



Comprehensive Approach to Performance Diagnosis

Daniel Fink
OptimalDBA.com



- Senior Oracle Database Performance Engineer
 - Main focus on Diagnosis and Optimization
 - Data Recovery and Training
- 11 years Oracle database administration
- Member of the Oak Table Network

daniel.fink@optimaldba.com
www.optimaldba.com



Copyright 2007 OptimalDBA.com OptimalDBA.com 2

Goal

- Establish a foundation for effective performance diagnosis using multiple tools/approaches.

“Winners know why they win”

Dave Ensor

Agenda

- Overview & Foundation
- Statistics
- Timed Events
- Extended SQL Trace
- Statspack

Overview

- Brief History of Performance Diagnosis
- Timing/Counts Details

Brief History of Performance Diagnosis

- *Guess*
- bstat/estat
- Wait Interface
- Statspack
- AWR/ASH

Guess

- Solutions based on reason, past experience, hunches
 - Not always inaccurate
 - Rarely repeatable
- Still in use today
 - AKA Silver Bullets
 - Items gleaned from Tips and Tricks

bstat/estat

- Introduced in v6
- Still shipped with 10gR2
 - *and in 11g!
- Gathers some info, but far from complete
 - Stats/Timed Events
 - No SQL info
 - Better than nothing!

Wait Interface

- More accurately, Timed Event Interface
- Events gathered at session/system level
 - Not all events are instrumented
- Session History introduced in 10g
 - Last 10 events

Statspack

- Introduced in 8i
 - Can be altered to be compatible with 7
- Automated gathering of information
- No session level information

AWR/ASH

- New in 10g
- I won't cover it today
 - Must be licensed separately
 - Even though it is running by default
 - Low rate of adoption

Counts

- Number of times an event occurs does not equal direct impact on performance
- However, you may be able to reduce response time by reducing the number of times the event occurs

Timing

- It is critical to know the units of measurement. Are they in seconds, centiseconds, milliseconds or microseconds?

□ 1 second = 100 centiseconds
 1000 milliseconds
 1000000 microseconds

□ 1 centisecond = 10 milliseconds
 10000 microseconds

Business Perspective

- Did the statement/process perform the correct actions (return the correct data, update the date correctly, etc.)?
- Did the statement/process perform the correct actions in an acceptable amount of time?

What is Performance?

- User perception of task time
- Benchmarked task time
- Not
 - artificial metrics
 - dashboards
 - ratios

Poor Performance

- User perception
 - Acceptable time != actual time
- Deviation from patterns

Causes of Poor Performance

- Resource Consumption
- Resource Contention
- User Perception

Resource Consumption

- Resource is Finite
 - Only 1 process can use 1 cycle of 1 CPU at a time
 - A disk can read only one block at a time
- Optimized Resource
 - A properly optimized system uses only as much resource as it needs and it uses all the resource it needs.

Resource Contention

- If a process is using a resource, others may not be able to use it
- The less resource you use, the less likely you will contend with someone else for the same resource

User Perception

- User may complain about performance
 - Not as fast as they think it should be
 - Compare with other functions that seem similar
- Ask user to assist
 - Often, this alone will improve user perception of performance

Response Time

- $ResponseTime = ServiceTime + WaitTime$
- Service Time
 - Time performing an action
 - From the session level, this means CPU activity
 - From a user level, it means work that they do
- Wait Time
 - Time spent waiting for another system/sub system
 - Not all code is instrumented

Session States

- Performing CPU activity
- Waiting on available CPU cycles
- Waiting on the completion of Oracle instrumented wait/timed event
- Waiting on the completion of Oracle uninstrumented wait/timed event

Know correlations

- CPU correlates LIO
 - Memory access requires CPU cycles
 - It could also mean
 - Sorting
 - Calculations/Manipulations
- PIO means LIO failed
 - Too much LIO
 - Another session is 'flushing' the cache

Gather Information While the Problem is Occurring

- Don't try to diagnose a problem that occurred a week ago
- Involve users
- Know what has changed

Hang, Loop or Slow?

- Hang - session is waiting on a condition or event that will never happen
- Loop - session is repeating an operation infinitely
- Slow - session is working, but performance is unacceptable

Performance Diagnosis Process

- Stage 1
 - What
 - Which
 - Who
 - Where
 - When
- Stage 2
 - Why
 - How

WHAT are the symptoms

- Describe the symptoms and only the symptoms
- Always start with the symptom
 - In early stages of diagnosis, defining the problem can cause other factors to be ignored
 - The same symptoms do not always mean the same root cause
- Over inform is better than under
 - Minor changes may seem unrelated
 - Comparative information is valuable

WHICH application is impacted

- Is it a single application?
- All applications accessing a single database?
- All applications?

WHO is impacted

- Does it impact all users?
- Only the users running a certain configuration (direct/citrix)?

WHERE are the impacted users located

- Is this limited to a single geographic location?
- Is this limited to a single network zone?

WHEN did it first start

- Did the change occur suddenly?
- Was it a gradual change?
- When was performance acceptable?

Stage 2

- Why/How
- Focus on root causes and resolutions

WHY is this occurring

- Baselines are key!
- Profile/gather data

HOW can we fix it

- Here's the crux!
- "Optimizing Oracle response time is, for the most part, a solved problem."
 - *Optimizing Oracle Performance*, pxiii
 - *I must respectfully disagree*
- Diagnosis != Solution

Don't Do It!

- "The most highly optimized sql statement is the one that is never run."
 - Anjo Kolk
- Code that runs...and never returns rows

Managing load

- Decrease the load
 - Reduce resource consumption
- Spread the load
 - Find 'slow' times

Avoid CTD

- Set a target
 - Make is measurable
- Stop when
 - The target is achieved
 - It is known that the target cannot be achieved
 - It is not worth additional resource to achieve the target

Statistics

- Overview
- Strengths & Weaknesses
- Using Statistics for Diagnosis

Overview

- **Statistics are just counters, nothing more**
 - Number of times
 - Amount of time
- **They are tracked at the session level**
 - Session values are rolled up into the system value

Strengths & Weaknesses

- **Strengths**
 - Identify what is happening right now
 - Compliments timed events
- **Weaknesses**
 - Cannot look backward
 - Might miss contention issues

Using Statistics for Diagnosis

- Counters stored in views
 - Monotonically increasing values
 - Only deltas are meaningful
- Updated near real time

Important views

- v\$sysstat
- v\$sesstat
 - v\$statname
- v\$essio
- v\$sql

v\$sysstat

- Rollup of all session statistic values since instance start
- Columns
 - statistic#
 - name
 - value

v\$sesstat

- Session level statistic values
 - sid (from v\$session)
 - statistic#
 - value
 - name (from v\$statname)

Important Statistics

- CPU used by this session (in centiseconds)
- session logical reads
 - consistent gets
 - db block gets
 - consistent changes
 - db block changes
- physical reads/physical writes

v\$sqlsession

- Session level i/o statistics
 - sid
 - block_gets
 - consistent_gets
 - physical_reads
 - block_changes
 - consistent_changes

v\$sql

- Statistics for individual statements*
 - sql_id/sql_child_number
 - parse_calls
 - disk_reads
 - direct_writes
 - buffer_gets
 - cpu_time
 - elapsed_time

What is my session doing?

- Use statistics to find non-timed event activity
- For example - CPU Utilization
 - Reading data from memory
 - Creating read consistent view
 - Non-data activity

```

SELECT sess.username,
       sess.sid||','||sess.serial# sid_serial,
       stnm.name,
       sesst.value
FROM   v$session sess,
       v$sesstat sesst,
       v$statname stnm
WHERE  sess.sid = 1004
       AND stnm.name IN
          ('CPU used by this session',
          'session logical reads')
       AND stnm.statistic# = sesst.statistic#
       AND sesst.sid = 1004

```

09:36:30 SQL> /

User	Sid,Serial	Statistic	Value
DEMO	1004,944	session logical reads	130467768
DEMO	1004,944	CPU used by this session	191886

09:36:31 SQL> /

User	Sid,Serial	Statistic	Value
DEMO	1004,944	session logical reads	130659110
DEMO	1004,944	CPU used by this session	192548

Timed Events

- Overview
- Strengths & Weaknesses
- Using Event Information for Diagnosis

Overview

- Instrumented code
 - Not all code is instrumented
 - # of events changes between releases
- Important Events
- Idle events
- Event States

Important Events

- physical i/o
 - db file sequential read (single block)
 - db file scattered read (multi block)
 - direct path read/direct path write
- logical i/o
 - buffer busy waits/read by other session
- locking/latching
 - enqueue
 - latch free

"Idle" events

- There are some events that have been designated as 'idle'
 - Implies that they do not contribute to the user experience of response time
 - "Can be safely ignored"
- It has been my experience that these events can often contribute time to the user response time.

Event States

- **WAITED UNKNOWN TIME**
 - `timed_statistics` is set to `FALSE`
- **WAITED SHORT TIME**
 - duration of less than 1 centisecond
- **WAITING**
 - event is current
- **WAITED KNOWN TIME**
 - event has completed

Strengths & Weaknesses

- **Strengths**
 - Identify what is happening right now
 - Limited history in 10g
- **Weaknesses**
 - Cannot look backward*
 - Might miss contention issues

Finding Event Information

- Views can tell you what the session is currently waiting on or last waited on
 - HOWEVER, the session may be doing something else!
- v\$sqlsession_wait
- 10g enhancements
 - v\$sqlsession
 - v\$sqlsession_wait_history

```
SELECT vs.username,
       vs.sid||','||vs.serial# sid_serial,
       vsw.state wait_state,
       vsw.seconds_in_wait,
       vsw.event,
       DECODE(vsw.pltext, NULL, NULL,
              vsw.pltext||'='||vsw.p1) p1,
       DECODE(vsw.p2text, NULL, NULL,
              vsw.p2text||'='||vsw.p2) p2,
       DECODE(vsw.p3text, NULL, NULL,
              vsw.p3text||'='||vsw.p3) p3
FROM v$sqlsession vs,
     v$sqlsession_wait vsw
WHERE vs.sid = 1083
      AND vs.sid = vsw.sid
```

```

13:37:42 SQL> /
User      Session ID Wait State  Wait Seconds
-----
Event                P1          P2          P3
-----
SCOTT      1083,12808 WAITING          0
db file sequential read  file#=8  block#=1576387  blocks=1
13:37:44 SQL> /
User      Session ID Wait State  Wait Seconds
-----
Event                P1          P2          P3
-----
SCOTT      1083,12808 WAITING          0
db file sequential read  file#=8  block#=1455808  blocks=1

```

```

SELECT vs.username,
       vs.sid||','||vs.serial# sid_serial,
       vs.state wait_state,
       vs.seconds_in_wait,
       vs.event,
       DECODE(vs.pltext, NULL, NULL,
              vs.pltext||'='||vs.p1) p1,
       DECODE(vs.p2text, NULL, NULL,
              vs.p2text||'='||vs.p2) p2,
       DECODE(vs.p3text, NULL, NULL,
              vs.p3text||'='||vs.p3) p3
FROM sys.v$session vs
WHERE vs.sid = 1083

```

```

13:37:42 SQL> /
User      Session ID Wait State  Wait Seconds
-----
Event          P1          P2          P3
-----
SCOTT      1083,12808 WAITING      0
db file sequential read  file#=8  block#=1576387  blocks=1
13:37:44 SQL> /
User      Session ID Wait State  Wait Seconds
-----
Event          P1          P2          P3
-----
SCOTT      1083,12808 WAITING      0
db file sequential read  file#=8  block#=1455808  blocks=1

```

```

SELECT vs.username,
       vs.sid||','||vs.serial# sid_serial,
       vsw.wait_time,
       vsw.wait_count,
       vsw.event,
       DECODE(vsw.pltext, NULL, NULL,
              vsw.pltext||'='||vsw.p1) p1,
       DECODE(vsw.p2text, NULL, NULL,
              vsw.p2text||'='||vsw.p2) p2,
       DECODE(vsw.p3text, NULL, NULL,
              vsw.p3text||'='||vsw.p3) p3
FROM v$session vs,
     v$session_wait_history vsw
WHERE vs.sid = 1083
      AND vs.sid = vsw.sid
ORDER BY vs.username, vs.sid, vs.serial#, vsw.seq# ASC

```

User	Session ID	Wait State	Wait Seconds
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1576387	blocks=1
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1326664	blocks=1
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1365528	blocks=1
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1325991	blocks=1
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1596969	blocks=1
SCOTT	1083,12808	WAITING	0
db file sequential read	file#=8	block#=1455808	blocks=1

Case Study

- Sudden slowness in report
 - Statistics & Timed Events showed read consistent view generation
 - Physical I/O on undo tablespace data files
 - High CPU utilization
 - Tracked back to update with new code
 - Update was a batch job that *should* have run overnight

Extended SQL Trace

- SQL Trace v. Extended SQL Trace
- Strengths & Weaknesses
- Using Extended SQL Trace for Diagnosis

SQL Trace v. Extended SQL Trace

- SQL Trace
 - records database calls
 - records basic resource information
- Extended SQL Trace
 - records database calls
 - records basic resource information
 - records bind variable values
 - records timed event information

Strengths & Weaknesses

■ Strengths

- Identify what is happening right now
- Complete record of calls
- Preserved
- Can be profiled

■ Weaknesses

- Cannot look backward
- Might miss contention issues
- Have to be able to repeat problem
- Connection architecture can complicate tracing

Activating Extended SQL Trace

- In order to enable extended sql trace for a given session, you must be able to uniquely identify that session
- Ideally, you trace only the process/action of interest
- Levels
 - Off - Level 0
 - Basic - Level 1/Level 2
 - Bind Variables - Level 4
 - Timed Events - Level 8
 - Bind Variables & Timed Events (WAITS) - Level 12

Enabling Extended SQL Trace

- Many Ways to enable extended SQL Trace
 - Do not set at instance level
 - Session level
 - Logon triggers
 - Application Code
 - Manual
- http://www.petefinnigan.com/ramblings/how_to_set_trace.htm

Full session

- Logon trigger
 - Enables tracing for entire session
 - Set and forget
- Permissions
 - User must have 'alter session' granted directly
- sys_context
 - Use to identify user
 - Add user specific information to tracefile name

```

CREATE OR REPLACE TRIGGER demo.starttracecust
  AFTER LOGON ON demo.SCHEMA
BEGIN
  IF UPPER(SYS_CONTEXT('USERENV', 'OS_USER')) IN
    ('CMILLSAP', 'MNORGAARD', 'LDEHAAN')
  THEN
    EXECUTE IMMEDIATE
      'ALTER SESSION SET timed_statistics = TRUE';
    EXECUTE IMMEDIATE
      'ALTER SESSION SET max_dump_file_size = UNLIMITED';
    EXECUTE IMMEDIATE
      'ALTER SESSION SET tracefile_identifier = '
      || SYS_CONTEXT('USERENV', 'OS_USER');
    EXECUTE IMMEDIATE
      'ALTER SESSION SET events ' || CHR(39) ||
      '10046 trace name context forever,level 12' || CHR(39);
  END IF;
END;
/

```

Application code

- Ideally, application code will have calls to enable extended sql trace
 - Enables tracing just the process of concern
 - Can be table driven
- May require "special" version of code

In flight

- Locate session of interest
 - Connection pooling, multithreaded server, single-call session architecture may cause problems
- Enable extended sql trace
 - dbms_system
 - dbms_support
 - dbms_monitor

```
dbms_system.set_ev(10,20,10046,8,'');  
  
dbms_support.start_trace_in_session(10,20,  
    waits=>true,binds=>false);  
  
dbms_monitor.session_trace_enable(10,20,  
    waits=>TRUE,binds=>FALSE);
```

Using Extended SQL for Performance Diagnosis

- Reading raw trace files
- Using a profiler
 - tkprof
 - Hotsos Profiler
 - OraSRP
- Identify response time consumers

Reading Raw Trace files

- Often the only source of detailed information
 - It is a transcript of the process
- Use your pattern matching skills

tkprof

- Basic tool available to all installations
- Transient Kernel Profiler
 - Generates summarized information
 - 9i first version to acknowledge waits
 - Can run any version of tkprof against a trace file
- Not complete, but pretty darn good
 - 3rd Party Tools better...but cost more

tkprof

- Command line

- trace file
- output file
- sort
- waits
- sys

```
tkprof test.trc test.tkp sort=prsela,exeela,fchela  
waits=yes sys=no
```

```

SELECT CODE,NAME,DATE_BEGIN,DATE_END,P_NAME,P_DATE,QTY
FROM SCOTT.VIEW_1
WHERE CODE=:p1
      AND DATE_BEGIN= TO_DATE(:p2, 'MM/DD/YY hh24:mi:ss')
      AND DATE_END= TO_DATE(:p3, 'MM/DD/YY hh24:mi:ss')
      AND P_DATE>= TO_DATE(:p4, 'MM/DD/YY hh24:mi:ss')
      AND P_DATE<= TO_DATE(:p5, 'MM/DD/YY hh24:mi:ss')

```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.10	0.30	0	0	0	0
Fetch	43	184.93	815.58	1285382	15144145	0	629
total	45	185.03	815.89	1285382	15144145	0	629

Rows	Row Source Operation
629	SORT GROUP BY
629	FILTER
629	NESTED LOOPS
629	NESTED LOOPS
629	NESTED LOOPS
629	NESTED LOOPS
1097754	NESTED LOOPS
1097754	HASH JOIN
7129358	TABLE ACCESS BY INDEX ROWID T1
14614433	NESTED LOOPS
1493824	TABLE ACCESS BY INDEX ROWID T2
1493824	INDEX RANGE SCAN IX_T2
13120608	INDEX RANGE SCAN IX_T1
15	TABLE ACCESS FULL T3
1097754	INDEX UNIQUE SCAN PK_T3
629	TABLE ACCESS BY INDEX ROWID T4

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Waited
-----	Waited	-----	-----
SQL*Net message to client	43	0.00	0.00
db file sequential read	1244879	1.83	604.51
direct path write temp	3096	15.11	24.94
direct path read temp	2700	0.27	4.16
db file scattered read	1	0.00	0.00

Identify Response Time Consumers

- Profile output
 - Sanity check for calls
- Raw trace file
 - Scan
 - Discard known idle time

Case Study

- Excessive Parsing
 - Overnight batch process taking too long
 - Tracing 1 application server
 - 35% of time spent parsing statement that updated 0 rows

Recap

- We now have the ability to look at session/statement information
- What if
 - that is not enough?
 - problems are reported from a week ago?
 - users complain it is different now?
 - contention is the problem?

Statspack

- Overview
- Strengths & Weaknesses
- Using Statspack for Diagnosis

History

- Introduced in 8i
 - Retrofit to 8.0
 - Still present in 11g
- No charge to install/use
 - Widely accepted
 - Statspack viewers available

Snapshot

- **Generating snapshots**

- Automated v. Manual
- Use Comments

- **Levels**

- 5 is default
- 6 gathers sql plan information *

- **Avoiding snapshot creep**

```
interval=>'trunc(sysdate, 'HH24') +  
((floor(to_number(to_char(sysdate,  
'MI')))/30)+1)*30) / (24*60)'
```

Reporting

- **Spreport**

- Version specific...beware of upgrades
- Lots of information
 - Very useful
 - Almost useless

- **Analytical sql**

- Useful for calculating deltas
- LAG/LEAD, TOP N

Strengths & Weaknesses

- Strengths
 - Identify changes
 - Spot trends
 - Might see contention issues
- Weaknesses
 - Cannot look at session
 - System perspective might be misleading

Using Statspack for Diagnosis

- Baselines/Current Stats
- Resource consumption
 - SQL activity
- Change control

SQL statement performance during period

- Summary - Top 15
 - Details
- Compare
 - Statement execution plan

SQL Hash Value	Elapsed Seconds	ELA	CPU Seconds	CPU	Memory Reads	LIO
991767228	573,910	1	550,041	1	5,329,725,555	5
1734005396	308,814	2	89,978	5	5,180,037,351	6
18145424	276,619	3	255,912	2	10,339,702,486	2
59317890	259,883	4	10,609	9	505,986,789	10
1864143593	241,138	5	209,655	3	10,427,480,396	1
353214313	182,234	6	165,734	4	8,058,994,503	3
3577213919	75,796	7	64,770	6	6,105,216,392	4
3384917102	61,531	8	3,108	18	105,825,982	18
4212608483	41,328	9	11,409	8	262,504,877	12
2530090178	31,424	10	1,168	35	46,455,157	38
1524128321	24,641	11	2,838	21	45,578,935	40
309891247	23,556	12	2,601	23	52,474,047	32
1855792429	21,302	13	688	67	0	4800
831455651	21,248	14	2,336	25	46,934,262	37
1190164045	21,018	15	683	69	0	4800

*** Statement Text ***

Formatted SQL Statement

```
-----  
SELECT DISTINCT PP1.PERSONAL_NUM PERSONAL_NUM,  
PP1.PERSONAL_REF_NAME REF_NAME, PP1.PERSONAL_REF_ID REF_ID,  
UPPER(PP1.PERSONAL_REF_ID || ' - ' || PP1.PERSONAL_REF_NAME ||  
' - ' || PP1.PERSONAL_LAST_NAME || ', ' || PP1.PERSONAL_FIRST_NAME ||  
' - ' || PP1.PERSONAL_SS_NUM ) || ' - ' || UPPER(PP1.PERSONAL_CITY ||  
' - ' || PP1.PERSONAL_ZIP) NAME_ID  
FROM PATIENT_PERSONAL PP1,  
CLAIM_HEADER CHI  
WHERE ( PP1.PERSONAL_NUM = CHI.PERSONAL_NUM )  
and (:arg_unitnum IS NULL ) or ( CHI.UNIT_NUM = :arg_unitnum)  
and (( PP1.PERSONAL_STATUS = 'A' ) OR  
(PP1.PERSONAL_STATUS = 'I' AND :arg_showactiveonly = 'N' ))  
and ( UPPER(PP1.PERSONAL_REF_ID) LIKE :arg_patient )  
ORDER BY name_id
```

*** Execution Plan(s) ***

HASH VALUE = .2407486190 SNAP ID = 2896

```
-----  
| Id | Operation | Name | Rows | Bytes | TempSpc | Cost (%CPU)|  
-----  
| 0 | SELECT STATEMENT | | | | | 11427 (100)|  
| 1 | SORT UNIQUE | | 21061 | 1624K | 3672K | 11034 (2)|  
| 2 | HASH JOIN | | 21061 | 1624K | | 10642 (2)|  
| 3 | TABLE ACCESS FULL | PATIENT_PERSONAL | 14546 | 994K | | 1549 (4)|  
| 4 | TABLE ACCESS FULL | CLAIM_HEADER | 119K | 1051K | | 9090 (2)|  
-----
```

*** Run Time Statistics ***

# of Executions	=>	96,660			
# of Rows	=>	12,163,820	# of Rows/Exec	=>	126
# of Memory Reads	=>	5,720,650,532	# of LIOS/Exec	=>	59,183
			# of LIOS/Row	=>	470
# of Disk Reads	=>	130,697,281	# of PIOs/Exec	=>	1,352
			# of PIOs/Row	=>	11
CPU Seconds	=>	177,226	CPU Secs/Exec	=>	1.834
			CPU Secs/Row	=>	.015
Elapsed Seconds	=>	863,888	Ela Secs/Exec	=>	8.937
			Ela Secs/Row	=>	.071

Case Study

- Month End Close slow
- Traced user processes
 - Physical I/O
 - Buffer Cache Contention
- Top Resource Consuming SQL
 - Too many sessions running the same poor sql that was not from the user processes
 - Tuned sql - reduced resource consumption & contention

Know which tool is appropriate

- Tuning specific sql
- Identifying what is consuming time in a single session
- Reducing resource consumption

Conclusion

- No one tool can solve every problem
 - A good process is a repeatable process
 - Learn as much as you can
- Build a tool kit you can use
 - Start with your brain
 - Use scripts

- Questions?
 - daniel.fink@optimaldba.com
- Presentations/Code
 - www.optimaldba.com
- RMOUG Training Days 2008
 - February 12 - 14, 2008
 - Denver, Colorado
 - www.rmoug.org