MERITS Profiler Data Sheet

Author: Norbert Debes, ORADBPRO Consulting Services (norbert.debes@oradbpro.com)

June 2009

The MERITS Profiler is the next generation SQL trace analysis tool. It not only generates accurate resource profiles at trace file and SQL or PL/SQL statement level but also has a unique real-time mode that allows it to analyze extended SQL trace files as they are being written by an active database server process and to automatically correlate the information therein with V$ dynamic performance views and the data dictionary. The detection of think time is yet another unique feature of the Profiler. The MERITS Profiler even contains a parser for the SQL language that allows it to recognize and aggregate similar SQL statements.

Response Time Contributor	Duration (s)	Percent (%)	Count	Average (s)
think time	254.821	41.8	265	0.961588
CPU	217.378	35.7	647142	0.000336
SQL*Net message from client	86.142	14.1	270501	0.000318
unknown	41.542	6.8	58	0.716233
log file switch completion	4.568	0.7	18	0.253801
log file switch (checkpoint incomplete)	3.879	0.6	12	0.323286
latch: shared pool	0.525	0.1	252	0.002084
SQL*Net message to client	0.467	0.1	270766	0.000002
log file sync	0.127	0.0	52	0.002450
log buffer space	0.064	0.0	2	0.031800
SQL*Net break/reset to client	0.048	0.0	272	0.000177
latch: library cache	0.045	0.0	87	0.000512
db file sequential read	0.021	0.0	5	0.004222
latch: enqueue hash chains	0.004	0.0	1	0.003633
Total	609.631	100.0	1189433	0.000513

Figure 1: A Resource Profile

SQL Trace vs. AWR, ASH, and Statspack

Why bother enabling SQL trace and analyzing trace files with the MERITS Profiler when there are tools such as AWR, ASH, and Statspack? Although these are very valuable tools, certain classes of problems simply cannot be solved by using them. Say you need to know the response time of a batch job. Only a SQL trace profiler can tell you how long the job ran, what portion of the run-time was spent on the CPU, and what portion was spent waiting. Only a SQL trace profiler can give you a sorted list of *all* the statements the batch job executed, sorted by their contribution to response time. ASH is a sampling tool. SQL trace accurately measures where the response time went and is more accurate than mere sampling. You should not settle for sampling when you can have accurate measurements. ASH reports omit the wait event “SQL*Net message from client” that may contribute very significantly to response time and may need to be reduced by changing application code. AWR and Statspack report instance level statistics. Those tools may obscure performance problems at session level since their data is aggregated at DBMS instance level. A SQL trace file covers a single session[1] with the highest amount of detailed information available. The strength of AWR, ASH, and Statspack lies in the fact that they record performance data and save it in a repository—usually once per hour. Thus you can answer questions like “what were the most expensive statements at a given point in time?”. All three tools cannot answer questions like “what SQL statements did the application execute while an end user who has a reproducible performance problem worked his way through a sequence of interactive screens or web pages”. This is just one example of a scenario where SQL trace excels at measuring response time and finding the root cause of performance problems.

Enriching SQL Trace Data and Real-Time Mode

SQL trace by itself is not sufficient for solving many types of performance problems but it as an excellent starting point. A SQL trace file does not tell you the values of initialization parameters. It also does not tell you which table columns are indexed and how selective they are. This is why the MERITS Profiler uses all available sources of performance diagnostic data to help you solve performance problems quickly and easily. Real-time trace file analysis ensures that ephemeral information in V$ views that may be indispensable in finding the root cause of a performance problem is captured before it becomes unavail-

1. Unless Shared Server is used.

able, e.g. due to being aged out of the shared pool. Of course the Profiler may also be used to analyzes trace files long after the traced process has terminated. This is called offline mode. Yet, even when the traced process has long terminated, real-time mode is available to pull object and system statistics pertaining to the objects in a trace file from the data dictionary[1]. This feature is a huge time saver, since a MERITS Profiler report generated in real-time mode contains the table and index structure as well as cost based optimizer (CBO) statistics for each table and index in the trace file. Thus, a performance analyst can tell right away whether an index is missing or an existing index is not used by the CBO, e.g. because the clustering factor is too high.

The MERITS Profiler vs. TKPROF

Most database administrators are experienced users of TKPROF, Oracle Corporation's own extended SQL trace analysis tool. A MERITS Profiler report by far exceeds the diagnostic expressiveness and the amount of detail provided by TKPROF reports. Knowing where the response time of an application was spent is crucial in solving performance problems. A resource profile is an apportionment of response time in tabular form. Each row of a resource profile contains a single contributor to response time, e.g. CPU time consumption or a wait event. The MERITS Profiler sorts response time contributors in descending order based on their contribution to overall response time. An example of a resource profile is in Figure 1 on page 1.

A MERITS Profiler report contains a resource profile for an entire trace file and for each SQL or PL/SQL statement captured in the trace file. TKPROF does not have the capability to create resource profiles at all. The derived wait event "think time" in Figure 1 is another unique feature of the MERITS Profiler. Each time a database client sends a request to an Oracle database server, it incurs a so-called network round-trip. The duration of a network round-trip is represented by the wait events "SQL*Net message from client" and "SQL*Net message to client". Think time is the portion of response time where the client did not make a request on the database server or put differently where the client was idle from the perspective of the database server. Knowing the contribution of think time to response time is crucial in proving that an alleged database performance problem is in fact not a database problem but an issue in some other layer of the software stack.

MERITS Profiler Design Concept

The novel design concept of the MERITS Profiler is to combine all available sources of performance related information to quickly reveal where response time is spent and to help performance analysts assess potential causes of performance problems. The data sources considered by the Profiler are:

- extended SQL trace files
- V$ dynamic performance views
- data dictionary views
- Statspack snapshots, reports, and the Statspack repository[2]

In real-time mode, execution plans, run-time statistics, and other information is retrieved from V$SQL_PLAN. Peeked bind variables are retrieved using the package DBMS_XPLAN. Object statistics for tables and indexes are also incorporated into Profiler reports. Even the contents of the buffer cache are incorporated into the report to help pinpoint tables that are insufficiently cached.

Aggregation of Non-Reusable SQL Statements

The MERITS Profiler includes a grammar specification of the SQL language. Thus it is capable of parsing SELECT, INSERT, UPDATE, DELETE, and MERGE statements and to decide whether statements that differ only by literal values are semantically identical. Instead of treating each such statement as a different statement, the Profiler aggregates such similar statements and treats them as only one statement. Finally there is a tool that allows analysts to assess the impact of non-reusable SQL statements. Instead of potentially thousands of semantically identical statements that contribute a few milliseconds to response time, a MERITS Profiler report will show the total impact of all similar SQL statements by aggregating all of the individual response times as well as associated wait events.

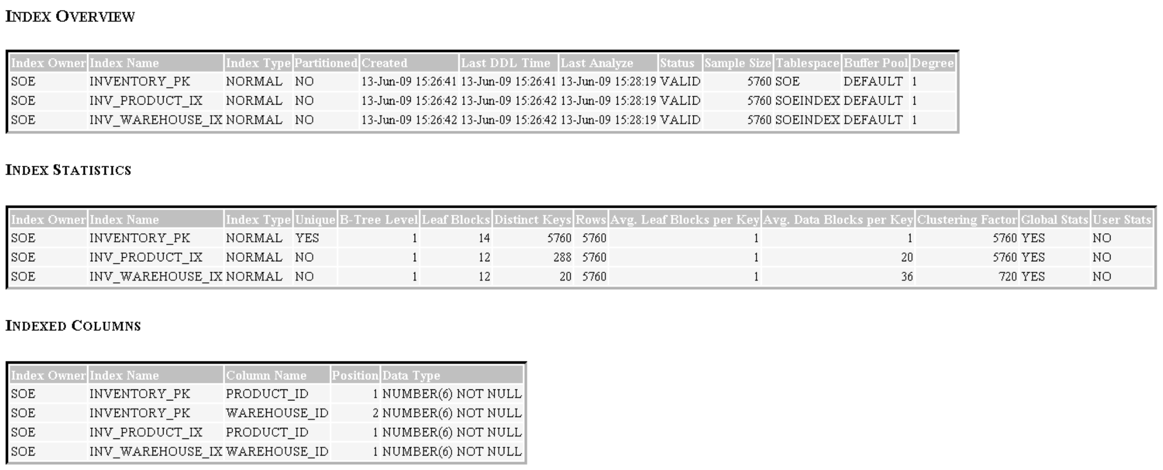
Using the MERITS Profiler

Installing the MERITS Profiler is a snap. Simply unzip the software distribution and adjust the Profiler and Java installation directories in a single command script. Once a user has set up a few properties in his environment, using the profiler is straightforward. Simply determine a session to trace and run:

```
$ mprof session=35.23 properties=user_props.properties
```

1. Assuming the database objects have not been dropped.
2. Feature currently under development; to be released in fall 2009

The preceding command reads user-specific settings from the file user_props.properties, enables SQL trace for the indicated session 35 with serial number 23, and starts reading the trace file in real-time mode. The Profiler is a multi-threaded application, such that the user can interact with the Profiler while a background thread reads and analyzes the trace file. The Profiler indicates whether the background thread has read the entire trace file. The background thread is capable of reading a growing trace file for as long as the user desires. At any time the user is free to interrupt the background process and to generate an HTML or plain text report based on the trace file and potentially other data sources available in real-time mode. A non-interactive mode is available too. In non-interactive real-time mode, the background thread keeps on reading the input trace file until it has not grown anymore within a configurable time-out period, indicating that the database client has disconnected or is idle. Once that happens, the Profiler report is created.



INDEX OVERVIEW

Index Owner	Index Name	Index Type	Partitioned	Created	Last DDL Time	Last Analyze	Status	Sample Size	Tablespace	Buffer Pool	Degree
SOE	INVENTORY_PK	NORMAL	NO	13-Jun-09 15:26:41	13-Jun-09 15:26:41	13-Jun-09 15:28:19	VALID	5760	SOE	DEFAULT	1
SOE	INV_PRODUCT_IX	NORMAL	NO	13-Jun-09 15:26:42	13-Jun-09 15:26:42	13-Jun-09 15:28:19	VALID	5760	SOEINDEX	DEFAULT	1
SOE	INV_WAREHOUSE_IX	NORMAL	NO	13-Jun-09 15:26:42	13-Jun-09 15:26:42	13-Jun-09 15:28:19	VALID	5760	SOEINDEX	DEFAULT	1

INDEX STATISTICS

Index Owner	Index Name	Index Type	Unique	B-Tree Level	Leaf Blocks	Distinct Keys	Rows	Avg. Leaf Blocks per Key	Avg. Data Blocks per Key	Clustering Factor	Global Stats	User Stats
SOE	INVENTORY_PK	NORMAL	YES	1	14	5760	5760	1	1	5760	YES	NO
SOE	INV_PRODUCT_IX	NORMAL	NO	1	12	288	5760	1	20	5760	YES	NO
SOE	INV_WAREHOUSE_IX	NORMAL	NO	1	12	20	5760	1	36	720	YES	NO

INDEXED COLUMNS

Index Owner	Index Name	Column Name	Position	Data Type
SOE	INVENTORY_PK	PRODUCT_ID	1	NUMBER(6) NOT NULL
SOE	INVENTORY_PK	WAREHOUSE_ID	2	NUMBER(6) NOT NULL
SOE	INV_PRODUCT_IX	PRODUCT_ID	1	NUMBER(6) NOT NULL
SOE	INV_WAREHOUSE_IX	WAREHOUSE_ID	1	NUMBER(6) NOT NULL

Figure 2: Index Information

Report Formats and Customization

Both HTML and plain text report file formats are available. The Profiler is highly customizable. The appearance of HTML reports is customizable with cascading style sheet (CSS) attributes. Many other aspects of the Profiler, such as the handling of think time and bind sections are also customizable. The SQL trace level is customizable in the same way as with the well known event 10046. If, for example, you know that you don't need to capture bind variables, the Profiler gives you the choice not to capture them resulting in smaller trace files and faster analysis.

Software Requirements

The Profiler is built with the latest Java technologies. It runs on any platform that supports Java and requires the following components:

- Java 1.6 or newer run-time environment (JRE)
- Oracle JDBC driver

The MERITS Profiler ships with an Oracle release 11.1.0.7 JDBC driver. Hence only a JRE must be installed.