Greenplum Database gpsupport Utility

Rev: A04 (gpsupport v1.2)

Updated: May, 2017

Note: The `gpsupport` utility is deprecated and will be removed in a future release.

Overview

You can use the `gpsupport` utility to collect Greenplum Database information when troubleshooting and diagnosing issues with Greenplum Database installations. The utility works on 4.2.x and 4.3.x Greenplum Database systems that are running on the Linux operating system. The utility can be run from an interactive shell that supports auto-complete. You can also run the `gpsupport` utility as a command from a BASH environment.

Installing and Running gpsupport

To use the `gpsupport` utility, download the binary file from *Pivotal Network* and copy it to the Greenplum Database master host. The file must have execute privilege on the master host. When you enter `gpsupport` from a command line window, the utility starts the `gpsupport` interactive shell. From the `gpsupport` shell, enter `help` to see the available options.

Some `gpsupport` commands, such as collecting logs from Greenplum Database segment hosts, require access to the hosts. When access to a segment host is required, the `gpsupport` utility uses SSH to connect to the remote hosts and installs the `gpsupport_segment` helper utility on the host, if needed. If a valid helper utility is already installed on the host, that helper utility is used. The default helper utility installation location is `/tmp`. To specify a different location for the helper utility, specify the location with the `segmentFileDir` option. The user must have write and execute privileges on the specified directory on the host systems.

The following examples use the `hosts` file that contains the names of the Greenplum Database hosts and installs the helper utility in the directory `/home/gpadmin/node_collector`. This example installs the helper utility from the `gpsupport` utility shell.

```
> hostfile ~/hosts collect logs segmentFileDir /home/gpadmin/node_collector
```

To exit the `gpsupport` utility shell, enter the `quit` command.

This command runs the same `gpsupport` command from a command line window:

```
$ gpsupport hostfile ~/hosts collect logs segmentFileDir /home/gpadmin/
node_collector
```

Summary

```
gpsupport { [hostfile host-file ]
          [dbHost master-host] [dbPort port ]
          [dbUser user-name] [dbName DB-name ]
          [remoteUser remote-user]
          [startDate start-date ] [endDate end-date]
```

```
        [singleDate date ]
        [outputDir path-to-dir]
        [verbose] [help] [quit] }

  {collect logs [failedSegs] [masterDataDir path-to-data-dir]
        [segDataDir path-to-dir]
        [segs list-of-hosts] [segmentFileDir path-to-dir]
        [anonymize] } |

  {collect queries [logDate date] [logDirpath]
        [logFile path] [outputFile path-to-output]} |

  {collect schemas [shemaName schema] [noGlobals]
        [noSchema] [noStats] [noTupleCount]
        [outputFile path-to-output] } |

  {collect core coreFile path-to-core-file
        coreHost host-name [binary name] } |

  {diagnose backup } |

  {diagnose connectivity hostfile host-file
       [srpc srpc-port] [sudp sudp-port]
       [rudp rudp-port] [nr processes]
       [to seconds] [mtu 1500 | 9000 [outputFile path-to-output] } |

  {diagnose system {host host_ID | local }
       [stdout | tarOut] [config config_file] } |

  {diagnose system tarIn=diagnose_gz_file}
```

Description

Collate files from segments onto the master, parse information from logs and system files, extract metadata from the database.

These commands can be run from the `gpsupport` shell:

- `aotable` - Check and optionally repair append optimized table for data corruption.
- `collect logs` - Retrieves and collates logs from a Greenplum Database system.
- `collect queries` - Extracts previously run queries from a Greenplum Database log file and outputs them to an SQL file on the master host.
- `collect schemas` - Dumps schemas, additional statistics, configurations, and metadata from a running Greenplum Database instance to an SQL file on the master host.
- `collect core` - Collects Greenplum Database core dump files and associated libraries.
- `diagnose backup` - Tests Greenplum Database catalog prior to running a backup. Run queries against the Greenplum Database catalog to simulate performing a backup.
- `diagnose connectivity` - Tests connections to Greenplum Database hosts using UDP. Displays network connection information and statistics including round trip time, packet lost etc.
- `diagnose system` - Validates various platform-specific configuration settings on the specified Greenplum Database hosts.

Options

dbHost *host* dbPort *port* dbUser *user* dbName *database*

Greenplum Database credentials. By default, `gpsupport` uses the `PGHOST`, `PGPORT`, `PGUSER`, and `PGDATABASE` environment variables to connect to a Greenplum Database

instance. Specify these options to override the respective environment variables for a specific command.

endDate *date*

Specifies the end date of a date range. The *date* is in the format *YYYY-MM-DD*. This option can be used with `startDate`. The default is the current date.

hostfile *path-to-hostfile*

The specified file contains the names Greenplum Database hosts in the cluster. In the file, each host name is on a separate line with no extraneous white space.

For the `collect logs` command, either this option or the `segs` option is required.

startDate *date*

Specifies the start date of a date range. The *date* is in the format *YYYY-MM-DD*. Any commands that operate on a specific date range, such as `collect logs` or `collect queries`, can start with `startDate`. If `startDate` is not set, the command works with all Greenplum Database data. This option is ignored if it is not valid for a command.

singleDate *date*

Specify a single date in the format *YYYY-MM-DD*. This option overrides the `startDate` and `endDate` options. For any commands that operate on a specific date range, such as `collect logs` or `collect queries`, you can specify a single date with `singleDate`. This option is ignored if it is not valid for a command.

remoteUser *username*

Specify this option if a user name is required to access the segments that is different from the *username* of the user running the `gpsupport` utility.

segmentFileDir *path-to-dir*

The `gpsupport_`segment utility, a helper utility used in log collection, will be copied to this path on each segment. Thee user must have write permissions to the specified directory. The default directory is `/tmp`.

The log files from segment hosts are uploaded to this directory on the master host. The directory must exist on the master host. The default directory is `/tmp`.

outputDir *path-to-dir*

Specify a working directory.

verbose

Output more detailed logging and debugging messages if available.

quit

Exit from the `gpsupport` utility shell.

collect logs

The `collect logs` command retrieves and collates logs from a Greenplum Database system.

Either the `hostfile` option or the `segs` option must be specified.

These are options for `collect logs`:

anonymize

In the collected log files, redacts the IP addresses of the hosts from which logs are collected. The IP address is replaced with the string `#.#.#.#`. Other IP addresses in the log files, such as addresses used in a query, or addresses in `ps_aux.out`, are not redacted.

failedSegs

Includes logs only for failed segment instances in the log tar file. The default is to not collect logs from failed segments.

masterDataDir *path-to-dir*

The log files from segment hosts are uploaded to this directory on the master host. The directory must exist on the master host. The default directory is `/tmp`.

segDataDir *path-to-seg-dir*

A directory on all segment hosts. The specified directory is created if it does not exist. Log files from the segments are copied to this directory before they are uploaded to the master host. The files are deleted from the segment hosts after being uploaded. The default directory is `/tmp`.

segs *list-of-hosts*

Comma delimited list of segment hosts. This option overrides the `hostfile` option that specifies a file that lists the segment hosts.

collect queries

The `collect queries` command extracts previously run queries from the Greenplum Database log files and writes them to a SQL file.

You can process log files from a specific date or date range by specifying the options `singleDate`, or `startDate` and `endDate`.

These are options for `collect queries`:

logDir *path-to-log-dir*

If neither `logDate` or `logFile` are set, all log files in the specified directory are processed. Otherwise, relative paths passed to `logDate` or `logFile` are resolved relative to this directory.

logFile *path-to-log-file*

Queries from the specified log file will be extracted.

This option cannot be used in conjunction with `singleDate`, or `startDate` and `endDate`.

outputFile *path-to-output-file*

Extracted queries will be written to the specified output file. If the file exists, it is overwritten. The default file is the `gp_extracted_queries.sql` file in the current directory.

collect schema

The `collect schema` command writes schemas from a running Greenplum Database instance, as well as additional statistics, configurations, and metadata, to an SQL file.

These are options for `collect schema`:

noGlobals

Do not dump global objects, such as roles and permissions.

noSchema

Do not dump the database schema.

noStats

Do not dump database statistics from the *pg_statistic* catalog table.

noTupleCount

Do not dump planner statistics from the `reltuples` and `relpages` columns of *pg_class* catalog table.

outputFile *path-to-output-file*

Extracted information is written to the specified file. If the file already exists, it will be overwritten. The default file is the `gp_schema_dump.sql` file in the current directory.

schemaName *schema*

Dump information for the specified schema and global objects. To exclude the global object information, specify the `noGlobals` option.

collect core

From the remote host `coreHost`, retrieves the core dump file `coreFile` and the libraries that are used by the binary that caused the core dump. The files are placed it in a tar file on the Greenplum Database master host. The collected files aid Pivotal Support in determining the cause of the core dump on the remote host.

These are options for `collect core`:

binary *name*

Name of the binary that created the core file. Default is the `postgres` binary.

coreHost *host-name*

Required. Name of the host from which to retrieve the core file.

coreFile *path-to-core-file*

Required. Path to the core file on the remote host. An absolute or relative path can be specified. If a relative path is specified, the path is resolved relative to `/var/core`.

These examples are run in the `gpsupport` utility shell to collect core files on the host `sdw2`:

```
> collect core coreHost sdw2 coreFile /var/core/
core.mdw.1424197534.26105
> collect core coreHost sdw2 coreFile core.mdw.1424197534.26105
```

diagnose backup

The `diagnose backup` command tests the database specified by the environment variable `PGDATABASE` prior to a back up. The utility runs the queries that are used during a backup operation on the master and all segment hosts. Errors during the test indicate that a back up would have failed if it was attempted.

A backup is not performed. The target database is not modified.

diagnose connectivity

The `diagnose connectivity` command performs a UDP network stream test by forcing every Greenplum Database host to communicate with each other simultaneously. The utility attempts to connect to Greenplum Database hosts using UDP and displays statistics including round trip time and packet loss.

When running this command, the `hostfile` option must be specified. If you run the `gpsupport` interactive shell, `hostfile` can be set in the shell.

By default for each host there is 1 sender process that creates a receiver for every remote node in the cluster. The user can increase the number of receivers per node with the *nr* argument, however, the default of 1 receiver is the primary use case for this tool. Increasing the number of receivers generally succeeds only in testing how fast the segment server can process incoming traffic and some normal loss might be reported.

These are options for `diagnose connectivity`:

srpc *srpc-port*

Sender RPC listening port. Default is 8111.

> **Note:** This does not conflict with `sudp` because RPC uses the TCP protocol.

sudp *sudp-port*

Sender starting UDP port. Default is 8111.

rudp *rudp-port*

Receiver starting UDP port. Default is 7111.

nr *processes*

Number of receiver processes per host. The default is 1.

Note: The default is sufficient for most cases. Adding too many receiver processes per host could result in a false positive.

to *to-seconds*

Timeout, in seconds, before failing the test. This should not be set lower than 120 seconds.

mtu 1500 | 9000

Packet size. The default MTU of 9000. The *mtu* value can be either 1500 or 9000. For the value 9000, the test sends 8192 byte packets. For the value 1500, the test sends 1450 byte packets. The default MTU of 9000 simulates standard Greenplum Database interconnect traffic.

These examples are run in the `gpsupport` utility shell to test connectivity.

Run with default options:

```
> hostfile ~/hosts diagnose connectivity
```

Run with 2 receivers per sender:

```
> hostfile ~/hosts diagnose connectivity nr=2
```

Change Sender TCP listening ports:

```
> hostfile=~/hosts diagnose connectivity srpc=6000
```

Change sender and receiver UDP listening ports:

```
hostfile=~/hosts diagnose connectivity sudp=6000 rupd=5000
```

Example output with packet loss:

```
hdw4 --> ALL Nodes | 42.42mb/s | 49.99923% Loss | 2545mb sent |
 1272mb received
    --> mdw:7113 | 42.42mb/s | 0.00000% Loss | 1272mb sent |
 1272mb received
    --> sdw1:7113 | 0.00mb/s | 100.00000% Loss | 1272mb sent | 0mb
 received
mdw --> ALL Nodes | 45.67mb/s | 50.00043% Loss | 2739mb sent |
 1369mb received
    --> hdw4:7114 | 45.67mb/s | 0.00000% Loss | 1369mb sent |
 1369mb received
    --> sdw1:7114 | 0.00mb/s | 100.00000% Loss | 1370mb sent | 0mb
 received
sdw1 --> ALL Nodes | 77.44mb/s | 0.00000% Loss | 2323mb sent |
 2323mb received
    --> mdw:7115 | 38.72mb/s | 0.00000% Loss | 1161mb sent |
 1161mb received
    --> hdw4:7115 | 38.72mb/s | 0.00000% Loss | 1161mb sent |
 1161mb received
```

diagnose system

The `diagnose system` command determines the platform on which you are running Greenplum Database and validates various platform-specific configuration settings. The command checks both the operating system configuration files and current runtime configuration. The `diagnose system` command can use a host file or a file previously created with the `tarOut` option to validate platform settings. At the end of a successful validation process, `diagnose system` displays the message `diagnose system complete`. If the message is `Several errors detected during diagnose system`,

one or more validation checks failed. To validate a set of hosts in the Greenplum Database cluster, specify the `hostfile` option with a file that lists the host files.

To run `diagnose system` to view or gather platform settings on hosts without running validation checks, specify either the `stdout` or `tarOut` option.

Pivotal recommends that `diagnose system` be run by `root` superuser. If you do not run `diagnose system` as `root`, the utility displays a warning message and will not be able to validate all configuration settings; only some of these settings are validated.

These are the options for `diagnose system`:

host *host_ID*

Checks platform-specific settings on the host in your Greenplum Database system specified by *host_ID*.

local

Checks platform-specific settings on the Greenplum Database master host where `gpsupport` is installed.

stdout

Display collected host information from `diagnose system`. No checks or validations are performed.

This option is not valid with the `tarOut` option.

tarOut

Save all collected data to a `.gz` file (zipped tar file) in the current working directory. `diagnose system` automatically creates the `.gz` file and names it `diagnose_system_`*`timestamp`*`.tar.gz`. No checks or validations are performed.

This option is not valid with the `sdtout` option.

tarIn=*diagnose_gz_file*

Use this option to decompress and check a `.gz` file created with the `tarOut` option. `diagnose system` performs validation tasks against the file you specify in this option.

Notes

The `gpsupport` utility can extract Greenplum Database statistics into an SQL file. When extracting the statistics for a column of type `anyarray`, `gpsupport` replaces the original value with NULL and places the original statistics information in a comment. If you run the SQL file to import the statistics, the statistics for columns that store array data are not imported. The other statistics in the file are imported correctly.

To collect statistics on the columns that store array data, you can run the `ANALYZE` command on the columns.

Examples

These example `gpsupport` commands are run from the command-line. If you run the commands from the `gpsupport` interactive shell, remove the command name `gpsupport` from the command.

Install the `gpsupport_segment` helper utility in gpadmin home directory on the hosts listed in the file `hosts`:

```
$ gpsupport hostfile=~/hosts collect logs segmentFileDir /home/gpadmin/
node_collector
```

Collect all logs from a subset of the hosts in the cluster:

```
$ gpsupport collect logs segs sdw1,sdw2,sdw3-1
```

Store all temporary files in `/tmp`:

```
$ gpsupport hostfile /tmp/hostfile collect logs masterDataDir /tmp
 segDataDir /tmp
```

Extract all queries from a nonstandard directory:

```
$ gpsupport collect queries logDir $MASTER_DATA_DIRECTORY/old_cluster_logs
```

Extract all queries from a log file copied from another cluster:

```
$ gpsupport collect queries logDir /tmp logFile gpdb-2014-01-01_000000.csv
```

This command is equivalent to the previous command.

```
$ gpsupport collect queries logFile /tmp/gpdb-2014-01-01_000000.csv
```

Collect schemas and output information to different files:

```
$ gpsupport collect schema noGlobals noTupleCount outputFile /tmp/
statistics.sql
$ gpsupport collect schema noStats noTupleCount outputFile /tmp/
global_objects.sql
```

Collect schemas and output only the database schemas:

```
$ gpsupport collect schema noGlobals noStats noTupleCount
```

Collect log files from a cluster with differing usernames:

```
$ gpsupport hostfile /tmp/hosts_file dbUser greenplum_user remoteUser
 cluster_admin collect logs
```

Extract all queries run in a certain date range:

```
$ gpsupport startDate 2012-01-01 endDate 2014-01-01 collect queries
```

Gather Greenplum Database host platform settings and validate the settings. The first command gathers the settings, the second command validates the information in the `gz` file created by the first command:

```
# gpsupport diagnose system local tarOut
$ gpsupport diagnose system tarIn=gpcheck-2015-05-05_12-00-00.tar.gz
```

Copyright