Pivotal™ Greenplum Workload Manager

Version 1.0

User Guide

Rev: A01

© 2015 Pivotal Software, Inc.
Chapter 1

About Greenplum Workload Manager

Greenplum Workload Manager is a management tool for Greenplum Database you can use to monitor and manage queries and to manage resource queues.

You can use Greenplum Workload Manager to perform tasks like these:

• Monitor Greenplum Database queries and host utilization statistics
• Log when a query exceeds a threshold
• Throttle the CPU usage of a query when it exceeds a threshold
• Terminate a query
• Detect memory, CPU, or disk I/O skew occurring during the execution of a query
• Create detailed rules to manage queries
• Add, modify, or delete Greenplum Database resource queues
Workload Manager Architecture

Greenplum Workload Manager is a set of Greenplum Database-specific plugins deployed on an extensible Pivotal architecture called Kraken. All of the application logic is isolated in these plugins. Workload Manager provides the following plugins:

Agent plugins:
- Publish information about active Greenplum Database queries
- Publish information about postgres processes
- Advertise query termination capability
- Advertise query throttling capability
- Advertise threshold logging capability

Configuration management plugins:
- Query the state of the Greenplum Database cluster periodically
- Inform the Kraken system of the Greenplum Database cluster state and size allowing gp-wlm to automatically grow when the database is expanded.
- Deploy configurations throughout the cluster.

Command-line interface plugins:
- Add, modify, or delete rules
- Monitor queries and skew
- Manage Greenplum Database resource queues

Rules engine plugins:
- Provide extended functionality used during rules creation

The Kraken runtime loads these plugins at execution time.
Chapter 2

Installing Greenplum Workload Manager

Prerequisites

- Red Hat Enterprise Linux (RHEL) 64-bit 5.5+ or 6 or CentOS 64-bit 5.5+ or 6
- Greenplum Database version 4.3.x

Running the Greenplum Workload Manager Installer

The Greenplum Workload Manager installer binary is installed on the Greenplum Database master node. It automatically distributes the software to all segment servers in the database cluster. Normally, the Workload Manager installer is run as an option in the gpcmdr setup utility. The installer detects the installed Workload Manager version, if any, and performs an upgrade if necessary. Run the ./gp-wlm.bin package installer with the --force option to force reinstallation of the current version or an earlier version.

The package installer has the following syntax:

```
./gp-wlm.bin --help
./gp-wlm.bin --install=DIR [ --force ] [ --install-concurrency=COUNT ]
[ --no-remove-old ] [ --skip-health-check ] [ --tool-manifest=FILE ]
```

Options

--help

Displays command syntax for the gp-wlm.bin installer.

--install=DIR

The --install option is required. It specifies the directory where Greenplum Workload Manager will be installed, for example /home/gpadmin.

--force

The installer checks the currently installed version and only performs an upgrade if the current version is older than the installer version. The --force option skips the version check and performs the upgrade.

--install-concurrency=COUNT

The maximum number of hosts to bootstrap at once. The default count is computed by the installer. This option places a limit on the number of processes the installer can fork.

--no-remove-old

By default, the installer removes all previous installation directories after an upgrade. The --no-remove-old option prevents the installer from removing old installation directories.

--skip-health-check

Do not perform a cluster health check after Workload Manager installation completes. This option is not recommended.

--tool-manifest filename

The optional --tool-manifest option specifies a text file containing a list of commands and their absolute paths. Workload Manager normally finds standard system commands on the path. If your environment has incompatible implementations of these commands on the path, create a manifest file that provides the absolute path to a standard version.
Following is an example tools manifest file:

```
stat=/home/gpadmin/bin/stat
readlink=/bin/readlink
ssh=/home/me/bin/myssh
```

The installer creates a `gp-wlm-data` directory in the installation directory and installs the Greenplum Workload Manager release into it. A symbolic link `gp-wlm` in the installation directory links to the specific Greenplum Workload Manager release directory.

1. Log in to the Greenplum master host as the `gpadmin` user.
2. Ensure that the Greenplum Workload Manager installer is executable.

```
$ chmod +x gp-wlm.bin
```

3. Run the Greenplum Workload Manager installer. Specify the absolute path to an installation directory where you have write permission. For example:

```
$ ./gp-wlm.bin --install=/home/gpadmin/
```

This command installs Greenplum Workload Manager in the `gp-wlm-data` subdirectory on all of the segments and creates the `gp-wlm` symbolic link. For example, the above command installs Workload Manager in `/home/gpadmin/gp-wlm-data/gp-wlm-release` and creates the symbolic link `/home/gpadmin/gp-wlm`.

*Note:* In rare cases, the installer can fail during the `cluster-health-check` phase. If the cluster is reported not healthy, re-run the installer with the `--force` option.

4. For convenience you may source `INSTALL-DIR/gp-wlm/gp-wlm_path.sh` to add the Workload Manager executables to your path.

To uninstall Greenplum Workload Manager, run the following command:

```
$ INSTALLDIR/gp-wlm/bin/uninstall --symlink INSTALLDIR/gp-wlm
```
Chapter 3
Managing Greenplum Workload Manager Services

Greenplum Workload Manager installs and runs four services on all segment hosts in the Greenplum cluster:

- agent
- cfgmon
- rabbitmq
- rulesengine

The services can be managed using the `INSTALLDIR/gp-wlm/bin/svc-mgr.sh` command. The command has the following syntax:

```
INSTALLDIR/gp-wlm/bin/svc-mgr.sh \
  --service=SVCMAME \ 
  --action=ACTION
```

`SVCMAME` may be agent, cfgmon, rabbitmq, rulesengine, or all. If `SVCMAME` specifies an individual service, only that service is modified. Specify all to manipulate all services.

The `ACTION` parameter affects only the local system, unless it is prefixed with `cluster-`, in which case it runs on all hosts in the cluster. The actions are:

- start / cluster-start – Start any of the Workload Manager services that are not running.
- stop / cluster-stop – Stop any Workload Manager services that are running.
- status / cluster-status – Determine if the services are running.
- restart / cluster-restart – Restart the Workload Manager services.
- enable / cluster-enable – Enable and start Workload Manager services.
- disable / cluster-disable – Stop and disable Workload Manager services.

If you source the `INSTALLDIR/gp-wlm/gp-wlm_path.sh` file in your shell, the Workload Manager scripts are in your path. Otherwise, you must provide the full path to the utility in the `gp-wlm/bin` directory.

When a service is stopped, it will not be restarted until the `start` action is invoked, or the local machine reboots, whichever comes first.

When a service is disabled, it will not be restarted until the `enable` action is invoked. This is persistent across reboot.

The following example checks the status of all Workload Manager services on the local host:

```
[gpadmin@mdw ~]$ svc-mgr.sh --service=all --action=status
RabbitMQ is running out of the current installation. (PID=22541)
agent (pid 22732) is running...
cfgmon (pid 22858) is running...
rulesengine (pid 22921) is running...
```

Checking the Health of Greenplum Workload Manager Services

At any time, the health of Greenplum Workload Manager services can be verified across the cluster by invoking the `cluster-health-check` utility. This tool confirms that all services are running across the
cluster, and that messages are being received from each machine in the cluster. Following is the syntax for `cluster-health-check`:

```bash
INSTALLDIR/gpwlm/bin/cluster-health-check --symlink=/absolute/path/to/installation/
symlink
  [--max-concurrency=N]
  [--max-cluster-checks=N]
  [--help]
```

Options:
- `-c` or `--max-concurrency`
  The `max-concurrency` option specifies the number of hosts to check at once. The default is a computed value based on the number of hosts in the cluster: 20 if there are fewer than 100 hosts, 50 if there are 100 to 199 hosts, and 75 if there are 200 or more hosts.

- `-m` or `--max-cluster-checks`
  The number of times to check for a healthy cluster. The default is 1.

- `-s` or `--symlink`
  The absolute path to the `gp-wlm` directory linked to the installed Workload Manager release. Required.

- `-h` or `--help`
  Display command usage information and exit.

If the command reports an error communicating with one or more services, the cluster may be restarted with this command:

```bash
INSTALLDIR/gp-wlm/bin/svc-mgr.sh --action=cluster-restart --service=all
```

This command stops and then restarts each of the Workload Manager services on each segment host.
Chapter 4

Using the Greenplum Workload Manager Command Line

The Greenplum Workload Manager command line utility, `gp-wlm`, provides access to Workload Manager capabilities. The utility may be run by entering commands interactively or by specifying equivalent actions using command-line options. The command-line options are useful for scripting, since they require no interactive user input.

To get help in interactive mode, issue the command: `help`

To get help for command line invocation, issue the command: `gp-wlm --help`

Below is the `gp-wlm` command syntax:

```
Usage: gp-wlm [-g?V] [-rq-add=<queue-name> with <queue-settings>]
             [-rq-delete=<queue-name>]
             [-rq-modify=<queue-name> with <queue-settings>]
             [-rq-show=all]
             [-rq-useradd=<user> to <queue-name>]
             [-rq-userdel=<user> from <queue-name>]
             [-rule-add=[transient] <name> <rule>]
             [-rule-delete=<name>]
             [-rule-dump=<path>]
             [-rule-modify=transient <name> <rule>]
             [-rule-restore=<path>]
             [-rule-show=all|<name>]
             [-gptop]
             [-set-domain=<domain>]
             [-set-host=<host>]
             [-schema-path=<path>]
             [-help]
             [-usage]
             [-version]`
```
Using gp-wlm in Interactive Mode

1. Start gp-wlm at the command line:

   $ gp-wlm

   The `gp-wlm` command prompt displays the name of the host where `gp-wlm` is running and the name of the Greenplum Database cluster or domain.

   Enter `help` at the interactive prompt for a usage message.

When using the `gp-wlm` command-line:

- Enter each command on a single line. Commands are executed immediately.
- Enter the `help` command to view a list of Workload Manager commands.

While entering a command, get help with command syntax by pressing the tab key to show valid options. This is especially useful when constructing a rule. In the following partial example, user entry is in bold.

```
mdw/gpdb-cluster> rule <tab>
   add     delete  dump    modify  restore show
mdw/gpdb-cluster> rule add <tab>
   <rule-name> transient
mdw/gpdb-cluster> rule add transient <tab>
   <rule-name>
mdw/gpdb-cluster> rule add transient myrule <tab>
   gpdb_record host:          pg_terminate_backend
   mdw/gpdb-cluster> rule add transient myrule gpdb_record(<tab>
   ) gpdb_segment_role message query_start usename
   current_query host pid session_id
   ...
```

Enter the quit command at the prompt to exit the `gp-wlm` interactive mode.
Setting the Workload Manager Target Host and Domain

Use the set host and set domain commands to set the default host and domain for the Workload Manager session.

It is recommended to only run the gp-wlm tool on the Greenplum Database master node and to leave the host and domain at their default values.

The default host is the name of the machine where you execute gp-wlm. The host name must be resolvable in DNS. You can specify different host and cluster names on the gp-wlm command line by supplying the --set-host and --set-domain command line options.

Example:

```
mdw/gpdb-cluster> set host smdw
smdw/gpdb-cluster> set domain gpdbsys
smdw/gpdbsys>
```
Chapter 5

Creating Workload Manager Rules

Rules trigger actions when they match events. The agent plugins on the segment hosts collect statistics and associated data. The rulesengine matches them to rules, and performs the specified actions in the agent plugins.

- Add Rule Command Syntax
- Understanding Rules
- Rule Actions
- Known Issues
- Best Practices for Rules
- Example Rules
- Querying the gp_wlm_records Table
Add Rule Command Syntax

The `rule add` command adds a rule. Here is the syntax for the rule add command:

```
rule add [transient] name action-name(action-args) when expression
```

**transient**

Rules may be *persistent* or *transient*. A persistent rule remains active until it is deleted, while a transient rule disappears when the rulesengine service is shut down on all hosts. Rules are persistent by default; you must include the `transient` keyword to create a transient rule.

**name**

A unique name for the rule.

**action-name**

The action to perform. One of the following:

- `host:throttle_gpdb_query` – specify a maximum allowed CPU utilization percentage for a Greenplum Database query.
- `pg_terminate_backend` – terminate a Greenplum Database session.
- `gpdb_record` – record an event about a query.

**action-args**

Arguments that pass values to the action, if needed. An argument is specified as an `arg-name=value` pair. Multiple arguments are separated by commas.

**expression**

A Boolean expression that filters targets for the action. The expression references one or more datum to filter the facts that trigger the action. The expression may contain regular expressions (regex).
Understanding Rules

The following is a diagrammed example that illustrates the components of Workload Manager rules.

Figure 1: Anatomy of a Rule

Scoped Action

The action to be invoked for each set of data where all conditions are true.

Scope

The scope, host: in this case, is the data that makes this action unique. A scope of host indicates that the action will be executed on the specific host that matches the data in the conditions. In this case, the rule only throttles the query on hosts where it is using more than 22% CPU.

An action that has no scope is a global action for the cluster. It can be triggered from a rule matching on any host but the action is global. For example, pg_terminate_backend, which cancels a query, is a global action because it is not valid to cancel a query on only a single host.

Action

The action name, throttle_gpdb_query in the example, is the name of the action to take for each set of data where all conditions are true. The action is followed by arguments that are unique to each action. In the example, the max_cpu=33 argument tells throttle_gpdb_query to limit Greenplum Database processes for the query on the host to a maximum of 33% CPU utilization.

Conditions

The conditional portion of the rule follows the when keyword and consists of a number of datum comparisons connected by the Boolean operators and or or.

Comparisons can be enclosed in parentheses to indicate precedence. A rule can contain any number of such comparisons. For example:

```
host:pid:cpu_util > 50 or (host:pid:cpu_util > 30 and session_id:host:pid:usename = "fred")
```

Scoped Datum

Datums are data items collected by the agent, and include operating system statistics, OS process statistics, and database query data. Like actions, datums are scoped. The scope specifies the source of the datum. The host:pid scope for the cpu_util datum means that the CPU utilization is the percentage of CPU used by an OS process executing on a specific
host. The session_id:host:pid scope for the username datum indicates that the data is from a Greenplum Database segment query process executing on a host.

Other datum scopes are host and session_id. A host scope qualifies datums that are from the operating system of a segment host. The session_id scope is used for datums for database queries that Workload Manager calculates by aggregating datums from all segments executing the query.

For a list of datums that belong in each scope, see *Workload Manager Datum Reference*.

**Operators**

Datums can be compared to values or other datums using the following operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Value Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>A number for numeric datums or a quoted string for strings.</td>
<td>Matches only when the values are exactly equal.</td>
</tr>
<tr>
<td>!=</td>
<td>A number for numeric datums or a quoted string for strings.</td>
<td>Matches when the values are not equal.</td>
</tr>
<tr>
<td>=~</td>
<td>Regular expression on the right side enclosed in /..</td>
<td>Performs a regular expression match between the string value and the specified regex.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Number</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Number</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Number</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Number</td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>

The following command adds a persistent rule named throttle_query that throttles any query containing the text "select count" to a maximum of 20% of CPU. (The entire command must be entered on one line.)

```
/mdw/gpdb-cluster> rule add throttle_query host:throttle_gpdb_query(max_cpu=20) when session_id:host:pid:current_query =~ /.*select count.*/
```

In this rule, host:throttle_gpdb_query() is an action with one argument, max_cpu, which is set to 20. The when expression applies the action to any query that contains the text "select count". In the when expression, session_id:host:pid matches any query fact. The current_query portion of the selector is a reference to a datum, the text of the query.
Rule Actions

host:throttle_gpdb_query
Throttle a Greenplum Database query on a specified host.
Arguments:

max_cpu - Hold process to a maximum of this percentage CPU utilization.
pid - The process to throttle.
session_id - The session to throttle.

The max_cpu argument is required. The pid and session_id arguments can be inferred from the session_id in the when clause and are normally omitted.

pg_terminate_backend
Terminate a query on all hosts.
Arguments:

session_id – The session ID of the query to terminate.

The normal use case is to omit the argument and allow the session ID to be inferred by using the session_id in the rule’s when clause. Workload Manager then determines which pid to terminate. See http://www.postgresql.org/docs/9.3/static/functions-admin.html for more details.

The following example terminates any query that has been executing for more than 20 seconds:

```
mdw/gpdb-cluster> rule add cancel_query pg_terminate_backend() when session_id:host:pid:runtime > 20
```

gpdb_record
Logs a message when a rule is matched.
Arguments:

current_query - The text of the current query
gpdb_segment_role - Role of the database instance: GPDB_MASTER or GPDB_SEGMENT
host - The hostname of the segment
message – Informative string describing the reason for recording.
pid - The postgres process associated with the query
query_start - Query start time
session_id - Session id of the query
usename - Name of the user logged into this backend

The normal use case is to only use the message argument and allow all other arguments to be inferred.

The following example logs all queries:

```
mdw/gpdb_cluster> rule add record_query gpdb_record(message="all") when session_id:host:pid:usename =~ /.*/
```
Displaying Rules

Use the `rule show` command to see existing rules. You can show all existing rules or specify a single rule by name.

```
rule show { all | rule-name }
```

The `rule show all` command in this example lists all registered rules:

```
mdw/gpdb-cluster> rule show all
--- Name --- ----------- Expression -----------
record_query     gpdb_record(message="all") when session_id:host:pid:usename =~ /.*/
cancel_query     pg_terminate_backend() when session_id:host:pid:runtime > 20
throttle_query   host:throttle_gpdb_query(max_cpu=20) when
                 session_id:host:pid:current_query =~ /.*select count.*/
```

This example lists a single rule by name:

```
mdw/gpdb-cluster> rule show throttle_query
--- Name --- ----------- Expression -----------
throttle_query   host:throttle_gpdb_query(max_cpu=20) when
                 session_id:host:pid:current_query =~ /.*select count.*/
```

Deleting a Rule

The `rule delete` command removes a rule.

```
rule delete rule-name
```

Modifying a Rule

Use the `rule modify` command to alter the expression for an existing rule. You may also remove the transient keyword from the rule declaration to convert it to a persistent rule. Conversion from persistent to transient is not currently supported.

```
rule modify [transient] name action-name (action-args)
when expression
```

This example modifies the `cancel_query` rule to alter the number of seconds a query runs on a host to trigger the rule from 20 to 25:

```
mdw/gpdb-cluster> rule modify cancel_query pg_terminate_backend() when
                 session_id:host:pid:runtime > 25
```

Saving Rules to Disk

The `rule dump` command saves all persistent rules in the cluster to a text file, one rule per line.

```
rule dump path
```

If you do not provide the full path to the file, the file is written relative to the directory where you started the `gp-wlm` session. The user running `gp-wlm` must have permission to write the file at the specified location. If the file exists, the `rule dump` command overwrites it.

The following example saves rules to the `/home/gpadmin/rules/20150910-1` file. If the `/home/gpadmin/rules` directory does not exist, an error is reported.

```
mdw/gpdb-cluster> rule dump /home/gpadmin/rules/20150910-1
```
Restoring Rules from Disk

The `rule restore` command restores all rules from a file. If a rule in the restored file matches an existing rule with the same name, an error message is displayed and the existing rule is unchanged.

```
rule restore path
```
Known Issues

To write a rule that performs a Greenplum Database action (gpdb_record, pg.terminate_backend, host:throttle_gpdb_query), the condition must include a session_id, even when the intended condition is based solely on process information. For example, the following rule appears to terminate any query that uses more than 20% of system memory:

```
pg.terminate_backend() when host:pid:resident_size_pct > 20
```

However, because this rule contains no session_id, Workload Manager cannot infer the query to terminate, and the rule will not be added. To get the desired behavior, add an always-true session_id condition to the rule, for example:

```
pg.terminate_backend() when host:pid:program_size_pct > 20
and session_id:host:pid:runtime > 0
```
Creating Workload Manager Rules

Best Practices for Rules

1. Avoid creating rules that modify the condition the rule's expression is matching. For example, consider this rule:

```
host:throttle_gpdb_query(max_cpu=20) when host:pid:cpu_util > 30 and session_id:host:pid:runtim > 0
```

If CPU usage goes above 30%, the rule triggers and reduces the usage to 20%. When the usage falls below 30%, the rule is no longer matched, so the throttling ends and usage can again climb to 30%. This creates an undesirable cyclic behavior. Instead, create a rule like the following:

```
host:throttle_gpdb_query(max_cpu=30) when host:pid:cpu_util > 20 and session_id:host:pid:runtim > 0
```

This rule triggers at 20% CPU utilization and throttles the CPU to 30% utilization. The throttling continues until utilization drops below 20%. The `session_id:host:pid:runtim` condition is true for any running query and provides the necessary `session_id` for the `throttle_gpdb_query` action.

2. Avoid creating rules that terminate a query based on skew alone. Consider the following rule:

```
pg_terminate_backend when session_id:resident_size_pct_skew > 10
```

This is a poor rule for two reasons. First, it terminates all queries when skew is above 10, including queries that were not contributing to skew. Second, well behaved queries can temporarily experience skew high enough to achieve this condition. For example, if the segments do not complete a query at the same time, skew can appear near the end of execution. A query could run normally across several nodes and then, as each node completes its portion of the query, its resource utilization drops, causing a temporary increase in skew while other nodes are still running.

3. Rules that match data with `datid:scope` will trigger for any database in the cluster unless a predicate is added to confine the match to a target database. For example, this rule triggers whenever the number of connections to any single database exceeds 10:

```
gpdb_record(message="exceeded 10 connections") when session_id:host:pid:runtim > 0 and datid:numbackends > 10
```

Add a predicate to filter for the database associated with the session:

```
gpdb_record(message="exceeded 10 connections on foo") when session_id:host:pid:runtim > 0 and datid:datname = "foo" and datid:numbackends > 10
```
Example Rules

This section provides examples of rules written for various purposes.

Note: Rules must be entered on a single line, but the rules shown in this section are wrapped for readability.

Record high cpu utilization queries

The following rule invokes the `gpdb_record` action when the gpadmin user runs a query and its total cpu utilization on a host exceeds 100%.

```
rule add simple gpdb_record(message="Too much cpu for gpadmin")
when session_id:host:total_cpu > 100
and session_id:host:pid:usename = 'gpadmin'
```

Throttle the cpu utilization of a query

This rule invokes the `host:throttle_gpdb_query` action when the cpu utilization of a process exceeds a threshold and the query has run for more than 20 seconds.

```
rule add throttle host:throttle_gpdb_query(max_cpu=30)
when host:pid:cpu_util > 20
and session_id:host:pid:usename = 'gpadmin'
and session_id:host:pid:runtime > 20
```

Cancel any query running longer than 120 seconds

This rule invokes the `pg_terminate_backend` action when a `session_id:host:pid:runtime` exceeds two minutes.

```
rule add kill_long pg_terminate_backend()
when session_id:host:pid:runtime > 120
```

Throttle and even out skew

This rule invokes `host:throttle_gpdb_query` when the total cpu usage of a query on a host exceeds 90% and the current query is a select on the `skewtest` table.

```
rule add skewrule host:throttle_gpdb_query(max_cpu=50)
when session_id:host:total_cpu > 100
and session_id:host:pid:current_query =~ /select.*skewtest/
```

You can observe the effects of this rule in the `gptop` GPDB Skew page.

Complex rule

This rule invokes `gpdb_record` for a query that meets the following criteria:

- a query has total CPU usage greater than 90% on a host and has been running for more than 45 seconds, or
- has cpu skew greater than 20%, and
- is a select on a table that contains "test" in its name.

```
rule add comborule gpdb_record(message="My Message")
when ((session_id:host:total_cpu > 90 and session_id:host:pid:runtime > 45)
or session_id:cpu_skew > 20)
```
The rule shows how you can group Boolean expressions with parentheses.

**Record queries with high memory usage**

This rule records a message when a query process exceeds 20% of the resident memory on a host.

```
rule add transient mem_high_segment_usage_20
gpdb_record(message="MEM: high segment pctusage - 20%") when
host:pid:resident_size_pct > 20
and session_id:host:pid:usename =-/*
```

**Record queries with memory (rss) skew above 10%**

This rule calls the `gpdb_record` action to log when a host

```
rule add mem_skew_10 gpdb_record(message="MEM: query skew 10")
when session_id:resident_size_pct_skew > 10
and session_id:host:pid:usename =-/*
```
Querying the gp_wlm_records Table

The gp_wlm_records table contains a record of events describing where, why, and how the gpdb_record action was triggered by a rule on the Greenplum cluster.

The primary identifier of each entry in the table is the ident column. This column stores a unique identifier that represents a specific rule that triggered on a specific node in the cluster. If a rule triggers on more than one node in the cluster at the same time, these are treated as separate events and each will receive a unique identifier.

Following are two sample entries from the gp_wlm_records table. In this example, the rule was created to track when a query exceeds a certain CPU utilization on any node in the cluster:

```
Fri Apr  3 19:03:32 2015 | BEGIN | 6aa79b15-577f-4f08-bbb0-6abad9a2c844 | ip-172-31-14-241 | 2015-04-03 19:00:49.990213+00 | CPU Exceeded | 15109       224691  |
GPDB_SEGMENT | gpadmin | select count(*) from bigtest A inner join bigtest B on B.str1 similar to '(x+x)+y';   ||
...  
Fri Apr  3 21:45:03 2015 |  END  |  6aa79b15-577f-4f08-bbb0-6abad9a2c844  |||||| |||
```

In the above example, the state column represents when a query began triggering a rule on a given node and when it stopped. The hostname column stores the host on which the rule triggered.
Chapter 6

Using the Workload Manager Graphical Interface (gptop)

The Workload Manager Graphical interface, gptop, is a curses interface that you can use to monitor live data for the rules engine, host statistics, active Greenplum Database queries, and database skew.

You can start gptop from the command line by running gptop in a terminal. If you are already using interactive gp-wlm, enter the gptop command to enter the monitor.

Note: If you use the PuTTY ssh/telnet client for Windows to run gptop, you may experience problems with function keys and line-drawing characters with the default settings. To support function keys, in the PuTTY Configuration window, choose Connection > Data and enter xterm-color or putty in the Terminal-type string field. To enable correct line-drawing characters, choose Window > Translation and set Remote character set to Use font encoding.

When you first start gptop, the GPDB Queries pane (see below) is selected. At any time, you can press the F2 key to get a pane selection menu. Use the Tab, Left-Arrow, or Right-Arrow keys to make a selection. Press F2 to close an open menu without making a selection.

An asterisk (*) next to a column heading indicates that the rows are sorted by that column. To change the sort order, press the F3 key, then choose the number of the column you want to sort by from the pop-up menu.

Press q or choose File > Exit to leave gptop.

The gptop monitoring features are under the Monitor menu. The Monitor menu has four options:

- GPDB Queries – Shows active Greenplum Database queries
- GPDB Skew – Shows skew statics for active queries
- Hydra – Shows statistics from the rules engine
- SysData – Shows performance statistics for each host in the cluster

**GPDB Queries**

The GPDB Queries monitor displays a line for each active Greenplum Database query.

- **SessID**
  - The session id for the query.

- **Time**
  - The number of seconds since the query began executing.

- **User**
  - The name of the Greenplum Database role that submitted the query.

- **ClientAddr**
  - The network address from which the query was submitted.

- **DatName**
  - The database name the query is running against.

- **Query**
  - The text of the query.
GPDB Skew

The GPDB Skew monitor shows calculated skew statistics for active Greenplum Database queries. Statistics are calculated on each host in the system and then sent to the master where they are summarized. You can select a host and press Enter to see statistics for the host. The calculated skew value is the cubed standard deviation across the cluster. Values closer to 0.0 indicate less skew. The GPDB Skew monitor shows the following columns for each active query:

SessID
- The Greenplum Database session ID for the query.

Time
- The number of seconds since the query started.

User
- The Greenplum Database role that submitted the query.

CPU-Skew
- A measure of CPU skew calculated as the cubed standard deviation of the total CPU for each host for the query.

MEM-Skew
- A measure of memory skew calculated as the cubed standard deviation of the total resident size percent for each host for the query.

READ-Skew
- A measure of disk read I/O skew calculated as the cubed standard deviation of the bytes read per second for each host for the query.

WRITE-Skew
- A measure of disk write I/O skew calculated as the cubed standard deviation of the bytes written per second for each host for the query.
Use `rq` commands to show, create, remove, and modify resource queues, and to manage the roles that are assigned to resource queues.

The `queue-settings` argument can contain the following properties:

### Table 1: Resource Queue Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_statements</td>
<td>Integer</td>
<td>Limits the number of queries that can be executed by roles assigned to the resource queue. Either the <code>active_statements</code> or <code>max_cost</code> property must be set on each resource queue.</td>
</tr>
<tr>
<td>max_cost</td>
<td>Float</td>
<td>Sets a maximum limit on the total cost of queries that can be executed by roles assigned to the resource queue. The cost of a query is estimated by the query planner and is measured in units of disk page fetches. Either the <code>active_statements</code> or <code>max_cost</code> property must be set on each resource queue.</td>
</tr>
<tr>
<td>min_cost</td>
<td>Float</td>
<td>Sets the minimum estimated query cost for a query to be managed by the resource queue. Queries with estimated costs below this threshold are executed immediately.</td>
</tr>
<tr>
<td>cost_overcommit</td>
<td>Boolean</td>
<td>If a resource queue is limited based on MAX_COST, a query that exceeds the MAX_COST limit is allowed to execute if the system is idle and COST_OVERCOMMIT is true. If COST_OVERCOMMIT is set to false, queries that exceed MAX_COST are always rejected.</td>
</tr>
<tr>
<td>priority</td>
<td>Enumeration</td>
<td>Sets the relative priority of queries executed by roles assigned to the resource queue. The allowed values, in order of increasing priority, are MIN, LOW, MEDIUM, HIGH, and MAX.</td>
</tr>
<tr>
<td>memory_limit</td>
<td>Integer (kilobytes)</td>
<td>Sets the total amount of memory that all active statements submitted to the queue may consume. The minimum is 10240KB. There is no maximum, but when a query executes it is limited by the segment host's physical memory. Set the parameter to -1 for no limit.</td>
</tr>
</tbody>
</table>

Specify the resource queue properties in a `parameter-name=value` format, for example:

```
mdw/gpdb-cluster > rq modify myrq with active_statements=10
```
Separate multiple queue settings with a comma. The queue-settings argument must not contain spaces. This example sets three properties:

```
mdw/gpdb-cluster> rq add ETL with
active_statements=3,priority=LOW,memory_limit=524288
```

A Greenplum Database role (login user) is associated with a single resource queue. Newly created roles are added to the `pg_default` queue if another resource queue is not specified.

Queries submitted by users associated with a resource queue are managed by the queue. The queue's settings determine whether queries are accepted or rejected, if they run immediately or wait for resources to be returned to the queue, how much memory to allocate to the query, and the relative amount of CPU the query will have.

Resource queues share the memory allocated to each segment. Adding a new resource queue or altering a queue's settings may require adjusting other resource queues to avoid over-allocating the available memory and causing queries to fail. See the Workload Management section in the *Greenplum Database Administrator Guide* for guidelines on configuring resource queues.

- Adding a New Resource Queue
- Deleting a Resource Queue
- Modifying a Resource Queue
- Displaying Resource Queues
- Adding Users to Resource Queues
- Deleting a User from a Resource Queue
Adding a New Resource Queue

The `rq add` command creates a new resource queue. The command has the following syntax:

```
rq add queue-name with queue-settings
```

You must set one or both of the threshold properties—`active_statements` or `max_cost`—when you create a new resource queue.

The following example creates an ETL queue that can run three concurrent queries at low CPU priority relative to other queries.

```
mdw/gpdb-cluster> rq add etl with active_statements=3,priority=low
```
**Deleting a Resource Queue**

Delete an existing resource queue by name using the `rq delete` command:

```
rq delete queue-name
```

It is not possible to delete a queue that has roles assigned to it.
Modifying a Resource Queue

Use the `rq modify` command to alter queue settings. You can add new settings or update existing settings by specifying properties in the `queue-settings` argument.

```
rq modify queue-name with queue-settings
```

The following example modifies the ETL queue to run two concurrent queries with a maximum of 524288KB (512MB) of memory. Each query will be allocated 256MB of memory.

```
rq modify etl with active_statements=2,memory_limit=524288
```
### Displaying Resource Queues

Use the `rq show all` command to display resource queues. This example displays settings for the `etl` and `pg_default` resource queues.

```sql
mdw/gpdb-cluster> rq show all
rsqname | resname             | ressetting | restypeid |
etl     | active_statements   | 2          | 1         |
etl     | max_cost            | -1         | 2         |
etl     | min_cost            | 0          | 3         |
etl     | cost_overcommit     | 1          | 4         |
etl     | priority            | low        | 5         |
etl     | memory_limit        | 524288     | 6         |
pg_default| active_statements   | 10         | 1         |
pg_default| max_cost            | -1         | 2         |
pg_default| min_cost            | 0          | 3         |
pg_default| cost_overcommit     | 0          | 4         |
pg_default| priority            | medium     | 5         |
pg_default| memory_limit        | -1         | 6         |
```
Adding Users to Resource Queues

The `rq useradd` command adds a user to a resource queue. The user is removed from their previous resource queue as users are associated with only one resource queue. The user's subsequent queries are managed by the new resource queue.

```
rq useradd user to queue-name
```
Deleting a User from a Resource Queue

The `rq userdel` command deletes a role from a resource queue. The user will be associated with the default queue, `pg_default`.

```
rq userdel user from queue-name
```
Chapter 8

Troubleshooting

You may collect all logs across the cluster using a single command. To create a tarball of all logs in the current directory, invoke:

```
bin/gather-cluster-logs.sh --symlink <LN>
```

where \texttt{LN} is the path to the \texttt{gp-wlm} symbolic link to the Greenplum Workload Manager installation directory.
Chapter 9

Workload Manager Datum Reference

The following table lists datums collected by the Greenplum Workload Manager. These datums can be used in Workload Manager rules to select facts that trigger an action. In rules, prefix datums with their scope, for example:

\[
\text{host:cpu\_util} > 35
\]

This will match any host with greater than 35% CPU utilization. The following expression matches a single postgres process on any host using more than 35% CPU:

\[
\text{host:pid:cpu\_util} > 35 \text{ and } \text{host:pid:name} = \text{'}postgres\text{'}
\]

Table 2: Datums

<table>
<thead>
<tr>
<th>Scope</th>
<th>Datum</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datid</td>
<td>blks_hit</td>
<td>integer</td>
<td>Number of times disk blocks were found already in the PostgreSQL buffer cache</td>
</tr>
<tr>
<td>datid</td>
<td>blks_read</td>
<td>integer</td>
<td>Number of disk blocks read in this database</td>
</tr>
<tr>
<td>datid</td>
<td>xact_rollback</td>
<td>integer</td>
<td>Number of transactions in this database that have been rolled back</td>
</tr>
<tr>
<td>datid</td>
<td>xact_commit</td>
<td>integer</td>
<td>Number of transactions in this database that have been committed</td>
</tr>
<tr>
<td>datid</td>
<td>numbackends</td>
<td>integer</td>
<td>Number of backends</td>
</tr>
<tr>
<td>datid</td>
<td>datname</td>
<td>string</td>
<td>Name of this database</td>
</tr>
<tr>
<td>host</td>
<td>day</td>
<td>integer</td>
<td>Day as 0 - 30</td>
</tr>
<tr>
<td>host</td>
<td>day_of_week</td>
<td>integer</td>
<td>Day as 0 - 6</td>
</tr>
<tr>
<td>host</td>
<td>day_of_week_</td>
<td>string</td>
<td>Mon, Tue, ...</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>hour</td>
<td>integer</td>
<td>Hour as 0 - 23</td>
</tr>
<tr>
<td>host</td>
<td>minute</td>
<td>integer</td>
<td>Minute as 0 - 59</td>
</tr>
<tr>
<td>host</td>
<td>month</td>
<td>integer</td>
<td>Month as 0 - 11</td>
</tr>
<tr>
<td>host</td>
<td>node_cpu_util</td>
<td>float</td>
<td>Current CPU utilization on this host, normalized by number of active CPUs.</td>
</tr>
<tr>
<td>host</td>
<td>year</td>
<td>integer</td>
<td>Numeric year</td>
</tr>
<tr>
<td>host:pid</td>
<td>resident_size_</td>
<td>float</td>
<td>The size of this process’ resident memory as a percentage of total system</td>
</tr>
<tr>
<td></td>
<td>pct</td>
<td></td>
<td>memory.</td>
</tr>
<tr>
<td>host:pid</td>
<td>program_size_</td>
<td>float</td>
<td>The size of this process as a percentage of total system memory.</td>
</tr>
<tr>
<td></td>
<td>pct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host:pid</td>
<td>dirty_size_bytes</td>
<td>integer</td>
<td>The size of dirty pages used in this process (bytes).</td>
</tr>
<tr>
<td>Scope</td>
<td>Datum</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>host:pid</td>
<td>data_size_bytes</td>
<td>integer</td>
<td>The size of data+stack memory region in this process (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>library_size_bytes</td>
<td>integer</td>
<td>The size of library memory region in this process (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>text_size_bytes</td>
<td>integer</td>
<td>The size of code memory region in this process (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>shared_size_bytes</td>
<td>integer</td>
<td>The size of all shared pages used by this process (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>resident_size_bytes</td>
<td>integer</td>
<td>The size of resident memory consumed by this process (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>program_size_bytes</td>
<td>integer</td>
<td>The total program size (bytes).</td>
</tr>
<tr>
<td>host:pid</td>
<td>writes_per_sec</td>
<td>float</td>
<td>The number of total write(2) calls per second by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>reads_per_sec</td>
<td>float</td>
<td>The number of total read(2) calls per second by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>disk_write_bytes_per_sec</td>
<td>float</td>
<td>The number of bytes written to disk per second by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>disk_read_bytes_per_sec</td>
<td>float</td>
<td>The number of bytes read from disk per second by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>write_bytes_per_sec</td>
<td>float</td>
<td>The number of bytes written per second (disk + net + IPC) by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>read_bytes_per_sec</td>
<td>float</td>
<td>The number of bytes read per second (disk + net + IPC) by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>writes</td>
<td>integer</td>
<td>Total number of write system calls made by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>reads</td>
<td>integer</td>
<td>Total number of read system calls made by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>disk_write_bytes</td>
<td>integer</td>
<td>Total number of bytes written to disk by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>disk_read_bytes</td>
<td>integer</td>
<td>Total number of bytes read from disk by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>write_bytes</td>
<td>integer</td>
<td>Total number of bytes (disk, network, IPC) written by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>read_bytes</td>
<td>integer</td>
<td>Total number of bytes (disk, network, IPC) read by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>avg_cpu_util</td>
<td>float</td>
<td>Average CPU utilization consumed by this process over the last two polling intervals.</td>
</tr>
<tr>
<td>host:pid</td>
<td>cpu_util</td>
<td>float</td>
<td>Percentage of total CPU utilization consumed by this process.</td>
</tr>
<tr>
<td>host:pid</td>
<td>state</td>
<td>string</td>
<td>Kernel state of this process; the man page for proc(5) for more information.</td>
</tr>
<tr>
<td>Scope</td>
<td>Datum</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>host:pid</td>
<td>long_name</td>
<td>string</td>
<td>By default, this is the absolute path to the process executable, but may be overridden by the process itself to status information in utilities like ps(1).</td>
</tr>
<tr>
<td>host:pid</td>
<td>name</td>
<td>string</td>
<td>The filename of the executable.</td>
</tr>
<tr>
<td>session_id</td>
<td>cpu_skew</td>
<td>float</td>
<td>CPU utilization skew across the cluster. Calculated as the cubed standard deviation of session_id:host:total_cpu from all hosts running a certain query.</td>
</tr>
<tr>
<td>session_id</td>
<td>resident_size_pct_skew</td>
<td>float</td>
<td>Resident memory utilization skew across the cluster. Calculated as the cubed standard deviation of session_id:host:total_resident_size_pct from all hosts running a certain query.</td>
</tr>
<tr>
<td>session_id</td>
<td>disk_read_bytes_per_sec_skew</td>
<td>float</td>
<td>Disk read skew across the cluster. Calculated as the cubed standard deviation of session_id:host:total_disk_read_bytes_per_sec from all hosts running a certain query.</td>
</tr>
<tr>
<td>session_id</td>
<td>disk_write_bytes_per_sec_skew</td>
<td>float</td>
<td>Disk write skew across the cluster. Calculated as the cubed standard deviation of session_id:host:total_disk_write_bytes_per_sec from all hosts running a certain query.</td>
</tr>
<tr>
<td>session_id:host</td>
<td>total_cpu</td>
<td>float</td>
<td>Total cpu utilization of all processes running a certain query on a host.</td>
</tr>
<tr>
<td>session_id:host</td>
<td>total_disk_read_bytes_per_sec</td>
<td>integer</td>
<td>Total disk read bytes-per-second of all processes running a certain query on a host.</td>
</tr>
<tr>
<td>session_id:host</td>
<td>total_disk_write_bytes_per_sec</td>
<td>integer</td>
<td>Total disk write bytes-per-second of all processes running a certain query on a host.</td>
</tr>
<tr>
<td>session_id:host</td>
<td>total_resident_size_pct</td>
<td>float</td>
<td>Total resident memory % of all processes running a certain query on a host.</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>xact_start</td>
<td>string</td>
<td>Time when this process’ current transaction was started</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>application_name</td>
<td>string</td>
<td>Name of the application that is connected to this backend</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>client_port</td>
<td>integer</td>
<td>TCP port number that the client is using for communication with this backend</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>client_addr</td>
<td>string</td>
<td>IP address of the client connected to this backend</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>backend_start</td>
<td>string</td>
<td>Time when this process was started, i.e., when the client connected to the server</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>usesysid</td>
<td>integer</td>
<td>OID of the user logged into this backend</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>datname</td>
<td>string</td>
<td>Name of the database this backend is connected to</td>
</tr>
<tr>
<td>Scope</td>
<td>Datum</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>datid</td>
<td>integer</td>
<td>OID of the database this backend is connected to</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>spillfile_count_across_cluster</td>
<td>integer</td>
<td>total number of spill files across the cluster created for this query</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>spillfile_size_across_cluster</td>
<td>integer</td>
<td>total size of spill files across the cluster created for this query</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>runtime</td>
<td>integer</td>
<td>Time since the query started</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>gpdb_segment_role</td>
<td>string</td>
<td>the current role of this GPDB segment (MASTER, SEGMENT, MIRROR)</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>query_start</td>
<td>string</td>
<td>Time when the currently active query was started</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>usename</td>
<td>string</td>
<td>Name of the user logged into this backend</td>
</tr>
<tr>
<td>session_id:host:pid</td>
<td>current_query</td>
<td>string</td>
<td>Text of this backend's current query</td>
</tr>
</tbody>
</table>