DBCollect/Security

< DBCollect

DBCollect safety, security and confidentiality guide

Contents

Purpose

The purpose of *dbcollect* is to provide an easy, standardized, automated way to collect (Oracle) database workload and configuration information, as well as Unix/Linux performance and configuration statistics.

The collected data is used for:

- Database platform sizings
- Performance analysis and troubleshooting
- Calculation of possible TCO reduction / system efficiency
- Insight in understanding the technical architecture of customer environments

dbcollect is just one part of a database workload assessment platform - but the only component that needs to be executed at customer environments. The other parts run at a dedicated lab environment outside of production environments.

As *dbcollect* is intended to run on production database servers, it must be safe to use and may not cause any issues to these servers and/or databases. Also important is to completely understand what data is collected and how it is transferred in a secure way for further analysis.

Author

dbcollect is written by Bart Sjerps as a private Open Source project and in no way funded or guided by his employer. The project is hosted on Github.com, and contributions (improvement on code, problem reports, documentation) are welcome. It is licensed under GNU Public License version 3 - which means everyone can use (or modify) dbcollect as they wish, under the GPL v3 conditions.

Safety

As dbcollect typically runs on production database systems, it has been designed to be failsafe where possible. This means that bugs or unexpected behaviour can not lead to data corruption or outages. To achieve this, the following design principles have been applied:

- dbcollect does not run as "root".

  Even if it is executed as root by the administrator, the first thing dbcollect performs is a switch to a different user. As such, on typical systems, dbcollect cannot modify Operating System files, memory, kernel settings etc.[1]

- The user under which dbcollect runs should be the Oracle (database) "SYS" owner (usually the user 'oracle').

  The reason for this is that dbcollect needs to execute SQL scripts with "sysdba" privileges without having to enter passwords for every database instance. As this user itself has full access to database data and Oracle binaries, additional steps have been taken to restrict access (see below)[2]

- Most file write operations are forced to only happen in the temporary directory ('/tmp') [3]
- All Oracle database operations are performed using Oracle SQL*Plus and can only perform database 'SELECT' statements.

  No items like views, procedures, functions, directories are created in the database. No data can be directly modified. [4]

- Some database audit events will be generated as _dbcollect_ makes SQL*Plus connections
- OS level commands are limited to those that do not require 'root' access and cannot modify configurations, but only collect configuration parameters.
- OS level commands that require 'root' can be used via _sudo_, for this, _dbcollect_ can be run as root with the ```--sudoers``` option, which will make it write a file named `/etc/sudoers.d/dbcollect` providing limited access to some OS commands for users member of the `dba` group. Using this is optional (but highly recommended on HP-UX

as it allows _dbcollect_ to get crucial CPU and disk information).

- The _dbcollect_ Python code is frequently verified with 'pylint' to detect and remediate potential bugs and issues.
- _dbcollect_ only runs one command or SQL script at a time, except when generating AWR or Statspack reports, then the default is limited to a maximum 50% of available CPUs unless changed with the --tasks parameter.
- As '/tmp' on most Unix/Linux systems is a separate file system and the only place where files are written, the system cannot become unstable due to filling up other file systems to 100% capacity.
- The temporary files in /tmp are cleaned up when dbcollect ends (even if there are errors, exception is if it is killed with SIGKILL)
- The only function in the tool that accesses external (internet) sites is the ```--update``` function which is contained in one simple Python module (```updater.py```) and is only used for updating _dbcollect_ itself, and is restricted to hard-coded URLs on the dbcollect repository on ```github.com```.

Notes:

1. This assumes the system has typical security settings where regular users don't have write access to OS files 2. Using another user would require the administrator to enter SYS passwords for each database instance - which would not make the tool more secure, only harder to work with 3. The output ZIP file can be created with a different name, using the ```--filename``` option, but aborts if the file already exists (cannot overwrite). The temporary directory can be changed with ```--tempdir``` but tempfiles are created in a subfolder and cannot overwrite existing files 4. The database logon itself as well as accessing Oracle AWR reports and other tables generates some audit logging in the database

Source code verification

- As _dbcollect_ is 100% open source, everyone with some Python and Oracle knowledge can verify what the tool can do. There are no hidden features.
- The output ZIP file is stored on the local system and not sent automatically to any external location. This needs to be done by the system administrator.
- The ZIP file contains regular text, html and other open file formats (such as Linux SAR) which are completely transparent. Administrators can inspect the ZIP file to verify its contents before sending it
- [Github releases](https://docs.github.com/en/github/administering-a-repository/about-releases) is the current mechanism to securely distribute the _dbcollect_ package. Git/github tools are available to users to verify all code change history. Nobody except the author has access to the repository for publishing code changes and new releases.
- The _dbcollect_ package is a [Python zipapp](https://docs.python.org/3/library/zipapp.html#the-python-zip-application-archive-format) package. This is not a binary format but just a ZIP file containing the Python (and other) files that can be executed directly. You can unzip the _dbcollect_ package using standard unzip: ```unzip dbcollect``` to inspect the Python code.

- All SQL*Plus scripts are located in the package's ```sql``` directory for further inspection.

Collected data

dbcollect does not collect any end user data, passwords, encryption keys, or other security sensitive information from either Oracle databases or the operating system. Items that are collected are:

- Oracle AWR or Statspack reports
- Additional Oracle information such as table and disk/diskgroup sizes, versions, compression settings etc.
- CPU, memory, network interface and disk settings
- Kernel configuration, installed packages, file systems, processes, vendor strings
- SAR (System Activity Report)
- AIX/Solaris: Partition and zone info

For exact details on what data is collected, check the following files/packages:

1. syscollect.py (OS and SAR data) 2. config.py (list of OS commands to run on each OS to collect system info) 3. awr_report.sql (AWR reports) or sp_report.sql (Statspack reports) 4. database.sql, instance.sql, dbinfo.sql, pdbinfo.sql (Additional database information)

Database / instance detection

The purpose of this tool is to make it as easy as possible to grab Oracle data. As such, it tries to find all Oracle instances on a host automatically. Earlier versions relied on ```oratab``` for this but it turned out many customers run Oracle instances not listed in oratab. So two separate methods are used:

1. Classic /etc/oratab or /var/opt/oracle/oratab parsing. This often fails to detect instances on RAC with newer Oracle versions as it lists the database/service name instead of the instance. 2. Using the Oracle inventory (inventory.xml) to find all possible ```ORACLE_HOMEs``` and look for ```hc_<sid>.dat``` files (running instances will always create such a file). 3. For both oratab and inventory reported instances, the Unix process list is checked to see if the instance is actually running before attempting to connect 4. Multiple entries with the same instance - but sometimes different ORACLE_HOME can be detected. The one with the most recent ```hc_<sid>.dat``` will be used.

SQL query text

AWR and Statspack reports do not contain actual table data. They do usually show the actual SQL statements of active queries. _dbcollect_ can strip AWR reports from SQL code and replace these with a 'removed' message (using the ```--strip``` option). Notes:

- This can only be done for AWR (html) reports, not for Statspack (as these are plain text and no good way exists of parsing the data)
- Parsing html requires the optional Python package lxml, or, if lxml is not installed, the default xml package (slower but always available)
- Stripping AWR reports can be enabled using the ```--strip``` option if it is preferred that the AWR reports contain no SQL code

- If stripping fails (due to errors when parsing html) an error is reported and the file remains unchanged

Execution time

The time it takes to run _dbcollect_ largely depends on:

- How many AWR/Statspack reports are generated (based on AWR retention, interval, amount of instances)
- If databases are single-instance or Oracle RAC (generating AWR on RAC is slower and more reports are generated)
- Whether or not certain known performance issues with Oracle AWR reporting are fixed (depends on Oracle version and patch level)

Expect a runtime between a few minutes (single instance, one database, standard retention, normal 10-day collect period) and many hours or even days on systems with large amounts of database instances, long retention, short AWR intervals and long collection period. For such systems, running in a ```screen``` or ```tmux``` session is recommended so you can disconnect while it is running.

ZIP and TEMP size

The resulting ZIP file size is usually between a few megabytes (system without Oracle, just OS reports), a few hundred MB (system with a few instances, normal retention/interval) and several gigabytes (large system with many instances and/or short AWR retention).

HTML formatted AWR reports vary in size between less than a megabyte and several megabytes (up to 10MB is not uncommon). Compression of HTML reports with ZIP usually achieves about 1:10 compression ratio. Calculation of size requirements:

Size of metadata + some text files: very small (ignored)

SAR files (Linux binary): depends on interval and amount of devices but usually about 31 days * 2-3 MB - usually less than 100MB, worst case 1-2 GB

AWR files: Daily amount of AWRs for a single day, one instance = 24 * 60/interval. Default interval is 1 hour = 24 AWRs per day. With Oracle RAC, multiply by number of nodes.

Total AWR size (per instance): Collected days (default 10) * daily amount of AWRs * average AWR size = 10 * 24 * 2 = 480MB

A Linux system with one instance, no RAC, default settings and default collection period will generate roughly a 100 MB ZIP file.

The TEMP file system size should be at least the size of the ZIP file plus a few hundred MB.

Transferring and processing results

To transfer the ZIP files a secure upload link will be provided by the author - or alternatively use other methods (such as corporate secure FTP transfers).

Confidentiality

The collected data will under no circumstances be provided to any other person or organization except with explicit permission. The resulting graphs and reports can sometimes be used for demonstration purposes (system and database names will be anonymized where possible).

Copyright and disclaimer

dbcollect is published under the GPL v3+ license, for details refer to [gpl v3](http://www.gnu.org/licenses/gpl-3.0.html)

Notes

1. This assumes the system has typical security settings where regular users don't have write access to OS files
2. Using another user would require the administrator to enter SYS passwords for each database instance - which would not make the tool more secure, only harder to work with
3. The output ZIP file can be created with a different name, using the `--filename` option, but aborts if the file already exists (cannot overwrite). The temporary directory can be changed with `--tempdir` but tempfiles are created in a subfolder and cannot overwrite existing files
4. The database logon itself as well as accessing Oracle AWR reports and other tables generates some audit logging in the database

---

Last edited on 10 October 2024, at 16:05.