

Identification of Performance Problems without the Diagnostic Pack

DOAG Webinar, 13 February 2015

Christian Antognini



BASEL BERN BRUGG LAUSANNE ZUERICH DUESSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

■ @ChrisAntognini

Senior principal consultant, trainer and partner at [Trivadis](#)

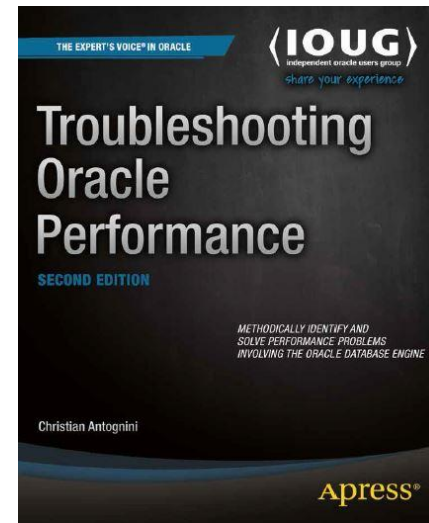
- christian.antognini@trivadis.com
- <http://antognini.ch>

Focus: get the most out of Oracle Database

- Logical and physical database design
- Query optimizer
- Application performance management

Author of *Troubleshooting Oracle Performance* (Apress, 2008/2014)

OakTable Network, Oracle ACE Director



■ Agenda

1. Introduction
2. Analysis of Reproducible Problems
3. Real-Time Analysis of Irreproducible Problems
4. Postmortem Analysis of Irreproducible Problems
5. Third-party Tools

Introduction

■ Disclaimer

The techniques described in this presentation are useful only to identify performance problems that lies in the database layer.

■ Objective of a Performance Analysis

Discover the most time-consuming SQL statements or PL/SQL code invocations.

For each of those time-consuming statements, gather additional information that can help in understanding the problem.

- Execution plan
- Runtime statistics like the number of processed rows and the CPU utilization
- Experienced wait events

■ Basic Questions that Require Answers

Is the problem reproducible at will?

- Yes: everything is much easier than if you can't! → Sec. 2
- No: see next bullet...

For irreproducible problems, is it possible to wait till the problem occurs again?

- Yes: a real-time analysis has to be carried out → Sec. 3
- No: a repository holding historical performance statistics is required → Sec. 4

Analysis of Reproducible Problems

■ Approach

The most efficient way to approach a reproducible problem is to take advantage of one of the available tracing and profiling features to perform a controlled measurement while an application is experiencing the problem.

The analysis starts by tracing the database calls through SQL trace.

- If most of the time is spent executing SQL statements, the trace file(s) contain all the necessary information for a detailed analysis.
- If most of the time is spent executing PL/SQL code, a profiling of the PL/SQL code is needed.

■ Analysis Without Diagnostics Pack

The analysis doesn't depend on the availability of the Diagnostic Pack option. The only feature you could consider to use is Real-time Monitoring.

- Except in 12c, it's useful for single executions only
- Diagnostic and Tuning Pack required

■ Tracing Database Calls

There are a number of ways to enable SQL trace.

- ALTER SESSION
- DBMS_MONITOR
- DBMS_SESSION

When enabled, SQL trace generates trace files containing not only the executed SQL statements, but also in-depth performance figures about their processing.

The trace files have to be analysed with a profiler.

- *TKPROF*
- Third party (e.g. *TVD\$XTAT* and *Method R Profiler*)

■ Profiling PL/SQL Code

The database engine provides two profilers integrated in the PL/SQL engine.

- Call-level profiler (a.k.a. hierarchical profiler): DBMS_HPROF
 - Available as of version 11.1
- Line-level profiler: DBMS_PROFILER

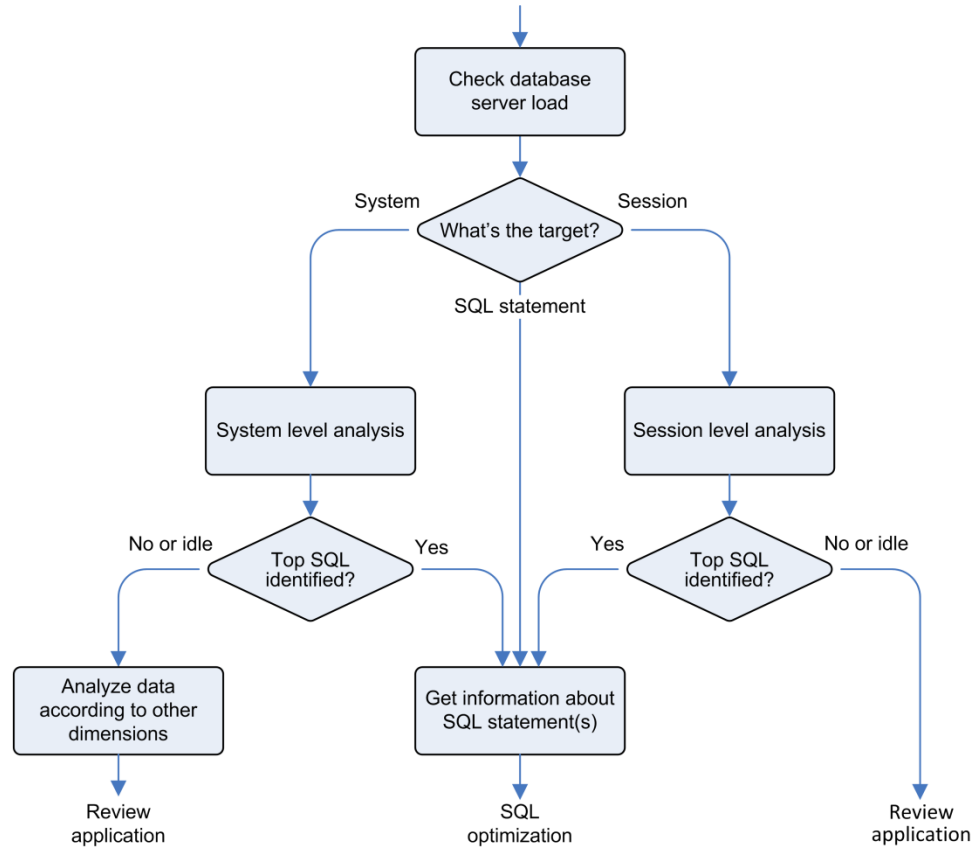
Except if line-level information is needed, the call-level profiler is superior.

The easiest way to use a profiler is to take advantage of the support provided by graphical tools.

- *SQL Developer*
- Third party (e.g. *TOAD* and *PL/SQL Developer*)

Real-Time Analysis of Irreproducible Problems

Approach



■ Dynamic Performance Views Provide the Necessary Information

- OS Statistics
- Time Model Statistics
- Wait Classes and Wait Events
- System and Session Statistics
- Metrics
 - Diagnostic Pack required for metrics history only
- Current Sessions Status
- Active Session History
 - Diagnostic Pack required
- SQL Statement Statistics
- Real-time Monitoring
 - Diagnostic and Tuning Pack required

■ Analysis Without Diagnostics Pack

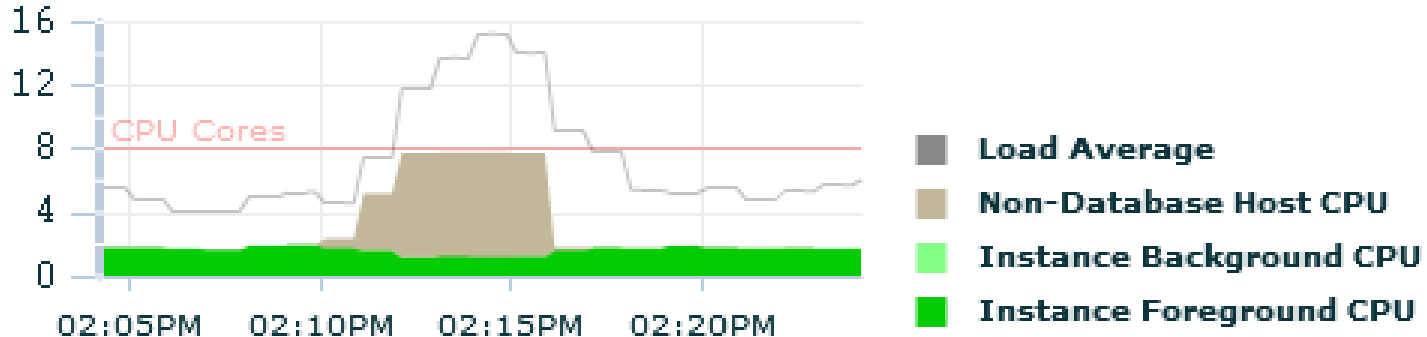
There are two main challenges:

- Enterprise Manager can't be used
 - Third-party tools that provide similar features exist
- Most of the dynamic performance views provide only cumulated statistics
 - Metrics are an exception
 - Utilities that sample the cumulated statistics are needed

This section focuses on a set of scripts that are freely available, so they can be used on any system.

■ Database Server Load

Check not only whether the database server is CPU bound, but also whether there are processes not related to the database instance that consume a lot of CPU time.



■ Database Server Load Based on V\$OSSTAT and V\$METRIC

```
SQL> @host load.sql 16
```

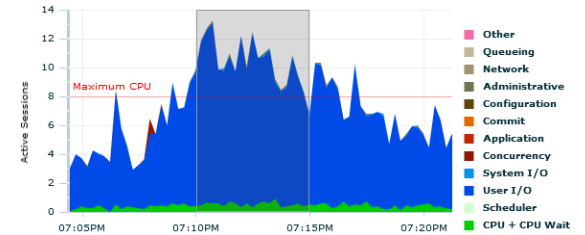
BEGIN_TIME	DURATION	DB_FG_CPU	DB_BG_CPU	NON_DB_CPU	OS_LOAD	NUM_CPU
14:05:00	60.10	1.71	0.03	0.03	4.09	8
14:06:00	60.08	1.62	0.03	0.04	4.13	8
14:07:00	59.10	1.89	0.03	0.04	4.96	8
14:08:00	60.11	1.93	0.03	0.03	5.29	8
14:09:00	60.09	1.73	0.03	0.59	4.60	8
14:10:00	60.10	1.57	0.02	3.64	7.50	8
14:11:00	60.16	1.15	0.02	6.60	11.82	8
14:12:00	60.11	1.21	0.02	6.60	13.77	8

...

■ System Level Analysis

Several steps have to be carried out:

1. Check the average number of active sessions and the portion of time they spent for every wait class
2. Check system-wide time model statistics
3. Check whether few SQL statements are responsible for most of the activity
4. (Optional) Check whether specific sessions/components/... are responsible for most of the activity



■ System Level Load Based on V\$SYS_TIME_MODEL and V\$SYSTEM_WAIT_CLASS

```
SQL> @system activity.sql 15 20
```

Time	AAS	Othr%	Net%	Adm%	Conf%	Comm%	Appl%	Conc%	SysIO%	UshrIO%	CPU%
19:10:11	9.7	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.9	94.8	3.8
19:10:26	10.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	1.0	94.6	3.9
19:10:41	10.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	1.0	94.8	3.8
19:10:56	9.9	0.0	0.0	0.0	0.0	0.4	0.0	0.0	1.0	94.6	4.0
19:11:11	9.8	0.0	0.0	0.0	0.2	1.0	0.0	0.0	1.2	93.7	4.0
19:11:26	9.5	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.9	94.8	3.9

...

■ System Level Time Model Based on V\$SYS_TIME_MODEL

```
SQL> @time model.sql 15 2
```

Time	Statistic	AvgActSess	Activity%
19:14:49	DB time	9.8	98.6
	.DB CPU	0.3	3.4
	.sql execute elapsed time	9.7	97.3
	.PL/SQL execution elapsed time	0.1	1.2
	background elapsed time	0.1	1.4
	.background cpu time	0.0	0.4

```
...
```

■ Top Sessions Based on V\$SYSSTAT, V\$SYS_TIME_MODEL, V\$SESS_TIME_MODEL and V\$SESSION

```
SQL> @active_sessions.sql 15 1 10
```

Time	#Sessions	#Logins	SessionId	User	Program	Activity%
19:14:49	117	0	195	SOE	JDBC Thin Client	1.8
			224	SOE	JDBC Thin Client	1.5
			225	SOE	JDBC Thin Client	1.5
...						
			16	SOE	JDBC Thin Client	1.4
			171	SOE	JDBC Thin Client	1.4
			68	SOE	JDBC Thin Client	1.4
			Top-10 Tot			14.9
...						

■ System Level Analysis with Snapper

Snapper is a script developed by Tanel Põder.

Its key functionality is to sample V\$SESSION.

During the sampling, it checks the status of the specified sessions and, for active sessions, it gathers information about their activity.

It's a very flexible and powerful script that accepts many parameters.

■ System Level Analysis with Snapper – List Top SQL Statements

```
SQL> @snapper.sql ash=sql_id 15 1 all
```

```
-----  
Active% | SQL_ID  
-----  
196% | c13sma6rkr27c  
186% | 8dq0v1mjngj7t  
122% | bymb3ujkr3ubk  
107% | 7hk2m2702ua0g  
82% | 0yas01u2p9ch4  
63% | 8z3542ffmp562  
62% | 0bzhqhhj9mpaa  
30% | 5mddt5kt45rg3  
...
```


■ Session Level Analysis with Snapper – List Top Wait Events

With Snapper it's also possible to target a specific session.

```
SQL> @snapper.sql ash=event 15 1 172
```

```
-----  
Active% | EVENT  
-----
```

```
22% | db file sequential read  
1% | ON CPU  
1% | db file parallel read
```

■ SQL Statement Information

For SQL statements that are responsible for a large part of the activity, more information is needed.

Runtime statistics

- V\$SQLAREA → [sqlarea.sql](#)
- V\$SQL → [sql.sql](#)
- V\$SQLSTATS → [sqlstats.sql](#)

Execution plan → DBMS_XPLAN

Postmortem Analysis of Irreproducible Problems

■ Approach

To analyse a performance problem that happened in the past, a repository containing performance statistics covering the period of time to analyse is required.

Oracle provides two repositories:

- Automatic Workload Repository (AWR)
 - Diagnostic Pack required
- Statspack

■ Automatic Workload Repository vs. Statspack

- AWR is fully integrated and automatically installed
- AWR stores system-level, SQL-level as well as session-level (ASH) data
- AWR is an Enterprise Edition option available as of 10g only
- Enterprise Manager provides a GUI for AWR
- Statspack requires a manual installation
- Statspack stores system-level and SQL-level data
- Statspack is free of charge and available with all editions since 8i
- No Enterprise Manager integration

■ Analysis Without Diagnostics Pack

Almost everything provided by an AWR report is provided by a Statspack report.

- There's no major difference in reading the two reports.

What's really missing is the persisted ASH data.

- Third-party implementations that allow to implement the roadmap discussed in the previous section exist

Third-party Tools

■ Third-party Tools

A number of third-party tools that doesn't require the Diagnostic Pack option exists!

Refer to Kyle Hailey's [Best Oracle Performance Tools](#) list for an overview.

I presently advise to use *Lighty for Oracle*.

- It has a very good price/performance ratio!
- It supports well the approaches for the analysis of irreproducible problems (both in real time and postmortem) described in this presentation and in my book

■ Summary



- It's possible to work without the Diagnostic Pack option
 - Doesn't make things easier, though
- A toolkit is required
 - Scripts and/or a graphical tool
- With and without Diagnostic Pack option it's essential to approach performance problems in a methodological way!

■ References

Troubleshooting Oracle Performance, 2nd Edition, Apress (2014)

<http://antognini.ch/top/>

Kyle Hailey's *Best Oracle Performance Tools* list

<http://datavirtualizer.com/best-oracle-performance-tools/>

The scripts referenced through the presentation can be downloaded from

<http://antognini.ch/downloads/top2/>

Tanel Pöder's Snapper can be downloaded from

<http://blog.tanelpoder.com/files/scripts/snapper.sql>

Questions and answers ...

Christian Antognini

Senior Principal Consultant

christian.antognini@trivadis.com



Trivadis
makes IT
easier.

BASEL BERN BRUGG LAUSANNE ZUERICH DUESSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA