Trace Collier Overview

Norman Dunbar

9th February 2017

Contents

Trace Collier

On of 3rd March, I received a *Cease and Desist* letter from a German lawyer, https://www.wuesthoff.de, acting on behalf of their client, Synaxus. Because of this request, My *TraceMiner2* utility has a new name, *Trace Collier*.

It appears that Synaxus has an unrelated software product with a very similar name to my old name, and have registered it as a trade mark.

Their product can be found at https://www.synaxus.de/index.php/en/traceminer/tm-40 why not take a look?

An improved version of *TraceMiner*. Written in C++ and quite honestly, a better product altogether! Requires an Oracle 10g or upwards trace file, but can *probably* be changed to cope with 9i, at the very minimum, if necessary. Maybe. It depends.

From version 1.06 onwards, Trace Collier has the ability to report the date and time that each execution took place. It does this only if you pass it the name of a trace file which was pre-processed by another of my useful utilities, *TraceAdjust*, available from https://github.com/NormanDunbar/TraceAdjust.

Improvements Over *TraceMiner*.

It *may* not be faster, C++ never usually is, but it's a lot more thorough and safer too. It is less likely to bale out when it can't understand something in a trace file. Also:

- You don't have to reconfigure anything and recompile if you have wider (ie, more digits) cursor IDs on your system. It appears that the cursor IDs, since Oracle 10g, are the address in memory of the cursor, as opposed to a simple incrementing integer. I may be wrong.
- Likewise, there's no recompilation required if you have more than a certain number of bind variables in a statement.
- There is no longer a limit on the length of a bind variable's name.
- If the trace file has a line that is " value= Bind#n" then it is correctly coped with as two lines: a NULL bind value for the current bind variable, and the start of the following bind variable's data. Previously, you had to edit this line in the raw trace file. Not good!
- NULL values for bind variables are processed according to whether they are PL/SQL statement `OUT` variables. For PL/SQL `OUT` variables the variable's name is used as it's value - so you know it's an `OUT` value. For SQL statements, the bind's value must be `NULL`.
- If a bind variable is used in multiple places in the same SQL statement `update some_table set ... created_by = :username, creation_date = :today, last_updated_by = :username, last_update_date = :today, ... ;`, for example, then it is correctly reported in *each* of those places. *TraceMiner*, of old, didn't know what to do when that happened!
- If a bind is a ROWID type (oacdty=11 or 69) then the value line will have no quotes around the ROWID value. The output for the bind in the report will wrap the value in a `CHARTOROWID()` call - just to be explicit. If, however, the bind type is a simple `VARCHAR2`/`CHAR` there can be no wrapping, so there isn't!
- If a PL/SQL statement returns a ref_cursor (data type = 102), it's not possible to get a name or value for that particular bind as it simply points at an address in memory. In this situation, Trace Collier will replace the bind 'value' with the text "REF_CURSOR" - just so you know what's what.
- The log file and verbose output debugging file are created in the same directory as the trace file. No more messing about with redirection etc.
- The default report format is HTML. Plain text output is always going to be messy, as it replicates the input SQL statements, line feeds and all, which can be very messy indeed according to the developer or code generator that created them.

- The report file now reports the `EXEC`, `PARSE`, `BINDS` lines as well as the line in the trace file where the SQL was first located.
- `COMMIT` and/or `ROLLBACK` statements indicate whether or not data was actually changed. If you see a "COMMIT (Read Only)" in the report, you may be committing unnecessarily as the entire transaction made no changes.
- The report file repeats the headings every 25 EXECs by default. This makes the need to scrolling back and forth to find out what the numbers in the various columns actually refer to, a lot less necessary! This can be defined with a command line argument `--pagesize=nn` or `-p=nn`. No spaces are permitted around the '=' sign.
- A `TraceCollier.css` file and a `favicon.ico` file are created in the same location as the trace file too. The CSS can be edited quite easily to match your installation standards - if you have any - or just to make things the way you prefer them. For some reason, Internet Explorer 11 completely ignores the icon file.
- If a css or icon file, as above, are detected in the trace file's folder, then they are not overwritten or changed in anyway. This way any changes you make are preserved.
- If running in verbose mode (`--verbose` or `-v`), the debugging output is much more readable, and much more verbose! If Trace Collier fails to parse a trace file, try running in verbose mode and see what gets output to the debug file.
- You can specify the maximum depth for cursors to be reported. The default is zero. (`--depth=n` or `-d=n`) Only cursors with a lower or equal depth will be reported. If you use `--depth=3` then only cursors with depth 0, 1 2 or 3 will be reported on.

Free eBook - Oracle Trace File Explained

You can always download the latest, free, copy of my eBook on Oracle Trace files at https://github.com/NormanDunbar/OracleTraceFilesExplained/releases.

Get the Latest Binary

Go to https://github.com/NormanDunbar/TraceCollier/releases, choose the latest release, and then choose your desired version. Files here will be for Windows or Linux in 32 and 64 bit versions. The source code used to build these versions is also available to download from the release that you are looking at - should you require it.

Download the Source

Go to https://github.com/NormanDunbar/TraceCollier and click the clone or download button. Choose the option to download a zip file.

Save it somewhere safe, I use my `SourceCode` folder, and extract it. This will create a new folder named `TraceCollier-master` - you can remove the `-master` bit if you wish, however, the following instructions assume that you didn't.

Compile the Source

On Linux

Use the Makefile This option is known to work on GNU/Linux and also, surprisingly, on AIX too. At least one of my "customers" is using Trace Collier on AIX.

```
cd TraceCollier-master
make -f makefile.gnu
```

CodeBlocks IDE There is a project file in the `SourceCode/TraceCollier-master/` folder, named `TraceCollieer.Linux.cbp`. Open that and select `Build->Build` or press CTRL-F9 to do the same. The executable will be found in `SourceCode/TraceCollier-master/bin/ReleaseXX` when it has completed. ('XX' is 32 or 64, depending on which build you chose.)

The build options in the project file allow you to compile Debug and/or Release versions for 32 and/or 64 bit Linux using the g++ compiler.

Also on Linux, there are a number of small shell scripts named `compile*.sh` and `rebuild*.sh` which can be used to build various targets from the command line, using the CodeBlocks project file as a "make" file. Most useful indeed.

On Windows

Visual C++/Visual Studio

Just don't! Trust me, you don't need the grief. It's not like Trace Collier uses anything non-standard, just plain old C++ and the Standard Template Library, for maximum portability. But then again, I did mention standards didn't I? That's a problem with Microsoft.

Honestly, try it if you want to. I haven't. *TraceMiner* of old needed so many changes to basic C code just to make it even think about compiling with Visual Studio, it's just not worth it.

Plus, I haven't even tried! :-)

Borland 10.1 compiler

```
cd TraceCollier-master
bcc32c -o bin\TraceCollier.exe TraceCollier\*.cpp
```

CodeBlocks IDE There is a project file in the `SourceCode\TraceCollier-master` folder, named `TraceCollier.cbp`. Open that and select `Build->Build` or press CTRL-F9 to do the same. The executable will be found in `SourceCode\TraceCollier-master\bin\ReleaseXX` when it has completed. (XX is 32 or 64 depending on your chosen build target.)

The build options in the project file allow you to compile Debug and/or Release versions for 32 and/or 64 bit Linux using the TDM 64/32 bit version of the g++ compiler for Windows.

Also on Windows, there are a number of batch files named `compile*.cmd` and `rebuild*.cmd` which can be used to build various targets from the command line, using the CodeBlocks project file as a "make" file.

Need a Free C++ Compiler?

https://www.embarcadero.com/free-tools is the place to look for one. It's only available in the 32 bit version, but it is modern and runs perfectly well on 64 bit Windows too.

You will need to sign up, but other than a few special offers, and a couple of training course advisory emails, you won't get too much hassle. And it's worth it for one of the finest Windows C/C++ compilers for free.

- Download it.
- Unzip it.
- Add the bin folder to your path.
- Use it and love it! ;-)

Command Line Options

Executing Trace Collier is easy. At a minimum it requires a single parameter, the name of a trace file. This can be a relative or absolute path as desired. The trace file is expected to have a `.trc` extension.

```
TraceCollier [options] trace_file
```

Possible options are:

- `--help` or `-h` or `-?` which provides brief help about the application and its options, then exits without doing anything.

- `--pagesize=nn` or `-p=nn` which sets the report page size to `nn` EXEC statements. This throws a couple of blank lines and repeats the headings in a text format report, or, starts a new table with appropriate headings in an HTML report. In either case, it makes reading easier. The default is 25 and this works well using Firefox on Windows 7, or Linux with a decent sized monitor. Even Internet Explorer copes!

- `--text` or `-t` which forces the report file to be created in plain text mode. The default is to create the report in HTML format.

- `--verbose` or `-v` which creates a debugging file that will contain a huge amount of debugging information. If you have problems with Trace Collier then this file will help me debug things. It's best, really, that you don't run the application in this mode unless absolutely necessary! You have been warned. :-)

- `--quiet` or `-q` will turn off all the `Cursor: #cccccc created at line nnnn` messages. Any `ERROR #ccccc` or `PARSE ERROR #cccccc` lines, and feedback lines will still be reported though. You can't turn those off.

- `--depth=n` or `-d=n` which determines which cursors will be examined and reported on. Any cursor with a `PARSE`, `EXEC` or `CLOSE`, with a `dep=` value less than or equal to the supplied

`--depth=n` will be reported. Anything with a `dep=` value greater than the requested depth will be ignored.

- `--feedback=n` or `-f=n` determines how often you get feedback about the progress of reading the trace file. The default is every 100,000 lines. Use a feedback of zero to disable feedback. Feedback is not disabled with the `--quiet` or `-q` option.

Trace Collier will create:

- A report file, the default is in HTML format, which is the same name as the trace file, but with the extension changed from `.trc` to `.html`.
- If the report is in HTML format, then `favicon.ico` will be created, *if one doesn't already exist* in the folder the trace file is found in.
- If the report is in HTML format, then `TraceCollier.css` will be created, *if one doesn't already exist* in the folder the trace file is found in. This file allows you to style the HTML report as per your company standards (well, up to a point) or to your preference.
- A debugging file. If and only if running in verbose mode. This will have the same name as the trace file used for input, but with the extension changed from `.trc` to `.dbg`.

Testing

Trace Collier has been tested on the following:

Compiled using Borland C++ 10.1:

- Windows 7.
- Windows Server 2012.

Compiled with G++ 4.8.5:

- Oracle Enterprise Linux 7.2
- Oracle Enterprise Linux 7.3
- CentOS Enterprise Linux 7.2

Please note: Trace Collier tries to use the Standard Template Library's REGEX facilities. However, these do not work in versions of the STL supplied with `g`++ at versions less than 4.9.0. there is a check in the source code (in `gnu.h`) to determine if the compiler works or doesn't, and if not, uses plain old scanning for the required data - such as cursor IDs, depth, lengths of SQL statements etc.

Compiled with G++ 5.x.x:

- Linux Mint 18.

Documentation

In development, `doxygen` version 1.8.15 was used. Versions previous to this *may* cause problems. Time and testing will tell.

The source code is documented with specially formatted comments which are collected by *Doxygen* http://www.doxygen.nl/index.html, and output as HTML files documenting all the source and header files, classes, variables, functions etc in the source code. Other output formats are available, but HTML is the default.

This is not User documentation, but Developer documentation.

There is a Doxygen configuration file included in the source code. It can be found in the folder `SourceCode\TraceCollier-master\Docs` and is named `TraceCollier.doxyfile`.

Sadly this file doesn't appear to enable relative paths to be used when extracting from the source files, so if you desire to generate the documentation, you will have to edit the document in a number of places:

Option	Current Value
PROJECT_LOGO	E:/SourceCode/TraceCollier/Docs/tm2_logo_25pct.png
OUTPUT_DIRECTORY	E:/SourceCode/TraceCollier/Docs
WARN_LOGFILE	E:/SourceCode/TraceCollier/TraceCollier/Docs/doxygen.log
INPUT	E:/SourceCode/TraceCollier/TraceCollier/
HAVE_DOT	YES

Replace `E:/SourceCode/TraceCollier` with your appropriate top level directory.

The configuration file for Doxygen assumes the presence of the `dot` (aka `Graphviz` http://www.graphviz.org/) utility to draw the class, call and caller diagrams. If you don't have dot, or do not wish to generate these diagrams, then set `HAVE_DOT` to `NO`.

Once you have edited - and saved - the configuration file, run it as follows (assuming you have Doxygen installed and on your path):

```
doxygen /path/to/config/file
```

for example:

```
doxygen C:/SourceCode/TraceCollier/Docs/TraceCollier.doxyfile
```

The HTML folder, found in the location you set for `OUTPUT_DIRECTORY` above, will contain the `index.html` file that starts the documentation proper.

If you have installed Doxygen, then there is a Wizard that you can run to load the configuration file and generate the documentation. On Windows 7, it is found under `start`->`all programs`->`doxygen`->`Doxywizard`.

Figure 1: The Trace Collier!

Free eBook Describing Oracle Trace Files

You can obtain my free eBook on Oracle Trace Files from https://github.com/NormanDunbar/OracleTraceFilesExplained/releases.

Image Credits

The miner image used in Trace Collier's logo - which you can see when you generate the documentation - is used with gratitude. I obtained it from http://www.clipartpanda.com/categories/miner-clipart.

Pandoc

The PDF version of the `README.md` (markdown) file was created using `pandoc`, which you can obtain and use for free from http://pandoc.org/. It will convert almost any source file into almost any output file.

The PDF was created thus:

```
export PANDOC_PDF=README.pdf
export PANDOC_COLOUR="Cool Grey"

pandoc --from markdown \
--to latex \
--output "${PANDOC_PDF}" \
--table-of-contents \
--toc-depth=3 \
--listings \
--include-in-header listings_setup.tex \
--variable fontfamily="utopia" \
--variable toccolor="${PANDOC_COLOUR}" \
--variable linkcolor="${PANDOC_COLOUR}" \
--variable urlcolor="${PANDOC_COLOUR}" \
--variable margin-top=3cm \
--variable margin-left=3cm \
--variable margin-right=3cm \
--variable margin-bottom=4cm \
README.md
```

Other, useful, output formats are `epub`, Word `docx`, Libre Office `odt`, `ReStructuredText`, `HTML` etc etc etc.