

A few months ago, I was listening to a local radio show that deals with automotive problems. A caller had a very perplexing issue and the hosts recommended that he take the car in to a mechanic so that they could diagnose the problem. The key point that I recall is the discussion of “diagnostic routines” and how they make decisions based on the results of simple tests and gathering of data. A few days later, a colleague asked me to diagnose a sql issue and we started talking about the steps I used to identify and resolve the issue. I realized that I was using my own diagnostic routines inside my head...but I never had formalized the process. Documenting the processes I use to diagnose problems is quite a challenge. We all know how to do things...it is often difficult to explain and document the reasons why we do what we do.

What happens when a user calls complaining that their session is hung or a developer wonders why a sql statement is not running well? Do you have a documented process or just try the ABORS* approach? When you do solve a problem, do you know why so you can repeat your success the next time? Without a disciplined approach to diagnosing problems, your work will be inefficient, ineffective and with sporadic results. Tips and tricks that used to work may be implemented with ineffective (even disastrous) impact because the problems are not the same, though the symptoms may be.

One of the issues in Oracle Administration (and IT in general) is a lack of discipline in diagnosing issues. Last year I read a paper that Robyn Sands was working on regarding using an industrial engineering approach to managing Oracle databases. It was a revelation.

There is not a ‘Grand Unified Oracle Tuning Process’. No ‘One Size Fits All’ checklist. No ‘Silver Bullets’ (except if you are working at Coors). You will have to develop your own processes for your own systems. To do this, you will need to understand the system, know what tools are available to you, and the fundamentals of diagnosis. The key to effective diagnosis is to have and follow processes that have been proven to work. This is not to say that the process will work every time...most systems are far too complex. However, you should be able to design and implement a process that is able to diagnose the most common issues for your system. In practice, the diagnostic process will be comprised of multiple sub-processes.




Goals

- Learn how to develop effective routines to diagnose problems
- Review knowledge, tools and processes
 - Diagnosing a system issue
 - Diagnosing a session issue
 - Diagnosing a statement issue



Disclaimers

- Not in-depth technical
 - More method/process oriented
 - Will provide tips on commands/tools
- Won't feed you fish
 - Focus on learning to fish



Winners know Why they Win

Dave Ensor

March 31, 2011

www.optimaldba.com

Page 4 of 87*

“Winners know why they win.” is a quote from Dave Ensor, the Grand Man of Oracle Performance. The key message is to know why you are doing a task. This means you have the triumvirate of disciplined diagnosis: tools, knowledge and process.

Different problems can exhibit similar symptoms; similar problems can exhibit different symptoms. If you cannot differentiate and determine root cause, the solution you implement may not solve the problem, it may even make the problem worse! I recently encountered an issue where the top timed event for a database system was ‘log file sync’, which is related to transaction commit processing. The ‘standard’ solution was to decrease the frequency of commits, but too frequent commits were only part of the possible cause for log file sync events. When the less frequent commits solution was implemented...the application performance did improve, but transactions were frequently failing with ITL deadlocks, which can be caused by too long and too large transactions...a side effect of reducing commits.

We All Diagnose Problems

- Every day issues
 - Car won't start
 - House too cold
 - Gravy too thin
- Learned routines
 - Symptom(s)/Decision(s)/Solution(s)
 - Adapt for conditions/situations

March 31, 2011

www.optimaldba.com

Page 5 of 87*

We all diagnose problems in every day life, problems that have nothing to do with computers.

Think about what you do when your car won't start. What process do you use? What questions do you ask? Tests do you perform?

You also adapt for conditions. For example, if you just filled up the car with gas, you probably won't ask the question "Am I out of gas?" Not that you know for certain that the engine is getting fuel.



Car stopped moving...

- | | |
|-------------------------|------------------------|
| 1. Out of gas | A. Don't tailgate |
| 2. Engine failure | B. Take foot off brake |
| 3. Alternator dead | C. Fix engine problem |
| 4. Foot on brake | D. Add gasoline |
| 5. Engine off | E. Repair alternator |
| 6. Car in front stopped | F. Turn on engine |



Agenda

- Overview
- Knowledge
- Tools
- Process
- Diagnostic Routine Examples



Overview

- Current state of diagnosis
- Convergence
- Problems and Errors
- What is a good routine and why should I use it?

ABORS*

- Guess and Grimace
- Groundhog Day
- Whack-A-Mole
- Panic and Point
- IAMM
- In Search Of

March 31, 2011

www.optimaldba.com

Page 9 of 87*

ABORS stands for A Bunch Of Random Stuff

The current state of Oracle diagnosis is less than disciplined. While we have made great strides in the last decade using approaches such as response time optimization, we are still not where we need to be.

Guess and Grimace This term was coined by Mogens Norgaard to describe the actions of people who engage in troubleshooting exercises using random ideas (brainstorming) instead of taking a more disciplined scientific approach. I often see this when

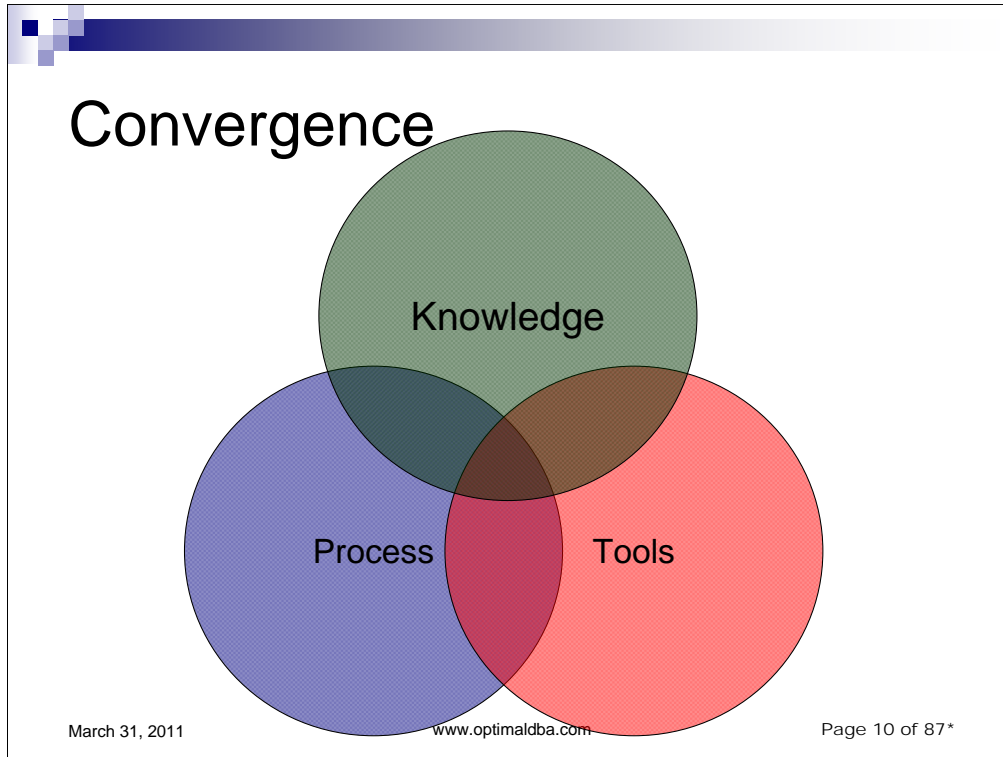
Groundhog Day Another Mogens term, this describes repetitive efforts that are not producing lasting results.

Whack-A-Mole This one is from Robyn Sands and describes the situation where the same situation keeps coming up and you never really solve it.

Panic and Point identify performance problems as originating anywhere except in your area of responsibility

IAMM How often do you hear the terms 'Firefighting', 'Tiger Team' or 'SWAT Team'? This is "Inappropriate Allocation of Macho Metaphor". One of my issues is that these teams practice/practice/practice

In Search of Solutions in search of problems - people propose solutions that they know, may have worked one time, but don't really have any analysis behind them.



A disciplined routine is the convergence of Knowledge, Tools and Process

Knowledge alone = helpless

Tools alone = dangerous

Process alone = bureaucratic

Knowledge + Tools = inconsistent

Knowledge + Process = inefficient

Tools + Process = ineffective

Knowledge + Tools + Process = Consistent, Effecient, Effective



The Role of Problem and Error

- Problem
 - The reason you develop a routine
- Error
 - The reason routines fail

Problems

- Simple
 - Baking a cake
- Complicated
 - Sending a rocket into space
- Complex
 - Raising a child

March 31, 2011

www.optimaldba.com

Page 12 of 87*

These descriptions are very subjective and must be considered in context. What is simple to a gourmet chef may be complex to a rocket scientist and vice versa.

Simple – like baking a cake. A few basic techniques are required.

Complicated – like sending a rocket to the moon. Sometimes can be broken down into a series of simple problems. Frequently requires multiple people/teams and specialized expertise. Timing and coordination become serious concerns. Once you learn and document, the process is repeatable with repeatable results.

Complex – like raising a child. Situations are unique and may require different approaches. Outcomes remain highly uncertain.

When learning how to develop a routine, pick easy, but important, problems.

Errors

- Error of Ignorance
 - Lack of Knowledge
- Error of Ineptitude
 - Failure to Apply Knowledge

March 31, 2011

www.optimaldba.com

Page 13 of 87*

If there is an issue with a routine, try to determine what category the error falls into and correct it.

Error of ignorance – increase/expand knowledge

Error of ineptitude – fix or adhere to the process.

What Makes A Good Routine?

- Relevant
- Comprehensive
- Efficient/Effective
- Deterministic
- Narrow Scope

March 31, 2011

www.optimaldba.com

Page 14 of 87*

Relevant – the routine is relevant to the application, system, version, situation. Using an 7.3 diagnostic routine for a 10.2.0.4 database is not necessarily relevant. If you find that you are skipping tests/questions because they no longer apply, it is time to review and update the routine.

Comprehensive – it needs to be able to cover almost every likely scenario. If you find yourself skipping steps because they are no longer relevant, it is likely that you are also missing steps.

Efficient – a routine that wastes time to arrive at the root cause is not going to be all that useful. The idea behind a routine is to have a process you can follow while under pressure.

Deterministic – if the same root cause with the same symptoms results in different diagnosis, then you have to question the routine.

Efficient

Process oriented

Act to save time, money, effort

Effective

Result Oriented

Getting the job done

Quality

Narrow

- Proper solutions are as narrowly scoped as possible
 - Decreases 'collateral damage'
 - Fewer side effects
- Requires a narrowly defined root cause

Very important to know what is happening; and to know what is NOT happening.

A Moment of Zen

$$S = 1/P$$

*The Solution is the
Inverse of the Problem*

March 31, 2011

www.optimaldba.com

Page 16 of 87*

Solutions and Problems are inherently related. If the solution is adding gasoline to the tank, then the problem had better be that the tank was out of gasoline. If the real problem is a flat tire, broken drive shaft or burned out transmission, adding gasoline to the tank will not get the car moving again. If the problem is that an index is not available for a query, then the solution needs to be creating an index or making it available for use.

Recall the log file sync and ITL deadlock from the introduction.

Knowledge

- “Your” System
 - Layers
 - Usage
- OUG so Oracle...

March 31, 2011

www.optimaldba.com

Page 17 of 87*

It is exceedingly difficult to diagnose problems when you don't understand the system. Actually, it is not difficult at all...it is impossible! Would you want your stalled car diagnosed by someone who has never seen an engine before? Or chest pain diagnosed by an auto mechanic? And yet many people are called on to diagnose and resolve problems with applications for which they have little training, documentation and understanding.

When it comes to an Oracle database/session/sql statement, it is important to understand how the database/session/statement work. How does a statement process data? What are the 4 states of a session? What do different timed events (wait events) mean? If you don't understand Oracle, you can't properly diagnose an Oracle problem. And it is not just understanding Oracle, you need to have accurate knowledge of the current version of Oracle. Trying to diagnose an issue on a 10g database using 7.3 knowledge is not going to be successful.



Oracle Architecture

- Instance
 - Processes
 - Memory
- Database
 - Files/Storage
 - Data

Latches and Locks

- Latches/mutexes
 - Protect memory structure
 - No queueing
- Locks/enqueues
 - Protect data structures
 - Queueing
- Deadlocks

March 31, 2011

www.optimaldba.com

Page 19 of 87*

The longer a lock is held, the higher the chance of encountering a blocking situation.

SGA – too much memory

Database – too long of transaction



Timed Events

- Not “waits”
 - System calls
 - Not every call is instrumented
 - Infinite/Renew

States/Times

■ States

- WAITING
- WAITED SHORT TIME
- WAITED KNOWN TIME
- WAITED UNKNOWN TIME

■ Times

- WAIT_TIME
- SECONDS_IN_WAIT*

March 31, 2011

www.optimaldba.com

Page 21 of 87*

Waiting – session is currently waiting. seconds_in_wait is incrementing

waited short time – wait was less than 1 centisecond

waited known time – wait enumerated in WAIT_TIME

waited unknown time – timed_statistics is off

WAIT_TIME – time waited on completed event

SECONDS_IN_WAIT – time since last event completed. NOTE – IF state is WAITED% and seconds_in_wait is incrementing, then the session is in an uninstrumented call or a bug.



Key events

- db file/direct path reads
- enqueues
- latches
- CPU
- “Idle” events
- <insert important events here>



Session States

- On CPU
- Waiting on CPU
- Instrumented Event
- Uninstrumented Event



Statement Processing

- Parse, Execute, Fetch

- Parsing/Optimizing

- Fetch

- Parallelism

- Run time

- Can be masked



Data Changes

- Redo, redo, undo, do
- Read Consistency



Oracle Options

- RAC
- Partitioning
- Parallel Query
- Version

Tool Proficiency

- What tools are at your disposal?
 - Strengths/Weaknesses
 - Blind Spots
- What situation calls for what tool?
 - Systemic
 - Session
 - Statement

March 31, 2011

www.optimaldba.com

Page 27 of 87*

A tool is only as good as the developer. And it will only show you what the developer considers to be important. One great example is that the Oracle instrumentation shows a great deal of information about physical reads, but almost no information about logical reads.



Tools

- SQLPlus
- oradebug
- GUI Tools
- Tracing/Profiling

SQLPlus

■ Commands

- sqlplus –prelim
- Events
- Hanganalyze/state dumps

■ Scripts

- AWR/Statspack
- ASH

March 31, 2011

www.optimaldba.com

Page 29 of 87*

Preliminary connection – attaches only to memory.

Note 986640.1

Able to execute limited commands



Events

- **Categories**

- dump information on request
- dump information on specific error
- generate trace diagnostics
- change database behavior

- **Mostly undocumented/unsupported**

- orausr.msg file



State Dumps

- Systemstate
 - entire system (all processes)
- Processstate
 - single process
 - oradebug for other sessions



Hanganalyze

- When the system is completely unresponsive
- Note 452358.1
- Document process...later....



GUI Tools

- SQL Developer/TOAD
- Enterprise Manager/Grid Control



Logging/Tracing/Profiling

- System logs
 - alert.log
 - trace files
- Tracing
 - raw data for analysis/profiling
- Profiling
 - summarizing/narrowing



Tracing

- SQL/Session
 - event 10046
 - dbms_monitor
- CBO
 - event 10053



What is Extended SQL Trace?

- SQL Trace writes all of the sessions database calls to a plain text file
- Extended SQL Trace includes
 - Bind variable values and/or
 - Wait events and durations
- Resulting output can be used to diagnose performance issues



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

March 31, 2011

www.optimaldba.com

Page 37 of 87*



Profiling

- `dbms_xplan`
- `tkprof`
- `dbms_profiler`
- `runstats test harness`

dbms_xplan

- `/*+ gather_plan_statistics */`
- `dbms_xplan.display_cursor`
 - `sql_id`
 - `child_number`
 - `format "ALLSTATS LAST"`
 - `*set serveroutput off`

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
 - Improvements in each release
 - 3rd Party Tools better...but cost more

tkprof Command

- trace file
 - output file
 - sort
 - Exclude to get a session replay
 - Cumulative
 - waits
 - sys
 - Don't forget "/ as sysdba" = sys
- ```
tkprof test.trc test.tkp sort=prsela,exeela,fchela
waits=yes sys=no
```



# DBMS\_PROFILER

- Indicates which steps in the pl/sql program units are taking the time
- To set it up, run the profrep.sql script
  - Can be found in \$ORACLE\_HOME/plsql/demo
  - In some installations, this seems to be missing, but can be downloaded from Oracle Technet
- Do not use profsum.sql as it is a resource hog
  - Custom version available online



# DBMS\_PROFILER

- start\_profiler
- do something in pl/sql
- stop\_profiler
- profsum

# DBMS\_PROFILER call

```
declare
 status number;
begin
 status := dbms_profiler.start_profiler('PROFILER DEMO');
end;
/

execute profdemo_pack.profdemo_p1;

declare
 status number;
begin
 status := dbms_profiler.stop_profiler;
end;
/
```

# profsum Output

Percentage of time in each module, summarized across runs

| UNIT_OWNER  | UNIT_NAME            | SECS | PERCENTAG |
|-------------|----------------------|------|-----------|
| <anonymous> | <anonymous>          | .10  | 62.50     |
| DEMO        | PROFDEMO_TRIGG<br>ER | .02  | 12.50     |
| DEMO        | PROFDEMO_PACK        | .02  | 12.50     |
| DEMO        | PROFDEMO_TYPE        | .02  | 12.50     |

Lines taking more than 1% of the total time, each run separate

| RUNID | HSECS  | PCT   | OWNER | UNIT_NAME            | LINE# | TEXT                                              |
|-------|--------|-------|-------|----------------------|-------|---------------------------------------------------|
| 3     | 107.26 | 671.7 | DEMO  | PROFDEMO_PACK        | 9     | insert into<br>profdemo_tab_1 values<br>(d3);     |
| 3     | 47.37  | 296.7 | DEMO  | PROFDEMO_TRIGG<br>ER | 2     | after insert on<br>profdemo_tab_1 for<br>each row |



## Runstats Test Harness

- Basic method to compare 2 statements

[http://asktom.oracle.com/pls/asktom/ASKTOM.download\\_file?p\\_file=6551378329289980701](http://asktom.oracle.com/pls/asktom/ASKTOM.download_file?p_file=6551378329289980701)

- It must be installed by DBA and certain privileges granted to v\$views
  - Also need to grant select on v\$timer
- You can set a threshold in the report so that only certain statistics are reported



## runstats Test Harness

```
SQL> exec runStats_pkg.rs_start;
<...Run statement1 here>
SQL> exec runStats_pkg.rs_middle;
<...Run statement2 here>
SQL> exec runStats_pkg.rs_stop;
<...Run statement1 here>
SQL> EXEC
 runstats_pkg.rs_stop(10000000);
```

```

Run1 ran in 247488 hsecs
Run2 ran in 8733 hsecs
run 1 ran in 2833.94% of the time

```

| Name                           | Run1        | Run2   | Diff          |
|--------------------------------|-------------|--------|---------------|
| STAT...buffer is not pinned co | 10,512,398  | 11,081 | -10,501,317   |
| STAT...buffer is not pinned co | 10,512,398  | 11,081 | -10,501,317   |
| STAT...session logical reads   | 15,144,888  | 15,655 | -15,129,233   |
| STAT...consistent gets         | 15,144,165  | 14,234 | -15,129,931   |
| STAT...consistent gets from ca | 15,144,165  | 14,234 | -15,129,931   |
| STAT...consistent gets         | 15,144,165  | 14,014 | -15,130,151   |
| STAT...consistent gets from ca | 15,144,165  | 14,014 | -15,130,151   |
| STAT...session logical reads   | 15,144,888  | 14,724 | -15,130,164   |
| STAT...table fetch by rowid    | 15,714,083  | 26,331 | -15,687,752   |
| STAT...table fetch by rowid    | 15,714,083  | 26,331 | -15,687,752   |
| STAT...buffer is pinned count  | 24,608,058  | 45,435 | -24,562,623   |
| STAT...buffer is pinned count  | 24,608,058  | 45,435 | -24,562,623   |
| LATCH.cache buffers chains     | 27,019,456  | 35,583 | -26,983,873   |
| LATCH.cache buffers chains     | 27,019,456  | 31,620 | -26,987,836   |
| STAT...session uga memory max  | 111,778,104 |        | 0-111,778,104 |
| STAT...session uga memory max  | 111,778,104 |        | 0-111,778,104 |
| STAT...session pga memory max  | 116,981,760 |        | 0-116,981,760 |
| STAT...session pga memory max  | 116,981,760 |        | 0-116,981,760 |

```

Run1 latches total versus runs -- difference and pct

```

| Run1       | Run2    | Diff        | Pct        |
|------------|---------|-------------|------------|
| 63,074,782 | 126,354 | -62,948,428 | 49,919.10% |



# Process

- Right, Right, Right
- Approaches
- Components
- Symptoms
- Review, Revise, Repeat



# Right, Right, Right

- Right Starting Point
- Right Path
- Right Tools



# Symptoms

- THE Starting Point!

- Very critical
- Work on wrong problem = FAIL!

- Doubt

- People tell you what they think happened
- Require hard evidence
- “How do you experience that problem?”



## Review, Revise, Repeat

- Document execution
- Post Event
- Continuous Improvement

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 52 of 87\*

If the process does not lead you to the root cause and solution, it needs fixed!  
What was the kind of error you encountered? Ignorance or Ineptitude?



# Diagnostic Approaches

- Approach v. Process
  - Approach is the strategic
  - Process is the tactical
- Approach and Process need to align

# Approaches

- Method R
- Resource Consumption
- Throwaway
- Cardinality Feedback
- Variance
- Root Cause Analysis

March 31, 2011

www.optimaldba.com

Page 54 of 87\*

## **Diagnostic Methods**

There are quite a few methods that will assist in diagnosing problems. It is important to look at any method with a critical eye and understand how it will assist you in resolving the issue at hand. There are three methods that I use consistently to focus my efforts and solve problems quickly. Incorporate these methods and any others you find useful into the process.

### **Response Time**

Method-R is the approach that you should focus on response time as the performance metric. When looking at a slow process, prioritize your work based on what steps in the process are consuming the most time. For additional information, you can visit <http://www.method-r.com/> or read "Optimizing Oracle Performance".

### **Resource Consumption**

There are times where you are dealing with systemic performance problems and cannot easily isolate a single session or statement. If you know what resources are valuable and scarce (usually one and the same), you can use that information to address systemic problems. For example, if the CPU on the system is frequently the bottleneck and you experience systemic performance problems, you want to focus on sessions/statements that are consuming the most CPU. If it is your i/o subsystem, you can look for sessions/statements that are reading data.

### **Throwaway**

This method of sql performance diagnosis looks for inefficiencies in the execution plans. When a step performs a large amount of work, but the results of that work are discarded or 'thrown away', this step is inefficient and a candidate for optimization. When looking at problematic sql, you should not focus on access paths or operations, but on those steps that read a large amount of data that is subsequently thrown away. For additional information, please read Martin Berg's excellent paper at <http://www.miracleas.dk/tools/throwaway2.pdf>

### **Cardinality Feedback**

Cardinality Feedback is a statement diagnostic method that looks at cost based optimizer and the incorrect decisions it can make.

Variance

Root Cause Analysis

5 Whys



# Developing a Routine

- Your Tasks

- Know Your System
- Tool Proficiency
- Question/Answer or Test/Result

- Your Knowledge

- Root Cause Analysis Process
- Oracle Architecture

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 55 of 87\*

This is the basic process for developing routines.

# Diagnostic Routine Components

- Symptom Description
- Question/Test
- Answer/Result
- Decision
- Determination
- Resolution
- Verification

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 56 of 87\*

A common problem is that the initial issue is described as a problem or solution instead of a symptom. It is very important to quickly and accurately gather information. If your initial response to the problem is to kill the system/session/statement, then you cannot gather any information and you are guaranteeing a return of the problem.

The Question/Test, Answer/Result, Decision process needs to be repeated until you arrive at a root cause.

Determination is arriving at the root cause.

Resolution and Verification are the act of fixing the problem and verifying that the fix is proper. While these are not strictly part of the diagnostic routine, they are key components to validating that the routine is working and the problem is actually solved.



# Format

- **What is the symptom being reported**
  - **The first question to ask and the method of answering the question**
    - **Answer 1.1 and Question 1.1**
      - Answer 1.1.1 and method to resolve
      - Answer 1.1.2 and Question 1.1.2
        - Answer 1.1.2.1 and method to resolve
        - Answer 1.1.2.2 and method to resolve
    - **Answer 1.2 and Question 1.2**



## Symptom or Root Cause?

- 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 always valuable, often critical

# Executing the Process

- Accurate Symptom Description
- Gather Information
- Isolate Causes
- Work the Checklist/Decision Tree
- Solve the Problem

March 31, 2011

www.optimaldba.com

Page 59 of 87\*

## Implementing the Process(es)

Once the process has been developed, tested and documented, it is time to use them. These processes are not meant to be static, but rather will change and adapt over time to new releases, additional knowledge, updated tools. It is very important that the process is constantly submitted to critical analysis to look for improvements.

### Accurate Symptom Description

It is very important to accurately describe the symptoms. "Oracle is down", "My session is hung" or "CPU is at 70%" are not very helpful as they are too broad and usually inaccurate. "I am unable to connect to the database" or "This 5 minute report has now been running for 30" offer more information and a good place to start. An inaccurate description can waste valuable time as teams focus on issues that are actually unrelated.

### Gather Information

There is often tremendous pressure to do something, anything, NOW! Unfortunately it often leads to mistakes, ineffective attempts and, worse, a solution that cannot be explained nor repeated. Terminating a session or bouncing an instance may temporarily resolve the issue, but it also prevents you from gathering important information that can help prevent the problem from occurring the next time.

### Isolate Causes

It is important to focus on the likely causes so as not to waste time on extraneous issues. Many systems will have several 'typical' problems, such as blocking locks or inefficient sql, which often cause the diagnostic process to focus only on a small number of causes. While these problems can assist in prioritizing your working list, you also need to know what symptoms preclude the typical problems. For example, if the users are complaining that they cannot log in, you don't want to spend time looking for inefficient sql. You have to be careful to not discard causes too quickly. There are times where the root cause is actually something that appears to be initially unlikely. In one case, a company experienced periodic response time issues and the root cause was a bad router attached to the database server, a cause that was originally discarded and never checked by the network support team.

### Work the List

The possible causes can be prioritized based on experience and system knowledge. Part of the list should also assign tasks to the different teams (DBA, System, Network, Application, etc.) and what order the tasks should be performed. Much of the work can be done concurrently (dba team checks the listener while the network team checks the interface).

### Solve the Problem

Once the root cause has been determined, you need to solve the problem and verify that it has been successful. This is the ultimate metric for diagnosing problems. There are times where you cannot solve the problem, usually because you don't control the problematic component. In this case, it is important to determine possible workarounds.



## Diagnostic Routine Examples

- Just examples...not total coverage
  - Simple examples
  - Not all tests/results/decisions are covered
- Use the examples to build your own

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 60 of 87\*

These examples are just a couple that are easy to demonstrate. They are not complete, but are intended to give you an insight into some basic routines and how to start developing them yourself.



# System Diagnostic Routine

- Knowledge
- Tools
- Process



# Knowledge

- Database/Instance
- Memory and Latches
- Transactions
- DB Time Model
- Operating System/Storage



# Tools

- AWR/Statspack
- ASH
- Statedumps
- Logs/Trace files
- GUI/sqlplus

## Multiple User Complaints

- “sessions are hung”
  - Multiple reports within a few minutes
- Cannot create new sessions
  - Actual error message not returned

This may start as a single session or statement diagnosis. Be prepared to change the routine if you find out that your initial assessment is not correct.

# Is the Instance Up?

- Yes, the required background processes running
  - Can you log in as a regular user?
    - No, you get the error ORA-00257 archiver error. Connect internal only, until freed.
      - Check archive destination and free space if full
      - Compare normal redo generation with current generation
        - Close to normal then check backup/archive log purge processing
        - In excess of normal then check for cause of excess redo generation
- No, the required background processes are not running
  - Check the alert log

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 65 of 87\*

This is an actual situation that has occurred several times.



# Hanganalyze

- Conditions
  - Archiver not stuck
  - Background processes running
- Generate files for Oracle support/analysis
  - Note 452358.1



## 2 Sessions

- Session 1

- hanganalyze
- Wait 1 minute
- hanganalyze

- Session 2

- systemstate dump
- systemstate dump

# Session Diagnostic Routine

- No systemic issue
- Critical to identify session!
  - Instrument application
  - Unique identifiers

March 31, 2011

[www.optimaldba.com](http://www.optimaldba.com)

Page 68 of 87\*

One key to diagnosing session issues is to understand the session's state. And what each of those states means. For example, a query that is spending a lot of time On CPU, the likely cause is a lot of logical i/o. If the event is db file sequential read, then it is doing a lot of physical i/o.

What is My Session Doing?

Response Time = CPU Time + Wait Time

CPU Utilization

- Reading data from memory
- Creating read consistent view
- Other activity

Wait/Event Interface

- Reading data from disk
- Locks, Latches
- Waiting on application or user



# Knowledge

- Locks/enqueues
- Session States




# Tools

- ASH
- Tracing
- Timed Events
- Processstate dump



# Process

- Identify the session
- What is the session doing?
- What should the session be doing?
- What is causing the session not to be doing what it should be doing?



## What are the possible session states?

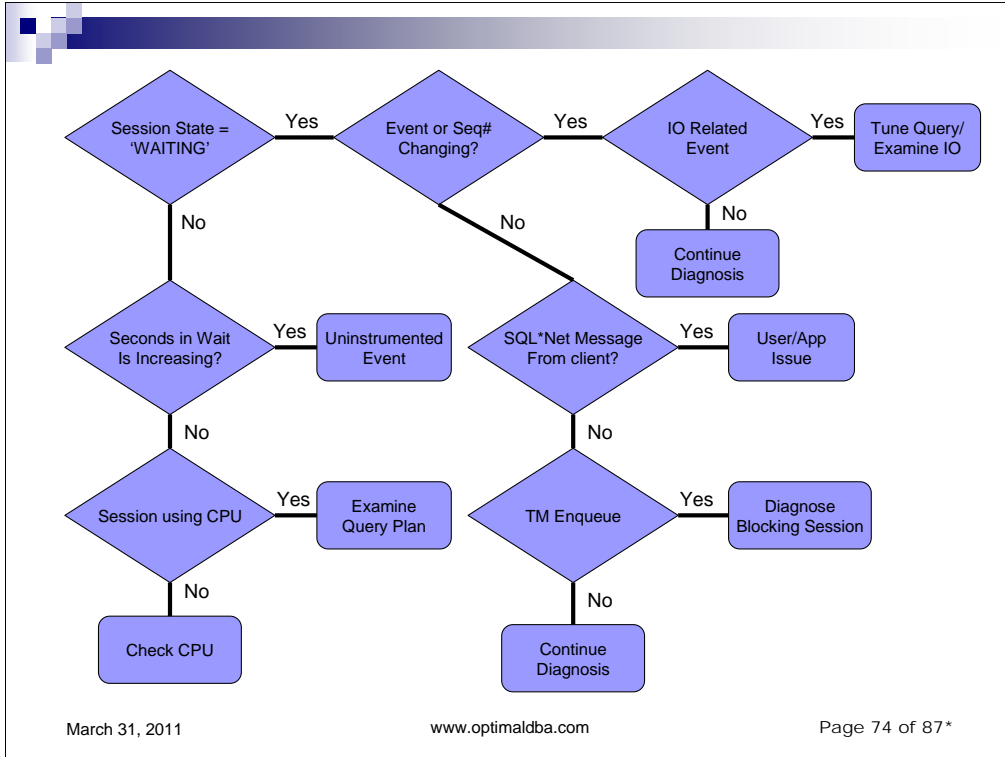
- On CPU
- Waiting on CPU
- Instrumented Event
- Uninstrumented Event

### ■ What do certain events mean?

- SQL\*Net Message
- TM Enqueue
- db file sequential read

# Routine (text)

- **Is the session waiting? (v\$session.state)**
  - **If No ('WAITED...')**
    - **Is v\$session.seconds\_in\_wait increasing?**
      - No - The session is using/waiting on CPU resource
      - Yes - The session has called an uninstrumented event
  - **If Yes ('WAITING')**
    - **Is the wait event (v\$session.event) or sequence (v\$session.seq#) changing?**
      - **If No**
        - Event is "SQL\*Net message from client" – session waiting on application or user for next action
        - Event is "Enqueue" – locate blocking session (v\$session.blocking\_session) and diagnose that session
      - **If Yes**
        - Events are Physical i/o (db file sequential/scattered read) – examine query plan
        - Event is "Read by other session" – examine query plan, look for other high i/o sessions





# Statement Diagnostic Routine

- Throwaway
  - Secondary Throwaway
  - Primary Throwaway
  - Isolate operations that consume the most
    - Time
    - Resource
- Cardinality Feedback



# Tools

- sql trace and tkprof
- dbms\_xplan
- Autotrace
- dbms\_profiler
- Events



# Routine

- What step(s) show the most throwaway?
- Is there an unindexed predicate?
  - Yes – create index for predicate
  - No – Is the predicate construct able to use the index?
    - Yes – diagnose optimization decision
    - No – alter predicate or create function-based index



# Secondary Throwing

- Easy to spot
  - Parent Row Source Operation significantly less than Child RSO
  - Filtering occurs after data read
    - Unindexed Predicate
    - Predicate unable to use index
    - Optimizer choosing not to use index

# Secondary Throwing

- Easy to spot

| Rows     | Row Source Operation        |
|----------|-----------------------------|
| 798      | SORT ORDER BY               |
| 309849   | NESTED LOOPS                |
| 798      | TABLE ACCESS BY INDEX ROWID |
| 798      | INDEX UNIQUE SCAN           |
| 309849   | TABLE ACCESS BY INDEX ROWID |
| 16083334 | INDEX RANGE SCAN            |

16,083,334 -> 309,849  
98.1% Throwing



## Primary Throwing

- Not as easy to spot
- Compare Reads with Rows
- Look at
  - Longest Event(s)
  - Operations Related to the Event

# Call & Event Information

| call    | count | cpu   | elapsed | disk   | query  | current | rows |
|---------|-------|-------|---------|--------|--------|---------|------|
| Parse   | 1     | 0.00  | 0.00    | 0      | 0      | 0       | 0    |
| Execute | 332   | 0.01  | 0.03    | 0      | 0      | 0       | 0    |
| Fetch   | 1138  | 13.15 | 857.26  | 111818 | 363714 | 0       | 806  |
| total   | 1471  | 13.16 | 857.30  | 111818 | 363714 | 0       | 806  |

Elapsed times include waiting on following events:

| Event waited on         | Times<br>Waited | Max. Wait | Total Waited |
|-------------------------|-----------------|-----------|--------------|
| db file sequential read | 111817          | 1.02      | 847.70       |
| read by other session   | 2               | 0.01      | 0.01         |

- Most time spent reading data
- 111,818 disk/physical reads

# Execution Plan

| Rows   | Row Source Operation                                                        |
|--------|-----------------------------------------------------------------------------|
| 25     | NESTED LOOPS (cr=319622 pr=110886 pw=0 time=845897463 us)                   |
| 101584 | MERGE JOIN CARTESIAN (cr=54023 pr=45915 pw=0 time=367527375 us)             |
| 253    | TABLE ACCESS BY INDEX ROWID PR (cr=2406 pr=0 pw=0 time=20183 us)            |
| 1882   | INDEX RANGE SCAN PRPRC_X (cr=999 pr=0 pw=0 time=5333 us)                    |
| 101584 | BUFFER SORT (cr=51617 pr=45915 pw=0 time=367464991 us)                      |
| 99585  | TABLE ACCESS BY INDEX ROWID OPR (cr=51617 pr=45915 pw=0 time=439842435 us)  |
| 99585  | INDEX RANGE SCAN ORPR_CPR_X (cr=1525 pr=772 pw=0 time=7535234 us)           |
| 25     | TABLE ACCESS BY INDEX ROWID ORD (cr=265599 pr=64971 pw=0 time=478493151 us) |
| 58724  | INDEX UNIQUE SCAN XPKORD (cr=203662 pr=28716 pw=0 time=147352908 us)        |

45915 + 64971  
110886 physical reads  
99.17% of statement I/O



# Cardinality Feedback

- #1 CBO issue
  - GIGO
- Look for errors in cardinality
  - E-Rows != A-Rows
  - 1% or 5% cardinality



# Cardinality Feedback

```
select /*+ gather_plan_statistics */
distinct object_type from skew
where owner = 'SYS'; -- or SYSTEM

select *
from table(dbms_xplan.display_cursor(
 format=>'ALLSTATS LAST'));
```

PLAN\_TABLE\_OUTPUT

-----  
SQL\_ID a389555qb4msz, child number 0  
-----

select /\*+ gather\_plan\_statistics \*/ distinct object\_type from skew  
where owner = 'SYS'

Plan hash value: 861362144

-----

| Id  | Operation                   | Name     | E-Rows | A-Rows |
|-----|-----------------------------|----------|--------|--------|
| 0   | SELECT STATEMENT            |          |        | 44     |
| 1   | HASH UNIQUE                 |          | 44     | 44     |
| 2   | TABLE ACCESS BY INDEX ROWID | SKEW     | 32321  | 1059K  |
| * 3 | INDEX RANGE SCAN            | SKEW_IDX | 32321  | 1059K  |

-----

Predicate Information (identified by operation id):

-----  
3 - access("OWNER"='SYS')

PLAN\_TABLE\_OUTPUT

-----  
SQL\_ID 5x29fszd3dcz8, child number 0  
-----

select /\*+ gather\_plan\_statistics \*/ distinct object\_type from skew  
where owner = 'SYSTEM'

Plan hash value: 861362144

-----

| Id  | Operation                   | Name     | E-Rows | A-Rows |
|-----|-----------------------------|----------|--------|--------|
| 0   | SELECT STATEMENT            |          |        | 14     |
| 1   | HASH UNIQUE                 |          | 44     | 14     |
| 2   | TABLE ACCESS BY INDEX ROWID | SKEW     | 32321  | 7406   |
| * 3 | INDEX RANGE SCAN            | SKEW_IDX | 32321  | 7406   |

-----

Predicate Information (identified by operation id):

-----  
3 - access("OWNER"='SYS')



## Questions?

- Paper/Presentation/Scripts are available at [www.optimaldba.com](http://www.optimaldba.com)
- email
  - [daniel.fink@optimaldba.com](mailto:daniel.fink@optimaldba.com)