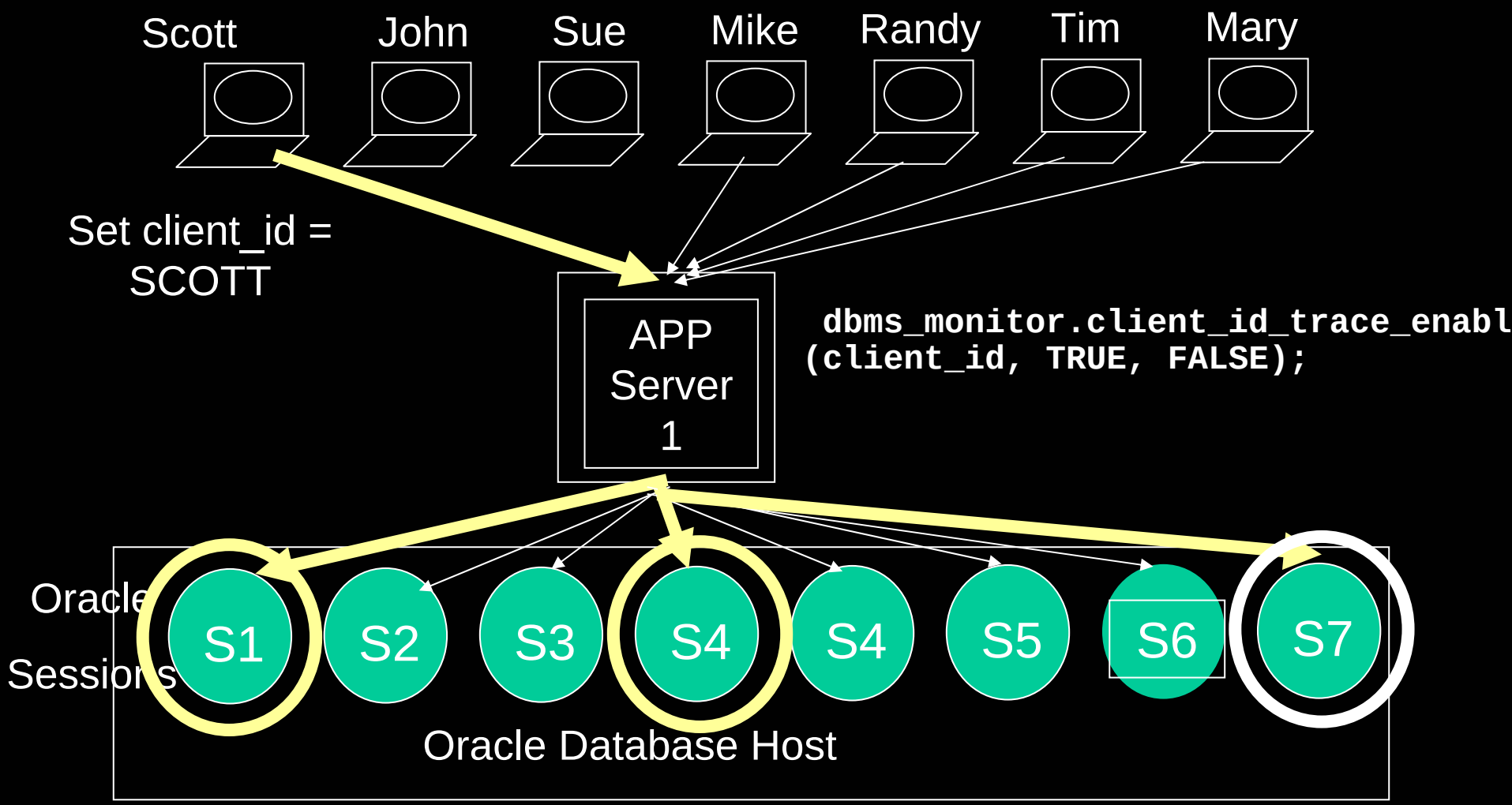
Session Dedicated trace



Session Pooling trace



Session Pooling



Extras

- 10gR2 added fields to ASH
  - Blocking Session Id, serial# and state
  - XID, transaction ID
  - RAC event Fixup
  - Plan Hash Fix up
    - 10gR1 - during parsing no plan, good way to find parsing problems
    - 10gR2 – get plan hash fixup – good but loose some ability to find parsing problems
  - Wait Class, needed this for grouping

How Many Active Sessions?

- 1 CPU means max 1 Avg Active Session unless there is a bottleneck

Active Sessions at Big Sites

- Oracle 4 way RAC internal apps
  - 10,000 connected, 200 active
- One Site
  - 3000 connected, 30 Active
- Site
  - 12,000 connected, 100 active

ASH – OS and other DBs

- SQL Server
- Linux
- DB2
- Sybase

The Challenger Incident

Launch: Pressure

Midnight before
January 28, 1986
Lives are on the line

List of Past Problems

HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

	SRM No.	Cross Sectional View: Erosion Depth (in.)	Cross Sectional View: Perimeter Affected (deg)	Cross Sectional View: Nominal Dia. (in.)	Top View: Length Of Max Erosion (in.)	Top View: Total Heat Affected Length (in.)	Clocking Location (deg)
61A LH Center Field**	22A	None	None	0.280	None	None	36°--66°
61A LH ~~CENTER~~ FIELD**	22A	NONE	NONE	0.280	NONE	NONE	338°-18°
51C LH Forward Field**	15A	0.010	154.0	0.280	4.25	5.25	163
51C RH Center Field (prim)***	15B	0.038	130.0	0.280	12.50	58.75	354
51C RH Center Field (sec)***	15B	None	45.0	0.280	None	29.50	354
41D RH Forward Field	13B	0.028	110.0	0.280	3.00	None	275
41C LH Aft Field*	11A	None	None	0.280	None	None	--
41B LH Forward Field	10A	0.040	217.0	0.280	3.00	14.50	351
STS-2 RH Aft Field	2B	0.053	116.0	0.280	--	--	90

*Hot gas path detected in putty. Indication of heat on O-ring, but no damage.
**Soot behind primary O-ring.
***Soot behind primary O-ring, heat affected secondary O-ring.

Clocking location of leak check port - 0 deg.

OTHER SRM-15 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY AND NO SOOT NEAR OR BEYOND THE PRIMARY O-RING.

SRM-22 FORWARD FIELD JOINT HAD PUTTY PATH TO PRIMARY O-RING, BUT NO O-RING EROSION AND NO SOOT BLOWBY. OTHER SRM-22 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY.

Correlated with Temperatures

BLOW BY HISTORY

SRM-15 WORST BLOW-BY

- 2 CASE JOINTS (80°), (110°) ARC
- MUCH WORSE VISUALLY THAN SRM-22

SRM 22 BLOW-BY

- 2 CASE JOINTS (30-40°)

SRM-13A, 15, 16A, 18, 23A 24A

- NOZZLE BLOW-BY

HISTORY OF O-RING TEMPERATU
(DEGREES-F)

MOTOR	MBT	AMB	O-RING	WIN
DM-1	68	36	47	10 m
DM-2	76	45	52	10 MP
QM-3	72.5	40	48	10 m
QM-4	76	48	51	10 m
SRM-15	52	64	53	10 m
SRM-22	77	78	75	10 m
SRM-25	55	26	29 27	10 m 25 m

Trying to Show data an Analysis

CONCLUSIONS :

- TEMPERATURE OF O-RING IS NOT ONLY PARAMETER CONTROLLING BLOW-BY

  SRM 15 WITH BLOW-BY HAD AN O-RING TEMP AT 53°F
  SRM 22 WITH BLOW-BY HAD AN O-RING TEMP AT 75°F
  FOUR DEVELOPMENT MOTORS WITH NO BLOW BY WERE TESTED AT O-RING TEMP OF 47° TO 52 °F

  DEVELOPMENT MOTORS HAD PUTTY PACKING WHICH RESULTED IN BETTER PERFORMANCE

- AT ABOUT 50°F BLOW-BY COULD BE EXPERIENCED IN CASE JOINTS

- TEMP FOR SRM 25 ON 1-28-86 LAUNCH WILL BE 29°F 9 AM
  38°F 2 PM

- HAVE NO DATA THAT WOULD INDICATE SRM 25 IS DIFFERENT THAN SRM 15 OTHER THAN TEMP

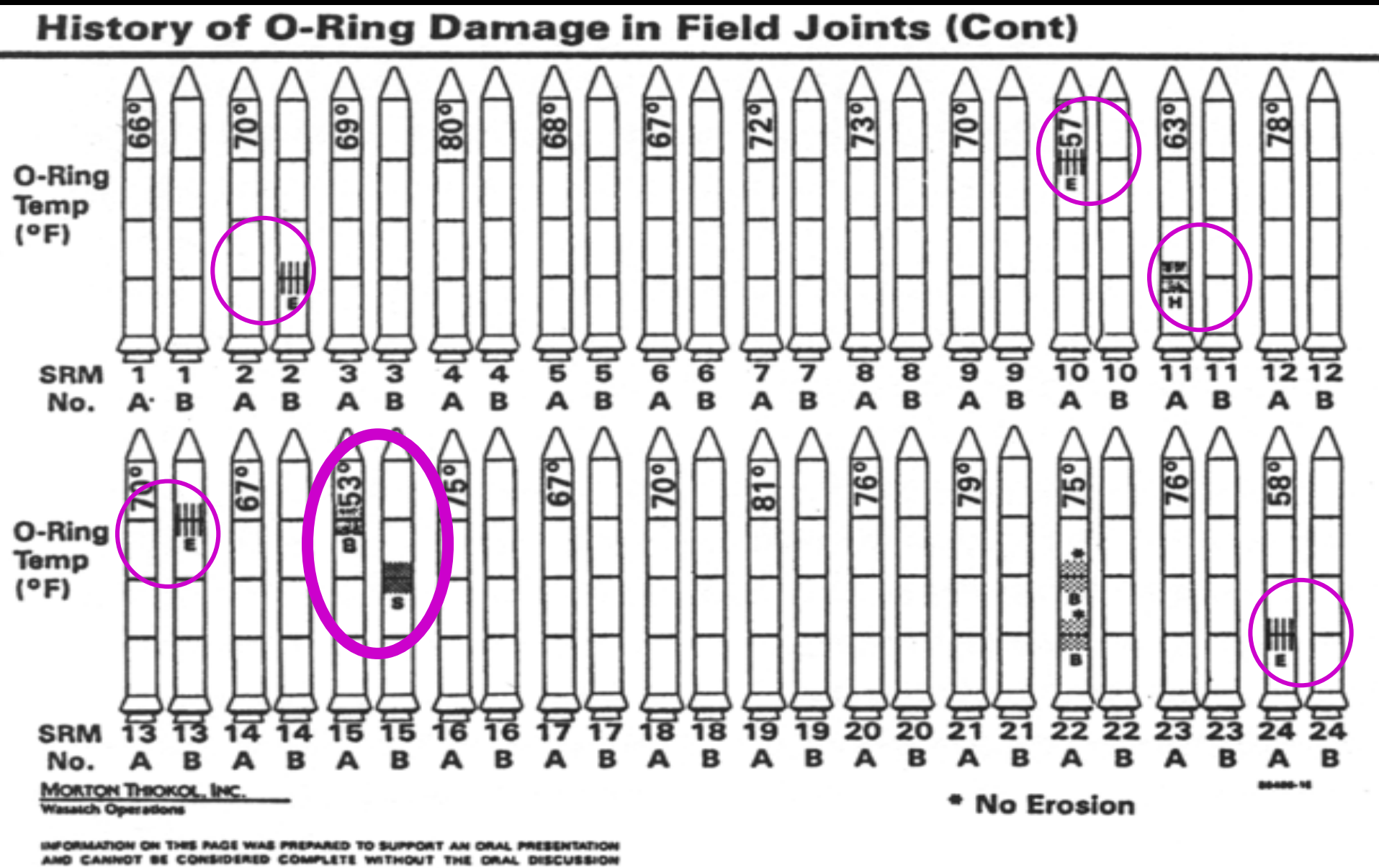
RECOMMENDATIONS :

- O-RING TEMP MUST BE $\geq$ 53 °F AT LAUNCH

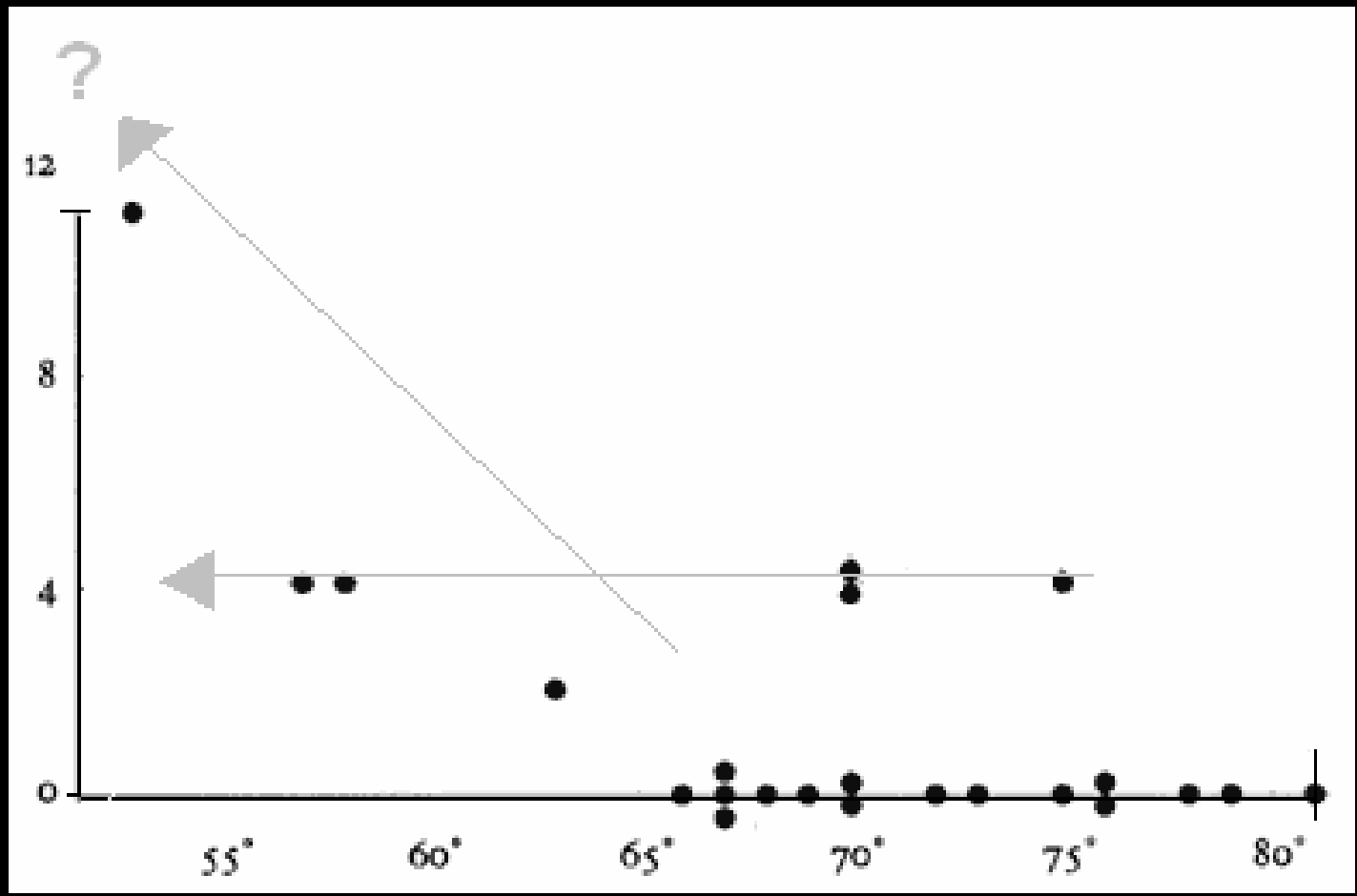
  DEVELOPMENT MOTORS AT 47° TO 52 °F WITH PUTTY PACKING HAD NO BLOW-BY
  SRM 15 (THE BEST SIMULATION) WORKED AT 53 °F

- PROJECT AMBIENT CONDITIONS (TEMP & WIND) TO DETERMINE LAUNCH TIME

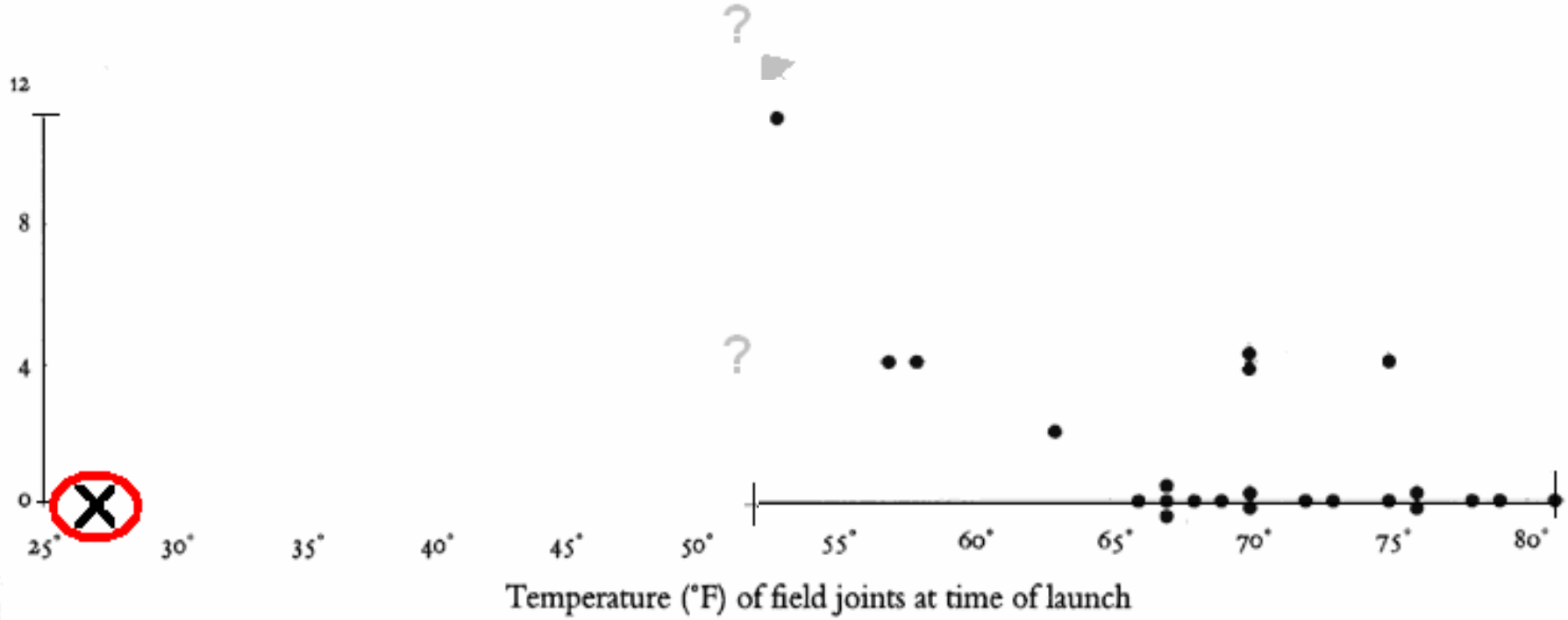
Congressional Hearings Evidence



Engineers Tried to Communicate



Even Clearer



Source: Edward Tufte, *Visual Explanations*

1986 Space Shuttle Challenger -- O-Ring Chart

O-ring damage index, each launch. By Edward Tufte.

Difficult

- NASA Engineers Fail
- Congressional Investigators Fail
- Data is Difficult

But …

Lack of Clarity can be devastating

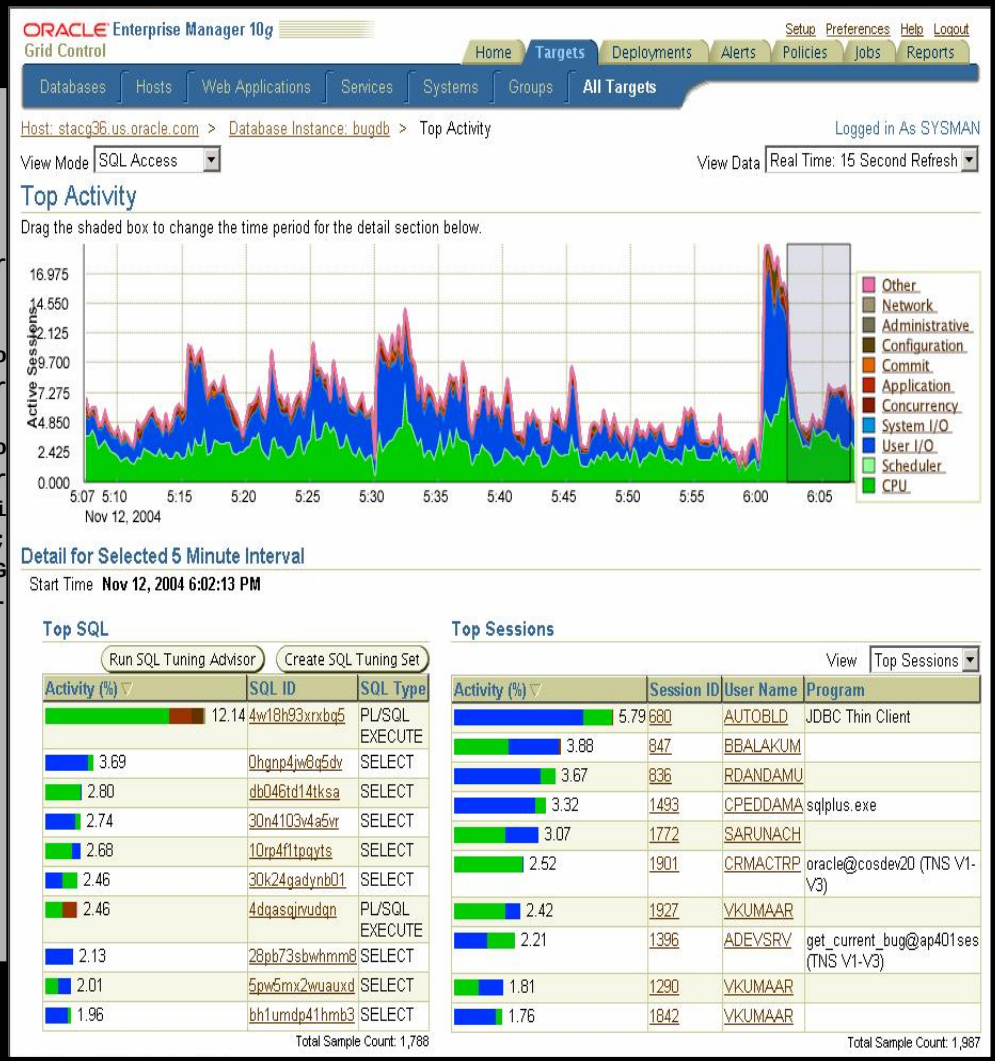
Imagine Trying to Drive your Car if you Dashboard looked like:

```
SVRMGR>
SVRMGR> set charwidth 12
Charwidth                      12
SVRMGR> set numwidth 10
Numwidth                       10
SVRMGR> Rem Select Library cache statistics.  The pin hit rate shoule be high.
SVRMGR> select namespace library,
     2>        gets,
     3>        round(decode(gethits,0,1,gethits)/decode(gets,0,1,gets),3)
     4>           gethitratio,
     5>        pins,
     6>        round(decode(pinhits,0,1,pinhits)/decode(pins,0,1,pins),3)
     7>           pinhitratio,
     8>        reloads, invalidations
     9>   from stats$lib;
LIBRARY      GETS       GETHITRATI PINS       PINHITRATI RELOADS    INVALIDATI
------------ ---------- ---------- ---------- ---------- ---------- ----------
BODY                 40        .95         40         .4          0          0
CLUSTER               0          1          0          1          0          0
INDEX                 0          1          0          1          0          0
OBJECT                0          1          0          1          0          0
PIPE                  0          1          0          1          0          0
SQL AREA            835       .938      82110       .999          0          0
TABLE/PROCED        117       .778        485        .94          0          0
TRIGGER               0          1          0          1          0          0
8 rows selected.
SVRMGR>
```

And is updated once and hour

Or This?

```
SVRMGR>
SVRMGR> set charwidth 12
Charwidth
SVRMGR> set numwidth 10
Numwidth
SVRMGR> Rem Select Librar
SVRMGR> select namespace
     2>        gets,
     3>        round(deco
     4>           gethitr
     5>        pins,
     6>        round(deco
     7>           pinhitr
     8>        reloads, i
     9>   from stats$lib;
LIBRARY      GETS       G
------------ ---------- -
BODY                40
CLUSTER              0
INDEX                0
OBJECT               0
PIPE                 0
SQL AREA           835
TABLE/PROCED       117
TRIGGER              0
8 rows selected.
SVRMGR>
```



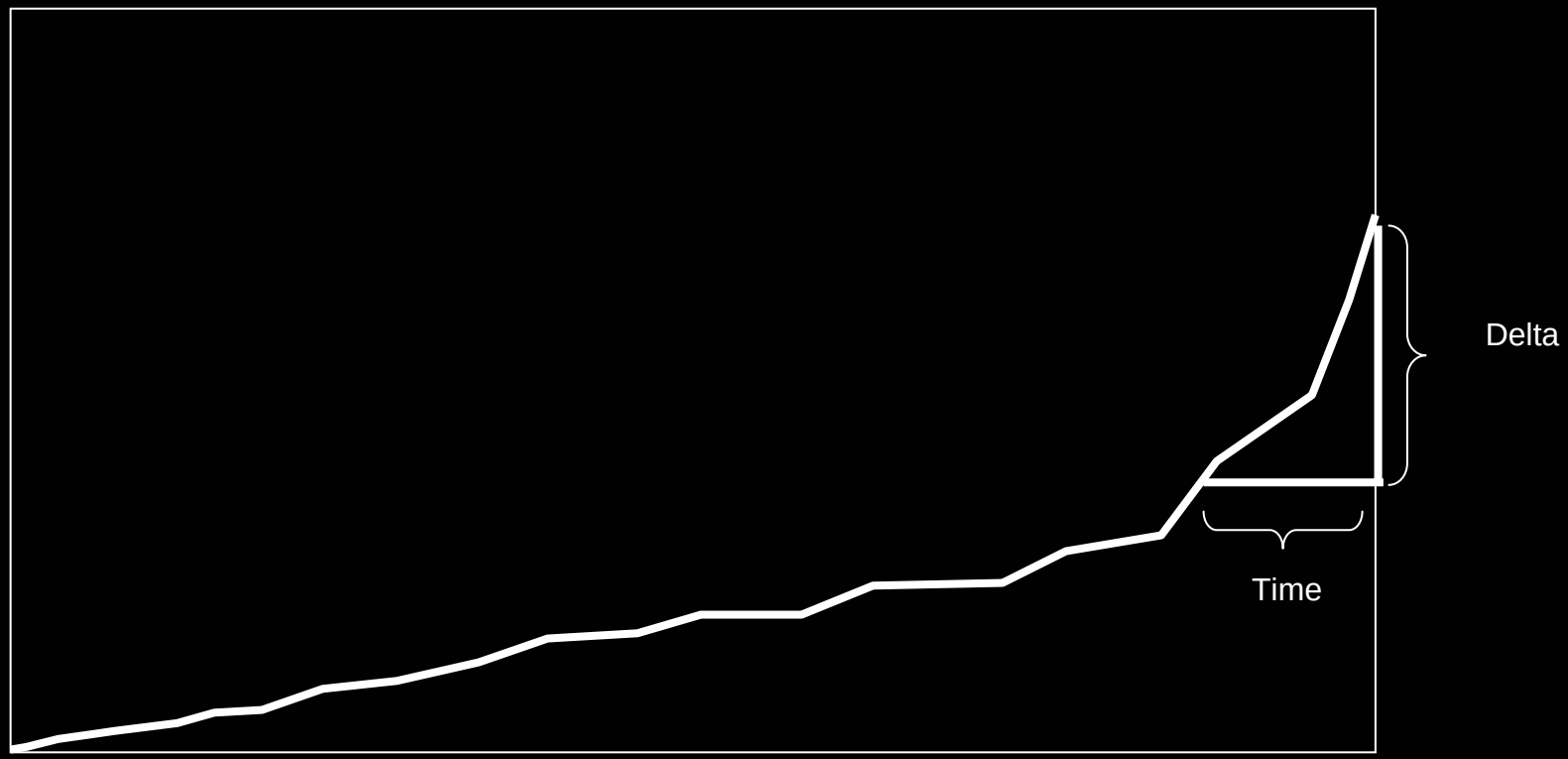
3 Types of Collecting

- Cumulative
- Current
- Sampling Current

Cumulative

- Cumulative Counters
  - v$sysstat
  - v$system_event
  - Electric meter

Cumulative Need Deltas



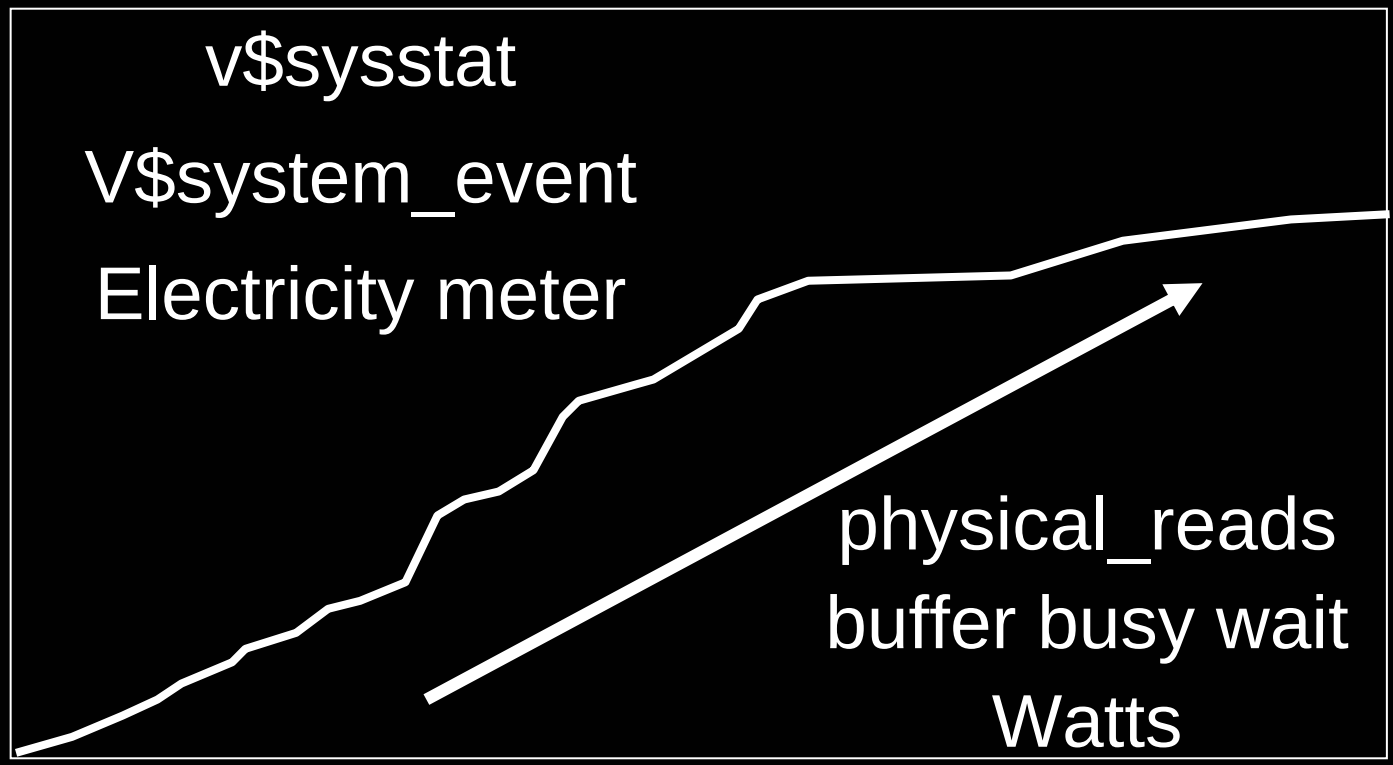
Delta = value_time_B – value_time_A

Rate = Delta/time

Counters and Rates

- Statistics - good
  - Transactions/sec
  - Commits/sec
  - Watts/month
- Waits – weird, either
  - Delta = Seconds spend over time period
  - Rate = Centi-secs/sec

Statistics & Waits are Counters



Statistics just keep growing

Current

- Current
  - v$session_wait
  - Current Stats (logons current,opened cursors current)
  - temperature

Sampling

- Sampling
  - ASH
  - Sunny days a year per city