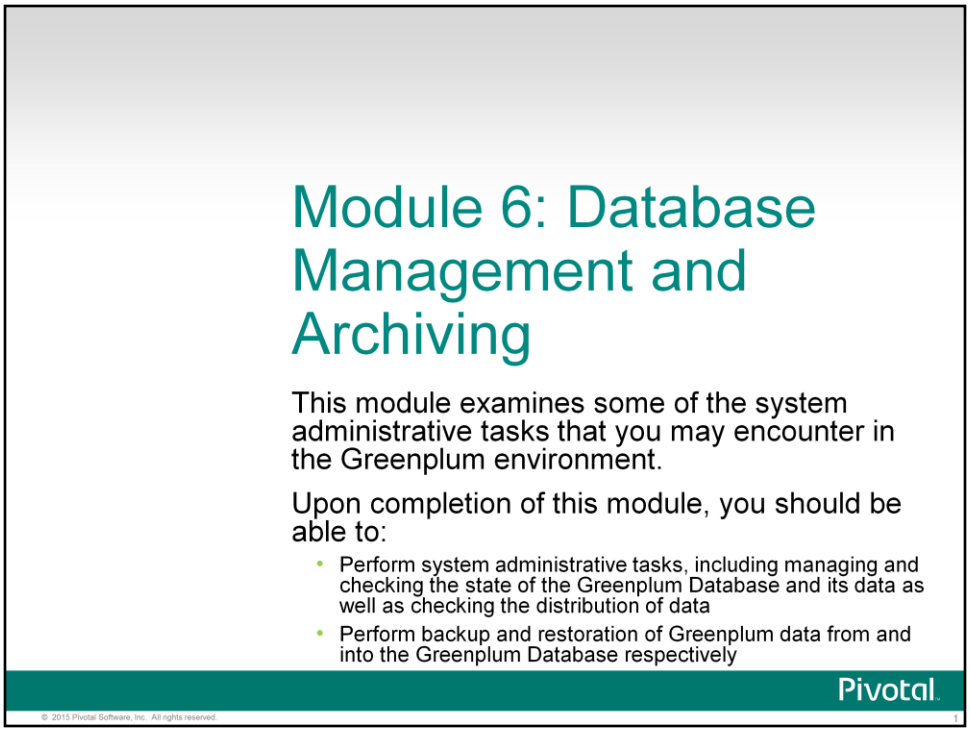
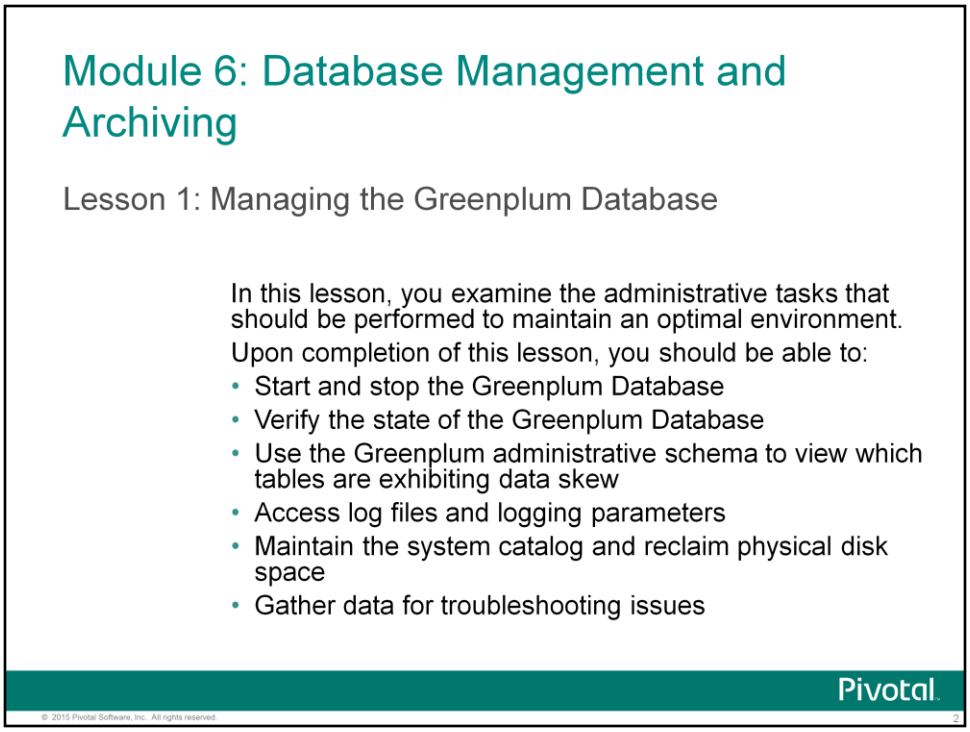


System maintenance is one of the key services that administrators provide for the environment they support. You may be required to make changes to parameters that require restarting the Greenplum environment. Monitoring the system requires knowing not only its current state, but also the state of any applications it supports. Data backup and restoration services are key to reducing the chances of data loss.

In this module, you will:

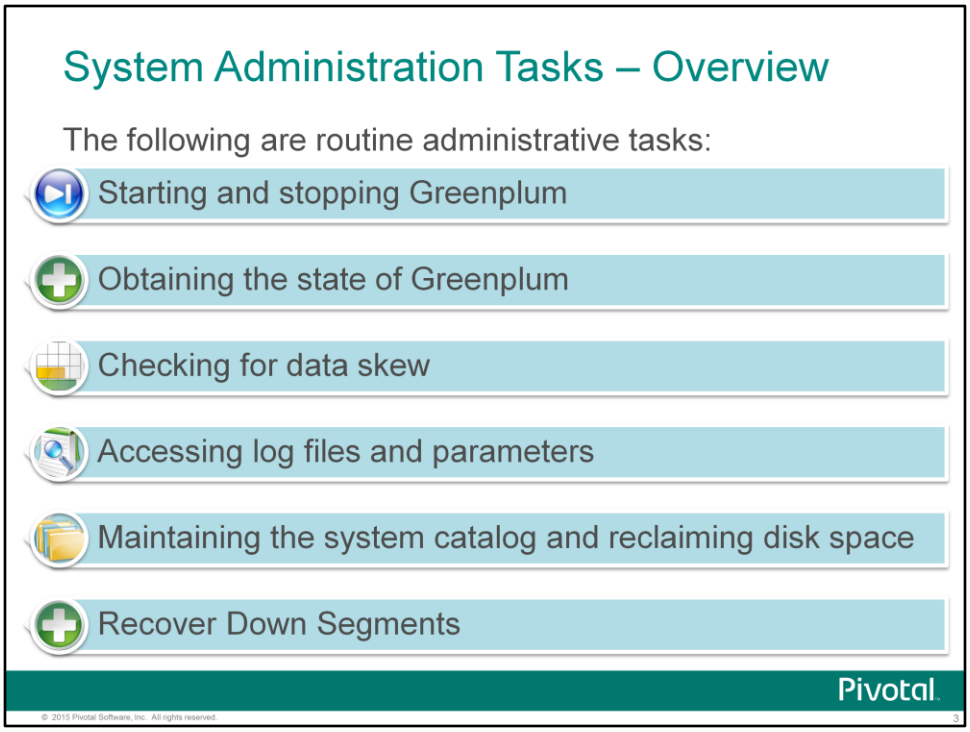
- Perform system administrative tasks for the Greenplum environment which includes managing and checking the state of the Greenplum Database as well as the state and distribution of data.
- Perform backup and restoration of Greenplum data from the Greenplum Database and back into the Greenplum Database.



As an administrator, you will encounter times where you must perform specific administrative tasks to keep the Greenplum Database environment running smoothly. You should be aware of how to start, stop, and restart the Greenplum Database and to keep track of the current state of the database. You may also be required to monitor the state of data and log files to maintain an optimal environment.

Upon completion of this lesson, you should be able to:

- Start and stop the Greenplum Database.
- Verify the state of the Greenplum Database using Greenplum commands.
- Use the Greenplum administrative schema to view which tables are exhibiting data skew.
- Access log files and logging parameters.
- Maintain the system catalog and reclaim physical disk space.
- Gather data that can be used for troubleshooting the environment should an issue arise.

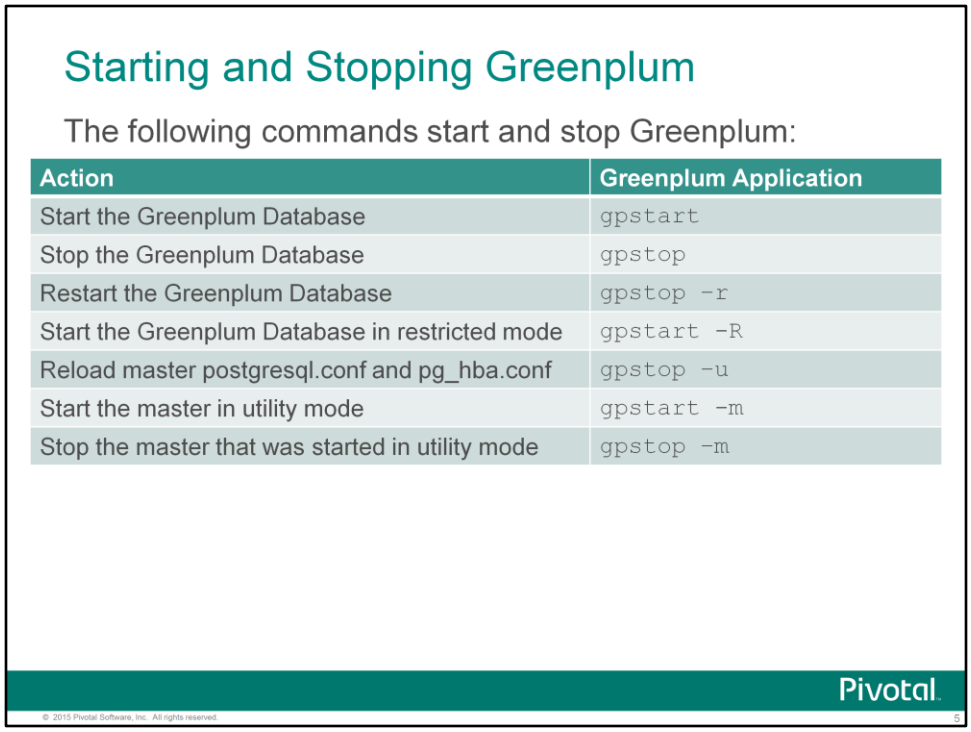


There are several routine tasks that you will perform while maintaining the Greenplum Database. These include:

- **Starting and stopping the Greenplum Database** – Changing a global parameter may necessitate a restart of the Greenplum Database. Because the Greenplum Database is distributed across segments and the master server, it is necessary to understand the shutdown and startup procedures before undertaking this task.
- **Obtaining the state of Greenplum** – A segment failure may go undetected if you do not have notifications configured and mirrors are enabled. Checking the state of the Greenplum environment lets you to verify the state of components and processes.
- **Checking for data skew** – Data skew can impact the performance of queries submitted to the Greenplum Database. Using the Greenplum administrative schema, `gp_toolkit`, you can verify how data is distributed across the segments. There are other commands that allow you to achieve that same goal.
- **Accessing log files and parameters** – Log files act as your audit trail for how components within the Greenplum Database are behaving. You must be aware of how to access these log files as part of your monitoring tasks as well as how to rotate them, if necessary.

System Administration Tasks – Overview

- **Maintaining the system catalog and reclaiming disk space** – Numerous database drops and create can cause growth in the size of the system catalog that could affect performance. Regular maintenance helps optimize system performance. If data is constantly changing, vacuuming tables should be on the list of tasks that you regularly perform.
- **Recover Down Segments**– In a system with mirrors enabled, the gprecoverseg utility reactivates a failed segment instance and identifies the changed database files that require resynchronization. Once gprecoverseg completes this process, the system goes into *resynchronizing* mode until the recovered segment is brought up to date. The system is online and fully operational during resynchronization.



In a Greenplum DBMS, each database server instance must be started or stopped across all of the hosts in the system in such a way that they can all work together as a unified DBMS. This is because the Greenplum Database system is distributed across many machines and database instances.

Greenplum provides the following commands to change the state of the database:

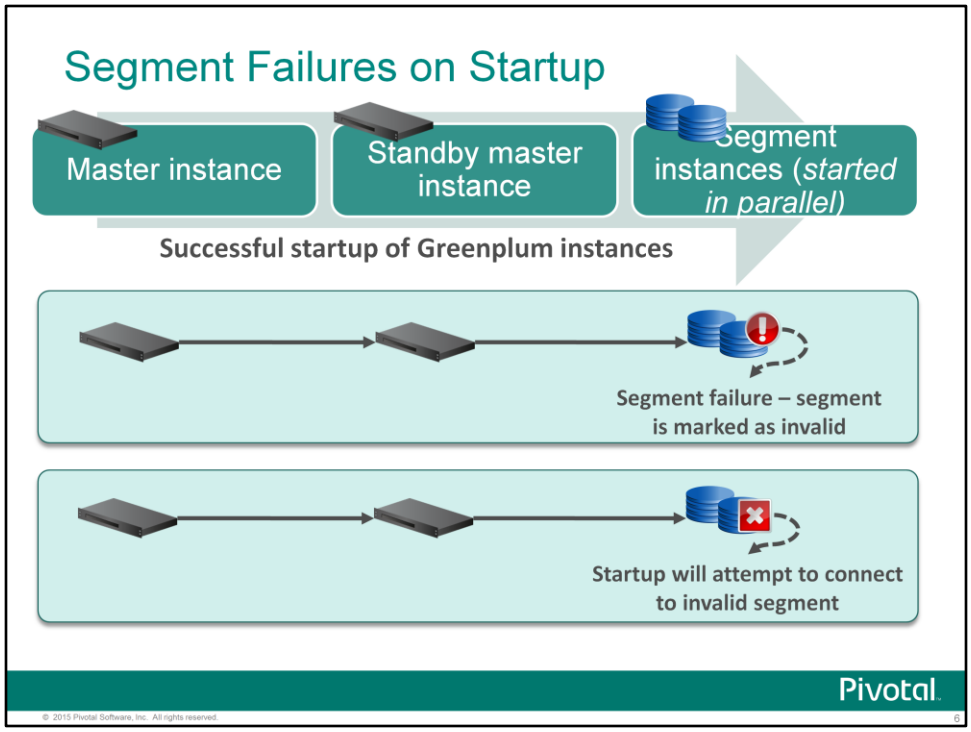
- **`gpstart`** – This command starts all instances in the array in the correct order.
- **`gpstop`** – This command stops all instances in the array in the correct ordering.

These scripts call the postmaster process on each segment instance with the correct flags. The master instance is always started last and is shut down first.

There may be cases where you want to start or stop only the master. This is called utility mode. In utility mode, you can:

- Connect to a database on the master instance only.
- Edit system settings in the system catalog, without affecting user data on the segment instances.

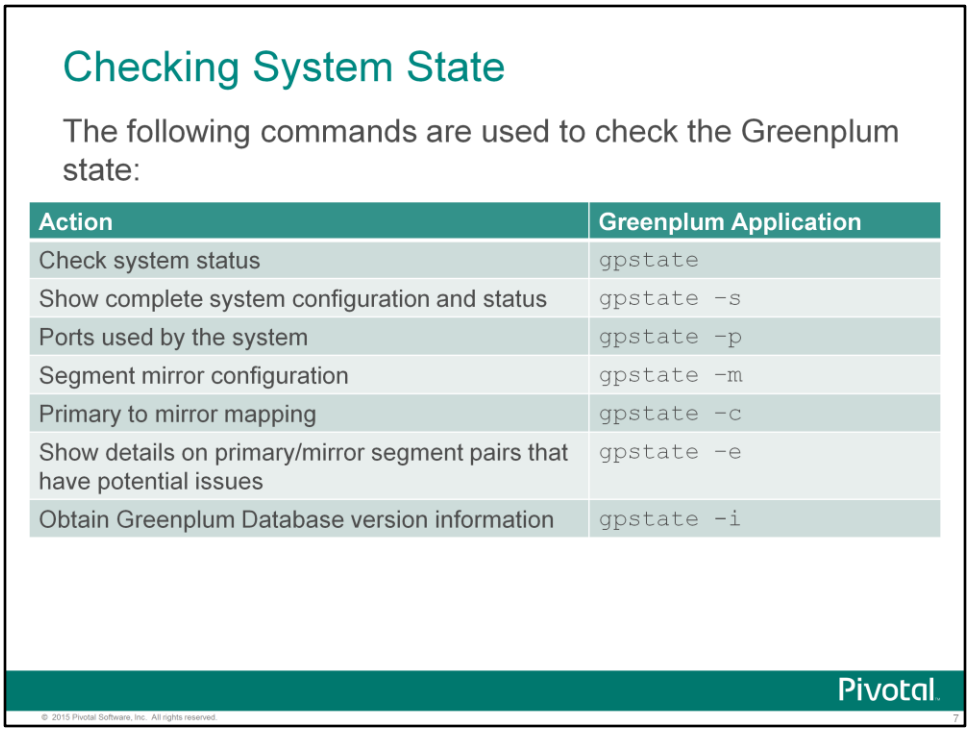
This is a very rare occurrence, but may be necessary in some situations, such as when reconfiguring the array to use different hosts.



If at startup a segment instance is not reachable:

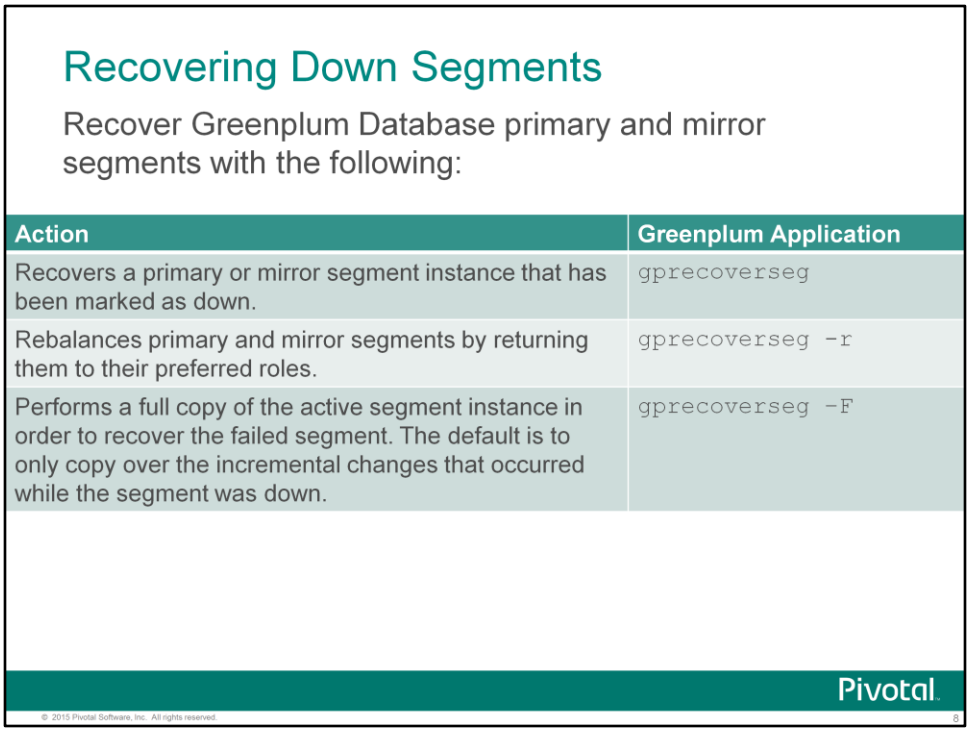
- The system will mark it as invalid, meaning that the segment instance is offline and is not operational.
- The startup process skips segments marked as invalid and does try to start the server process, `postmaster`, on invalid segments.

This is a protective measure in case the segment instance needs some administrative action before it can safely be made operational again. When troubleshooting failed segment instances, you may at times need to force a startup of segment instances marked as invalid. For example, if a simple error, such as incorrect file permissions or an offline host, prevents startup of a segment instance, you will want to fix the problem that caused the error in the first place, and then retry the startup. Once the segment instance is up and running then it will be marked valid again next time the master tries to connect to it.



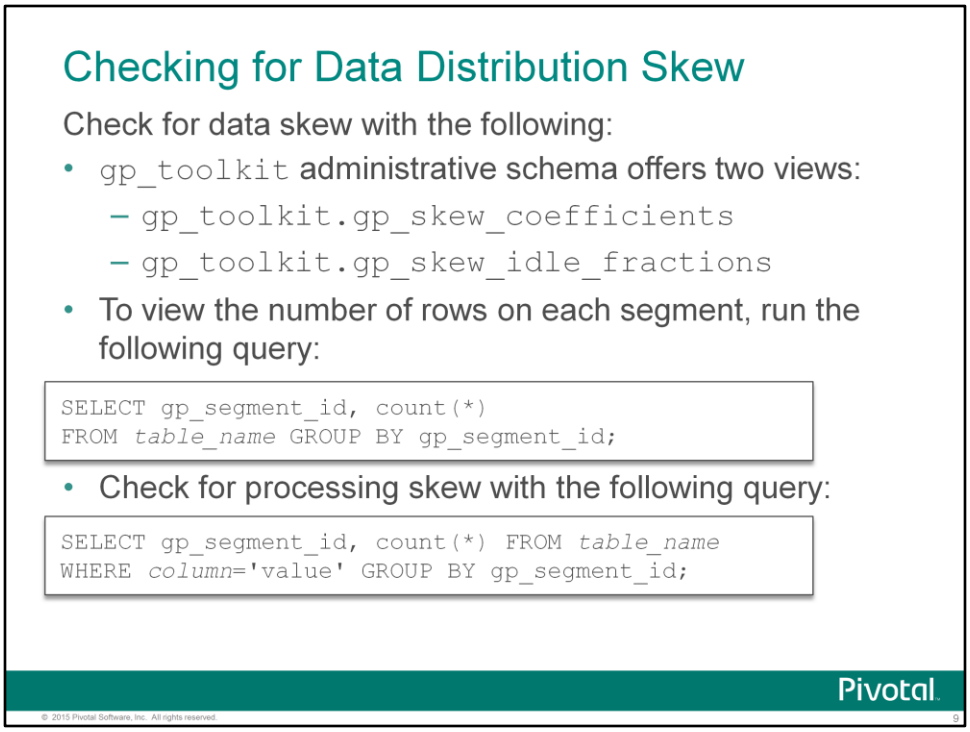
The `gpstate` command displays status information about the Greenplum Database system, including:

- Current system configuration consisting of the number of configured segments, segment ports, hosts, data directories, primary/mirror status, mapping of the primary segment to its respective mirror segment.
- Segments that are marked as invalid.
- Mirror segment status.



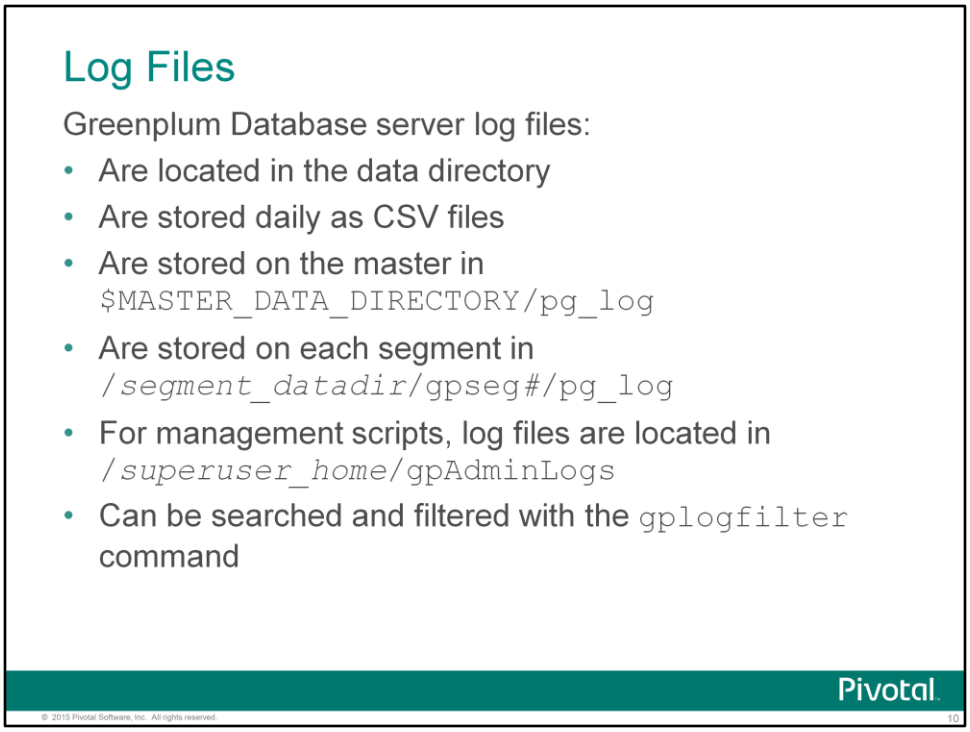
In a system with mirrors enabled, the gprecoverseg utility reactivates a failed segment instance and identifies the changed database files that require resynchronization. Once gprecoverseg completes this process, the system goes into *resynchronizing* mode until the recovered segment is brought up to date. The system is online and fully operational during resynchronization.

A segment instance can fail for several reasons, such as a host failure, network failure, or disk failure. When a segment instance fails (primary or mirror), its status is marked as *down* in the Greenplum Database system catalog. In the case of a primary segment failure, its mirror is activated as the primary in *change tracking* mode. In order to bring the failed segment instance back into operation again, you must first correct the problem that made it fail in the first place, and then recover the segment instance in Greenplum Database using gprecoverseg.



Performance can be inhibited if data is not equally distributed across all active segments. You can use any or a combination of the following to check for data skew:

- The gp_toolkit schema provides two views that can be used to identify data skew.
  - The `skccoef` column of the `gp_skew_coefficients` view is calculated as the standard deviation divided by the average. A higher value indicates greater data skew. The view also provides the table's name and namespace.
  - The `siffraction` column of the `gp_skew_idle_fractions` view gives a percentage of the system that is idle during a table scan. This is an indicator of uneven data distribution or query processing skew. A value of .1 indicates a 10% skew. Tables with more than a 10% skew should be reevaluated.
- Data distribution can be viewed with the following SQL command:
  `SELECT gp_segment_id, count(*) FROM table_name GROUP BY gp_segment_id;`
  This lists the number of rows on each segment. If all of the segments have roughly the same number of rows, the table is considered to be balanced.
- Processing skew, or the amount of time one query spends on acquiring data over another, can be determined with the following command:
  `SELECT gp_segment_id, count(*) FROM table_name WHERE column='value' GROUP BY gp_segment_id;`
  This returns the number of rows returned by segment for the given `WHERE` clause.



The log files for each Greenplum database server instance can be found in the data directory of that instance. If you are not sure of the location of the data directories, you can use `gpstate -s` to determine the location of the log files.

Log files for the Greenplum management scripts are written to the `gpAdminLogs` directory in the home directory of the superuser by default. These logs are on the master only. The log file for a particular script execution is appended to its daily log file each time that script is run.

Greenplum Database uses Postgres write ahead logging (WAL) for transaction logging on the master and each segment instance. WAL is on automatically. Refer to PostgreSQL documentation has more information about WAL.

Logging Configuration Parameters

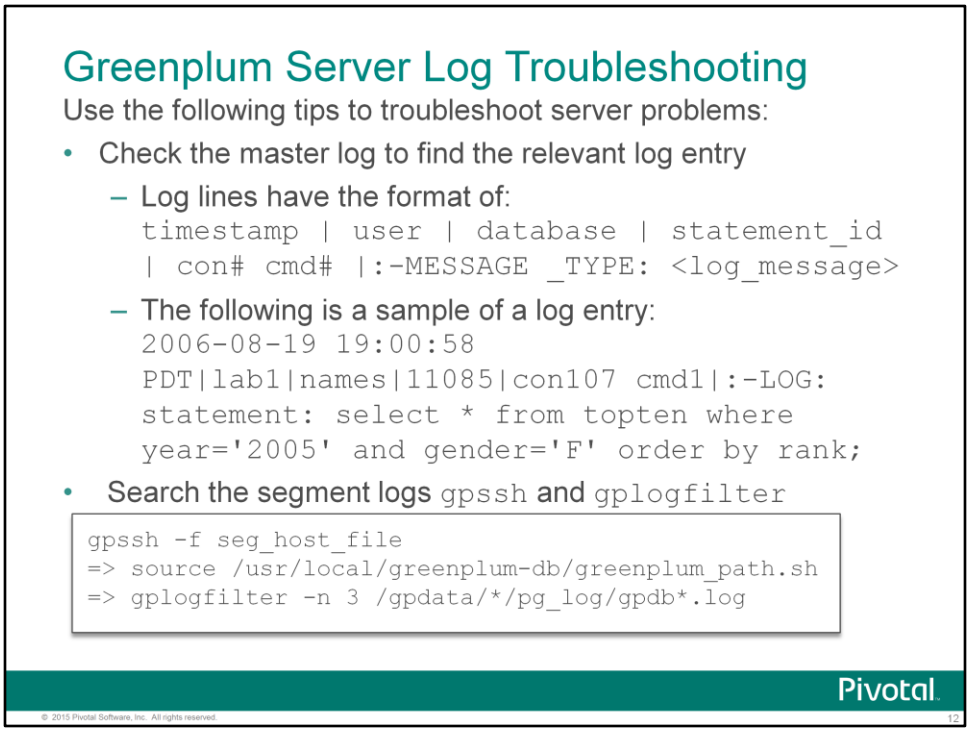
The following are commonly used for log configuration:

Log Parameter	Description
`client_min_messages`	Controls which message levels are sent to the client; accepts range from `DEBUG5` to `PANIC`
`log_min_messages`	Controls which message levels are written to the server log; accepts range from `DEBUG5` to `PANIC`
`log_connections`	Each successful connection is logged; accepts `on|off`
`log_statement`	Controls which SQL statements are logged; accepts `NONE`, `DDL`, `MOD`, or `ALL`
`log_rotation_age`	Determines the maximum lifetime of an individual log file; accepts values in minutes
`log_rotation_size`	Determines the maximum size of an individual log file; accepts values in kilobytes
`log_duration`	Causes the duration of every completed statement to be logged; useful for query profiling

Pivotal

The slide lists several configuration parameters to set log levels, connection attempts, log rotation age and size, and query duration. These are numerous other parameters that can be used to fine tune logging to your requirements.

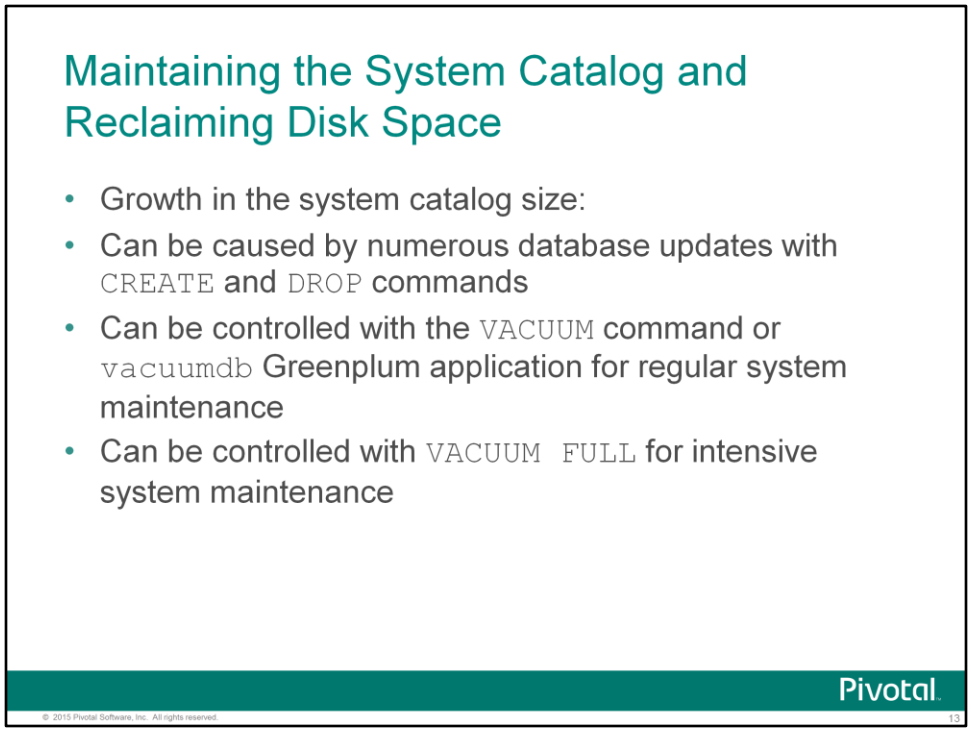
Refer to the *Greenplum Database Administrator Guide* for the complete list and details of each logging parameter.



To investigate an error or a log entry:

- First examine the master log file. This log file contains the most information.
- Each entry in the log has this information by default:
    - timestamp, user, database
    - *statement_id* – Each statement issued to the Greenplum Database is assigned a *statement_id*. All processes working on the same statement will have the same statement ID. The master increments this ID each time it receives a new SQL statement from the client.
    - `con#` – This number identifies a client session and connection.
    - `cmd#` – This is the command count number of the related session. Message type can be one of `PANIC`, `FATAL`, `ERROR`, `WARNING`, `NOTICE`, `INFO`, `DEBUG1`, `DEBUG2`, `DEBUG3`, `DEBUG4`, `DEBUG5`. It is then followed by the actual log message text.
- Use the *statement_id*, connection number, and command count to find any related log entries on the segment instances. Note that a master log entry will not always have related segment entries.

The `gplogfilter` utility provides a way to search all segment logs at once.



Numerous database updates with `CREATE` and `DROP` commands cause growth in the size of the system catalog. This can affect system performance.

For example, after a large number of `DROP TABLE` statements:

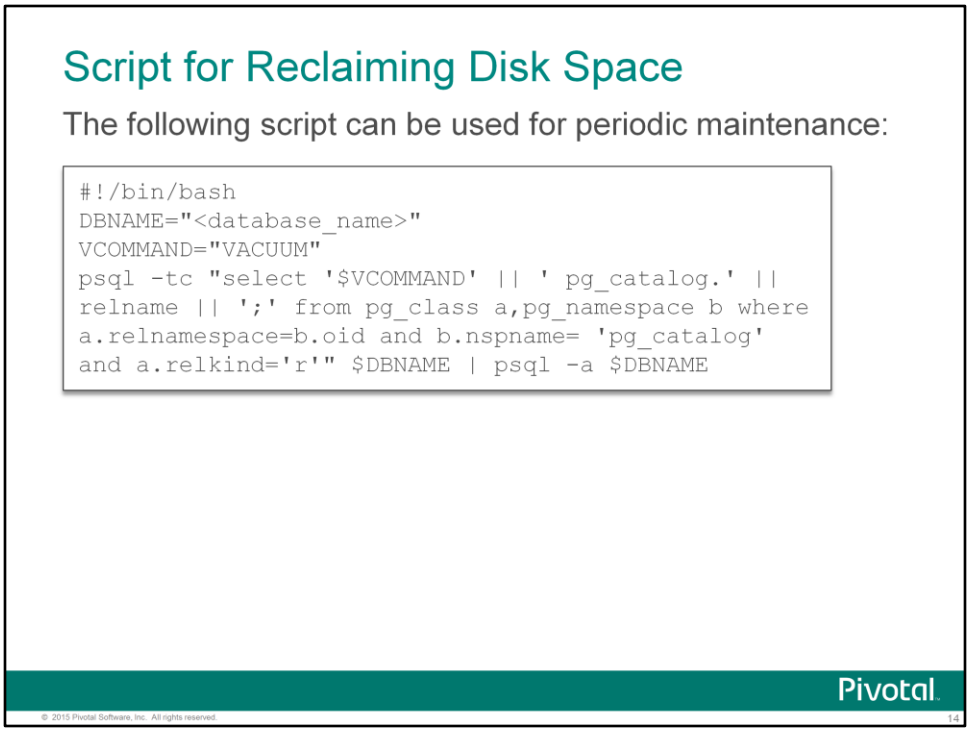
- The overall performance of the system begins to degrade due to excessive data scanning during metadata operations on the catalog tables.
- The performance loss may occur between thousands to tens of thousands of `DROP TABLE` statements, depending on your system.

Greenplum recommends that you regularly run a system catalog maintenance procedure to reclaim the space occupied by deleted objects. If a regular procedure has not been run for a long time, you may need to run a more intensive procedure to clear the system catalog.

Periodically use `VACUUM` on the system catalog to clear the space occupied by deleted objects. This can be executed while the system is up, but should be run during a quiet period at off-peak hours.

A `VACUUM FULL` can be performed if regular system catalog cleaning is not performed regularly and the catalog has become bloated with dead space. This must be performed during system down time due to exclusive locks in use against the system catalog.

Periodic vacuuming reclaims unused space. `VACUUM FULL` is not recommended in large databases. It locks all tables and can take a really long time. A regular `VACUUM` is sufficient and is best to do at low-usage times. `ANALYZE` makes sure the planner always has the latest database statistics.



The script shown can be used to dynamically obtain the list of system tables found in the catalog for a specified database and execute a `VACUUM` against these tables. The script can be executed on a regular basis to remove bloat from the system catalog and remove the need to execute a `VACUUM FULL` on catalog tables. This is particularly useful for databases that are frequently updated, where tables are added and removed frequently.

Checking Database Object Sizes and Disk Space

`gp_toolkit` schema views for disk usage (in bytes):

Action	gp_toolkit View
Total size of all indexes for a table	`gp_size_of_all_table_indexes`
Size of a database	`gp_size_of_database`
Total size of an index	`gp_size_of_index`
Size of schemas in this database	`gp_size_of_schema_disk`
Disk size of a table	`gp_size_of_table_disk`
Uncompressed table size for append-only tables	`gp_size_of_table_uncompressed`
Amount of disk free, as determined by the OS `df` command	`gp_disk_free`

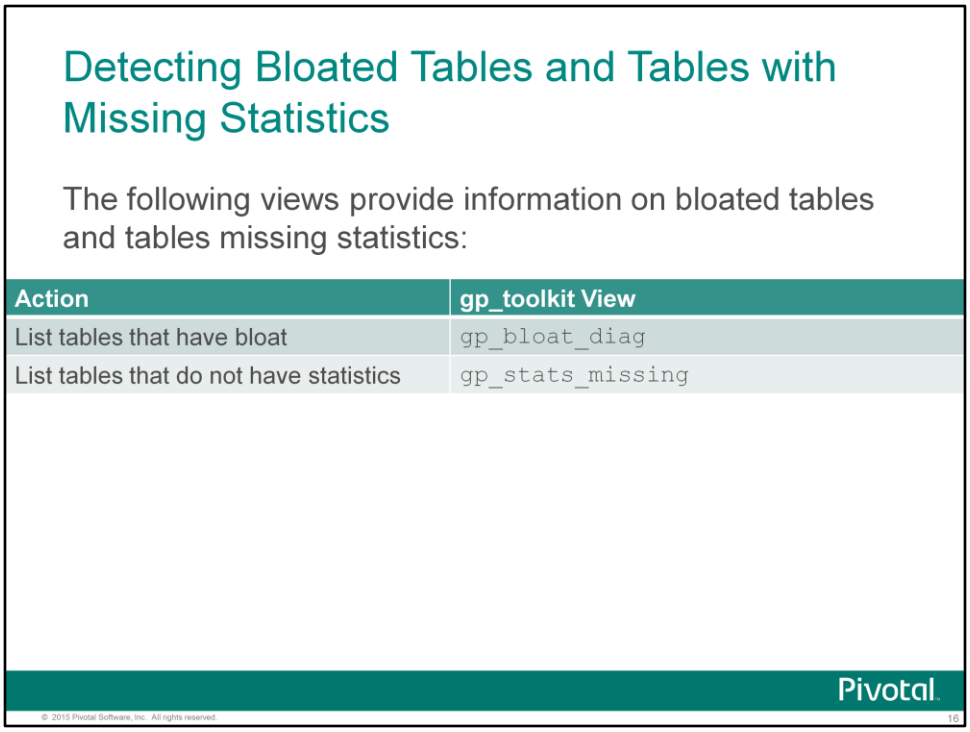
Note: Objects are shown by their object IDs. Look up the relation name in the `pg_class` table.

Pivotal

Monitoring disk space ensures that you have enough storage space on your segment hosts as your data grows. The `gp_size_*` family of views available from the `gp_toolkit` administrative schema, can be used to determine the disk space usage for a Greenplum database, table, schema, or index.

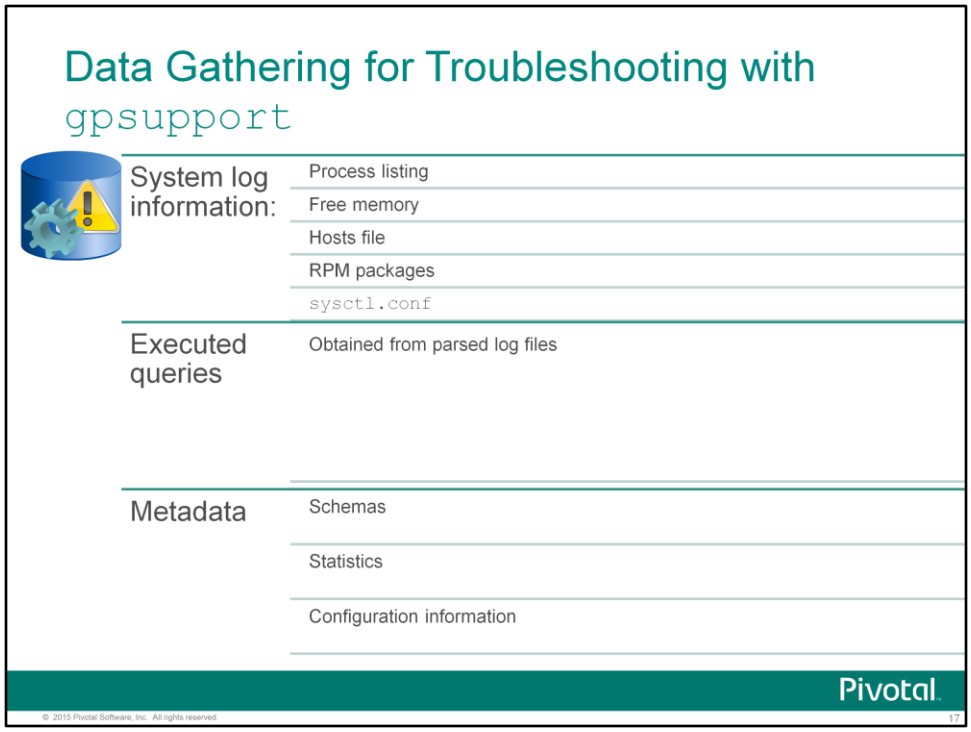
Objects are listed by their object ID and not by name. To check the size of a table or index by name, you must look up the relation name in the `pg_class` table. For example:

```
SELECT relname as name, sotdsize as size, sotdtoastsize
as toast, sotdadditionalsize as other
FROM gp_toolkit.gp_size_of_table_disk as sotd,
pg_catalog.pg_class
WHERE sotd.sotdoid=pg_class.oid ORDER BY relname;
```



Bloated tables or tables that do not have statistics can adversely affect query performance. It is important to use `VACUUM` or `VACUUM ANALYZE` to clean bloated tables and clean and generate statistics respectively. The following `gp_toolkit` views provide information on the tables with regards to bloat and statistics:

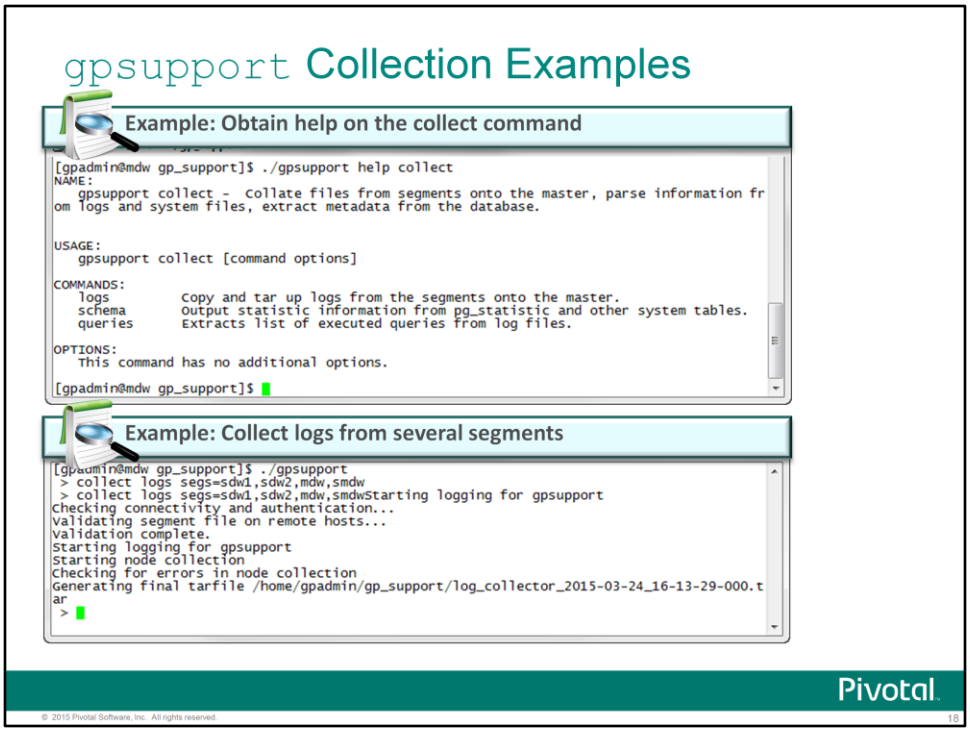
- The **`gp_bloat_diag`** view lists the tables that have bloat. A table is bloated when the actual number of pages on disk exceeds the expected number of pages given the table statistics. Use `VACUUM` to reclaim disk space occupied by deleted or obsolete rows. This view is accessible to all users, however non-superusers will only be able to see the tables that they have permission to access.
- The **`gp_stats_missing`** lists tables that do not have statistics and therefore may require an `ANALYZE` or `VACUUM ANALYZE` be run on the table. Empty tables do not have statistics associated with them and will be displayed when selecting this view.



The `gpsupport` utility is a separate downloadable g-zipped compressed binary used to gather Greenplum Database information. The gathered data is used to troubleshoot and diagnose issues for GPDB 4.2.x and 4.3.x in a Linux operating system environment. The collected information can be provided to support personnel to help troubleshoot the environment.

Information is collected from the masters and segment servers and is stored to the master. The type of information or data that can be collected includes:

- Log files from segments, including, process listings, memory information, the hosts file, a listing of the packages installed on the system, and the contents of the `sysctl.conf` file. Information is collected from all segments you specify, master, standby, and segment servers, either directly on the command line or from a specified hosts file. The utility pushes an agent to the segment servers to collect information. This agent, `gpsupport_segment`, is pushed to the `/tmp` directory by default. Its location can be changed however.
- Executed queries collected from parsing log files. By default, the log files are collected from `/data/master/gpseg-1`.
- Metadata from the database, including schemas, statistics, and configuration information. This includes global objects, database statistics information collected from the `pg_statistic` catalog table, and planner statistics information collected from the `pg_class` catalog table.

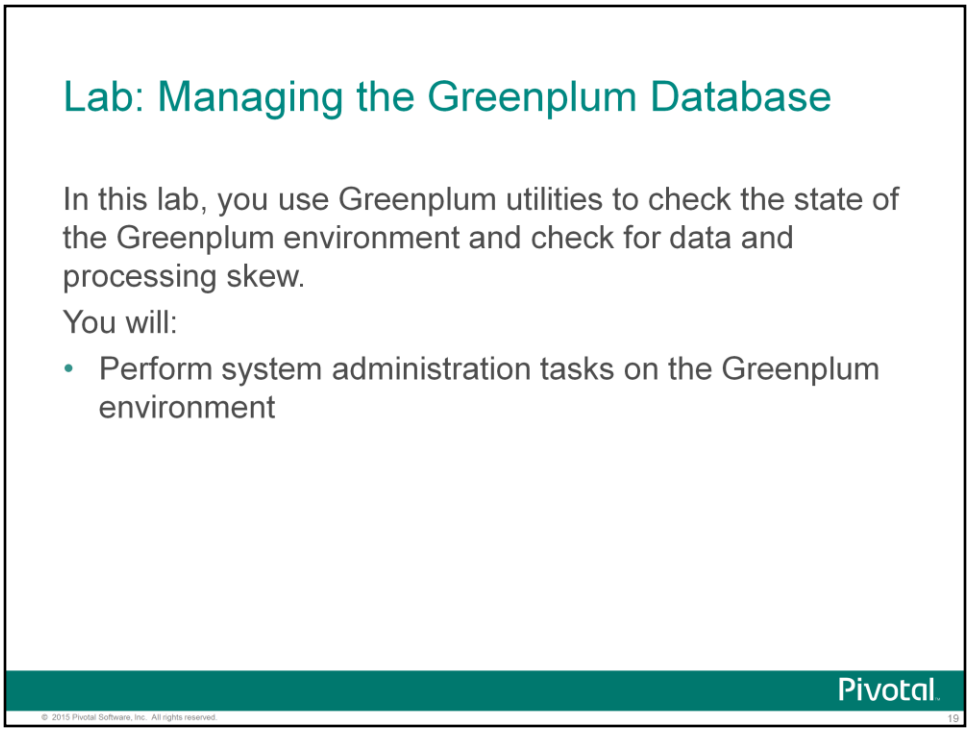


`gpsupport` provides both an interactive interface and a command line interface. Command usage is obtained with the help command. It can also be used to obtain additional information on other commands as shown in the first example where help is obtained on the `collect` command.

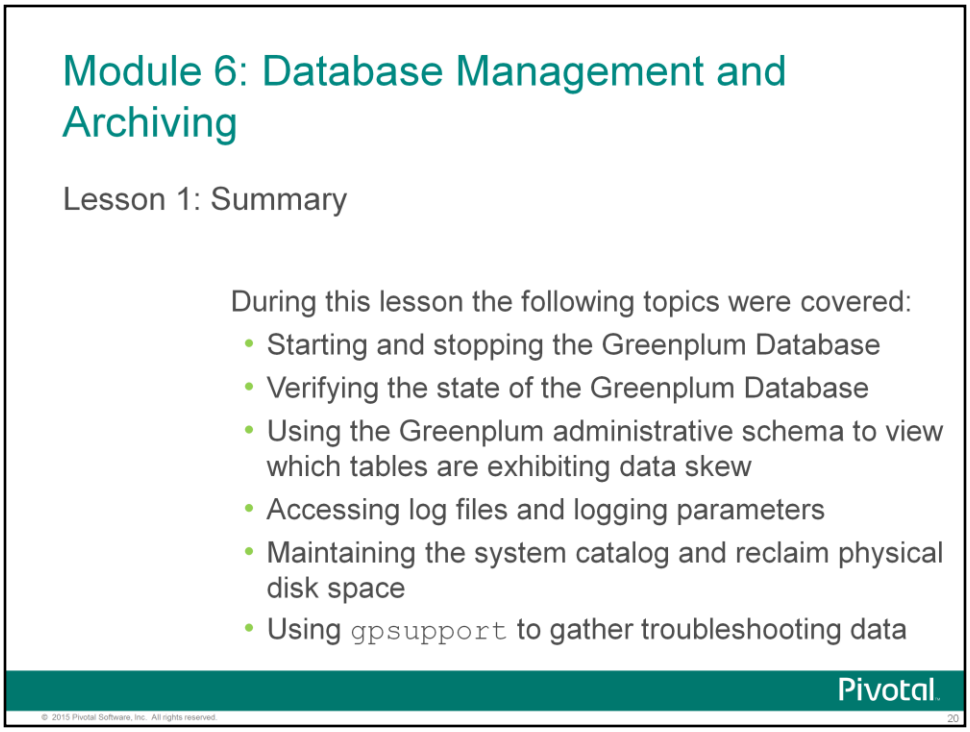
In the second example, `gpsupport` is executed in interactive mode. The `collect logs` command is executed with the `segs` option which allows you to specify which segment servers to collect information from. This is a comma separated list and overrides the `hostfile` global option which is used to specify the location of a host file. The host file requires that each host be listed on a line by itself.

The logs are saved to the directory from which you have executed the `gpsupport` command. You will require read and write permission to the directory you use for saving the captured information. You can specify a different output directory by using the `masterDataDir` option to this command. For example, to save the output from the second example to the `/tmp` directory, use the following command:

```
$ gpsupport collect logs segs=sdw1,sdw2,mdw,smdw
masterDataDir=/tmp
```

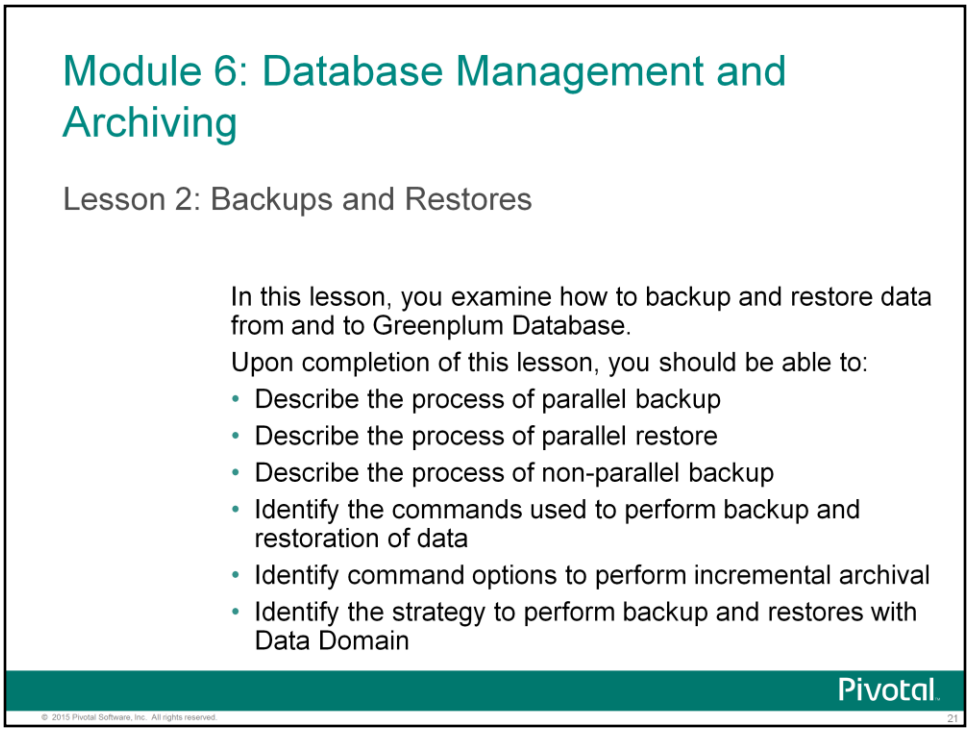


In this lab, you use Greenplum utilities to check the state of the Greenplum environment and check for data and processing skew.



This lesson covered several database management tasks, including starting, stopping, and verifying the state of the Greenplum Database. The administrative schema lets you find the tables that are exhibiting data skew, which can impact overall performance when querying tables. Other administrative tasks include accessing log files and logging parameters as well as maintaining the system catalog and reclaiming physical disk space, particularly in environments where there are a large number of deletes or updates for both system tables and user tables.

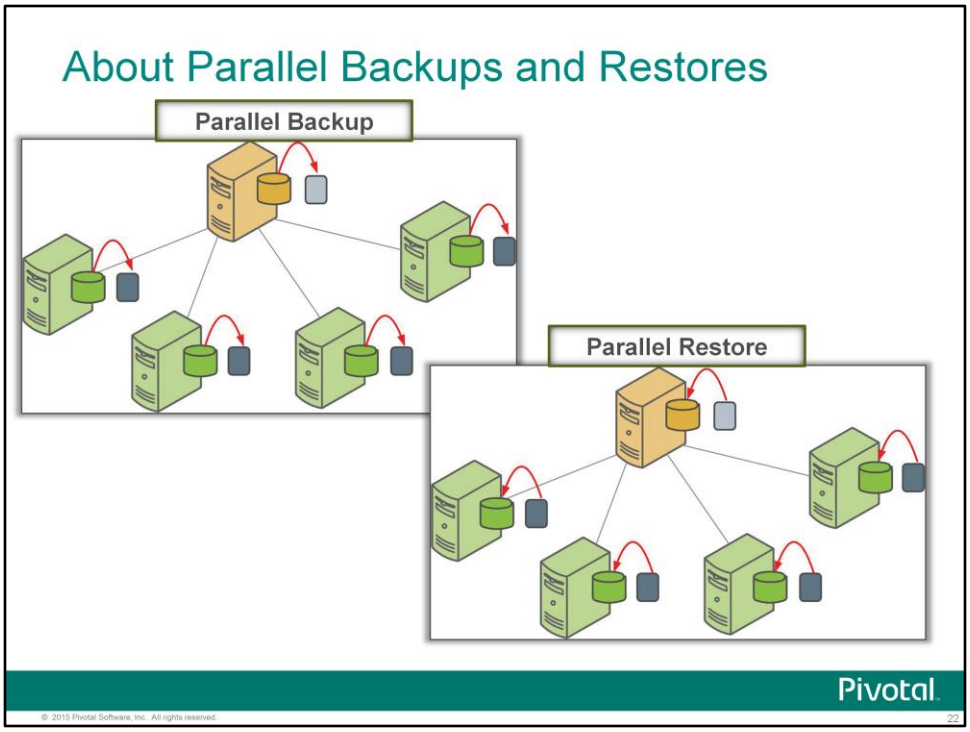
Troubleshooting involves gathering information from several different areas to analyze the underlying issue. The gpsupport utility is used to collate data from named segments from logs, metadata, system processes and information to create a complete picture of the system.



Regular backups improve the chances of recovering data in the event of a system failure or data corruption. Backups can also be used to migrate data from one system to another.

In this lesson, you will:

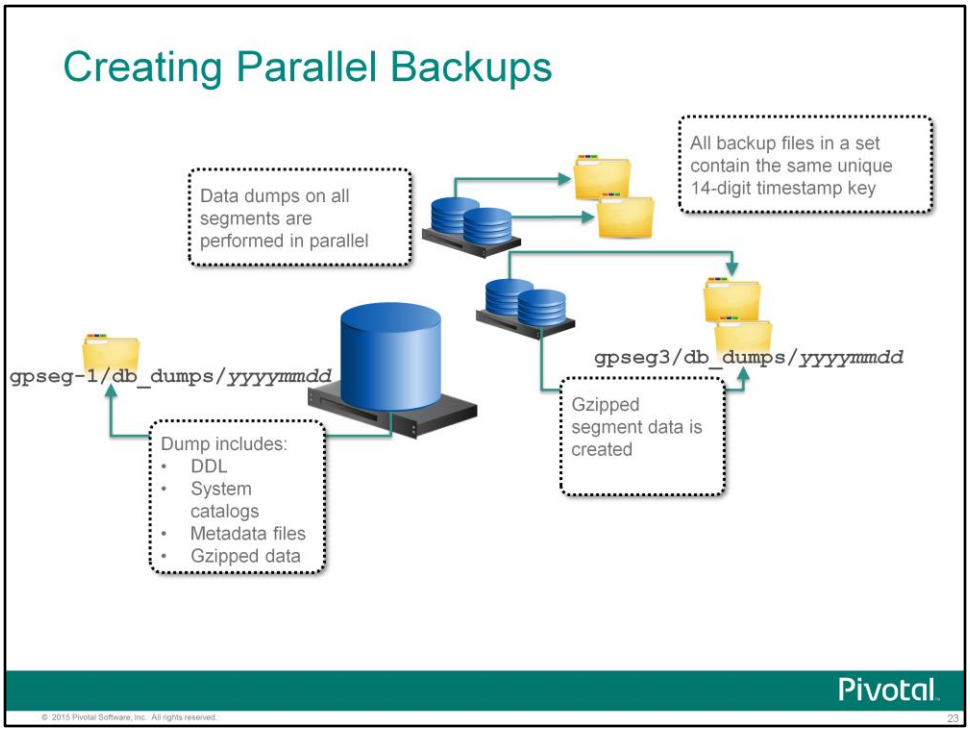
- Describe the process of a parallel backup.
- Describe the process of a parallel restore.
- Describe the process of a non-parallel backup.
- Identify the commands used to perform backup and restoration of data.
- Identify the options to support incremental backup and restore of append-only tables
- Identify the overall strategy for performing backup and restore with Data Domain.



In Greenplum Database:

- You initiate a parallel dump operation from the master which dumps DDL for the database, schemas, and system catalog.
- This starts dump agents on the segments to dump all segment databases in parallel.

Likewise during a parallel restore, restore agents load each of the segment databases in parallel from their corresponding dump file as well as the data definitions for the master.



When performing a parallel backup, several actions occur:

- Every primary segment is dumped in parallel.
- A single dump file is created in the data directory for each of these segments. Within the data directory for the primary segment, two additional levels are created: `db_dumps/<yyyymmdd>`. The dump file created for the segment contains the data for an individual segment instance.
- The backup files are identified by a unique 14-digit timestamp key as part of the filename. All files within the same backupset are identified by this unique identifier.
- The master dumps the configuration of the database, or the system catalog, as well as DDL statements for recreating the database and schemas. This is saved to the master data directory under the same hierarchy.

As the dump files are created on the local file system for each segment, you must ensure there is sufficient disk space for the dump files both on the master and the primary segments.

Dump Files Created During Parallel Backup

Master Segment Dump File	Description
gp_cdatabase_1_<*dbid*>_<*timestamp*>	CREATE DATABASE statement
gp_dump_1_<*dbid*>_<*timestamp*>	Database schemas
gp_dump_status_1_<*dbid*>_<*timestamp*>	Log file
gp_dump_1_<dbid>_<timestamp>_post_data	Post database setup (GUCs and search paths)
gp_dump_1_<*dbid*>_<*timestamp*>_ao_state_file	List of append-optimized tables
gp_dump_1_<*dbid*>_<*timestamp*>_co_state_file	List of column-oriented tables
gp_dump_<*timestamp*>.rpt	Database dump report

Primary Segments Dump File	Description
gp_dump_0_<*dbid*>_<*timestamp*>	Data for the segment
gp_dump_status_0_<*dbid*>_<*timestamp*>	Log file

Note: Each backup set shares the same unique timestamp. This timestamp is required for restoring a backup set.

Pivotal

By default, the dump files are created under the data directory of the instance that was dumped. Files are saved to <*data_directory*>/db_dumps/*yyyymmdd*.

On the master, dump files of the following are created:

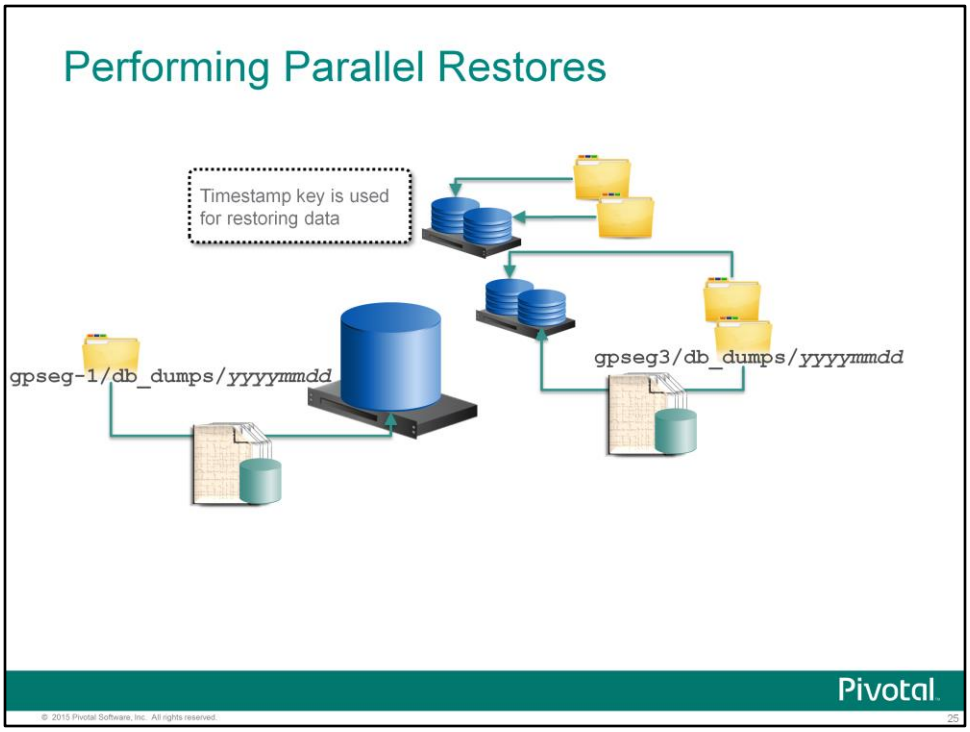
- CREATE DATABASE statement
- DDL to recreate schema and database objects
- Log or status of the backup
- Post-database creation operations
- State files for append and column-oriented tables
- Report file that details the status of each segment with regards to the backups.

On the segments, dump files of the following are created:

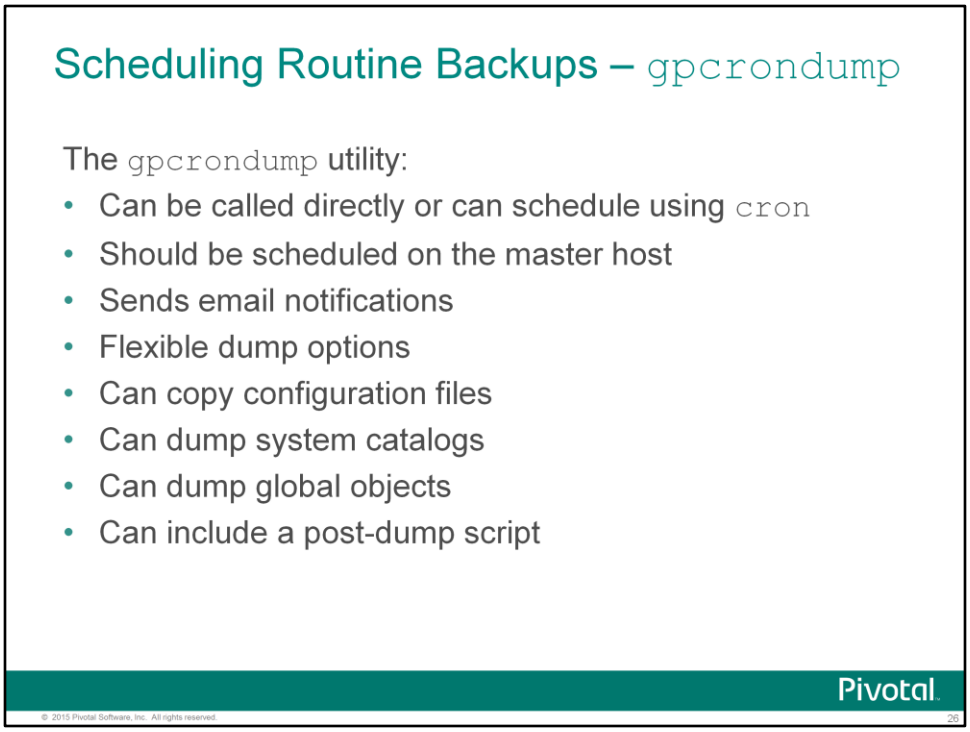
- COPY statements and user data
- Log or status of the backup

You can identify which instance a dump file belongs to by the *dbid* in the dump file name. The master instance is always 1_<*dbid*>, with the *dbid* usually being a 1 for the primary master. The segment instances are usually 0_<*dbid*>. This is assigned in the gp_configuration system catalog table.

Note that each dump file has the timestamp key which identifies the backup set. This timestamp is needed by a restore to identify the backup set.



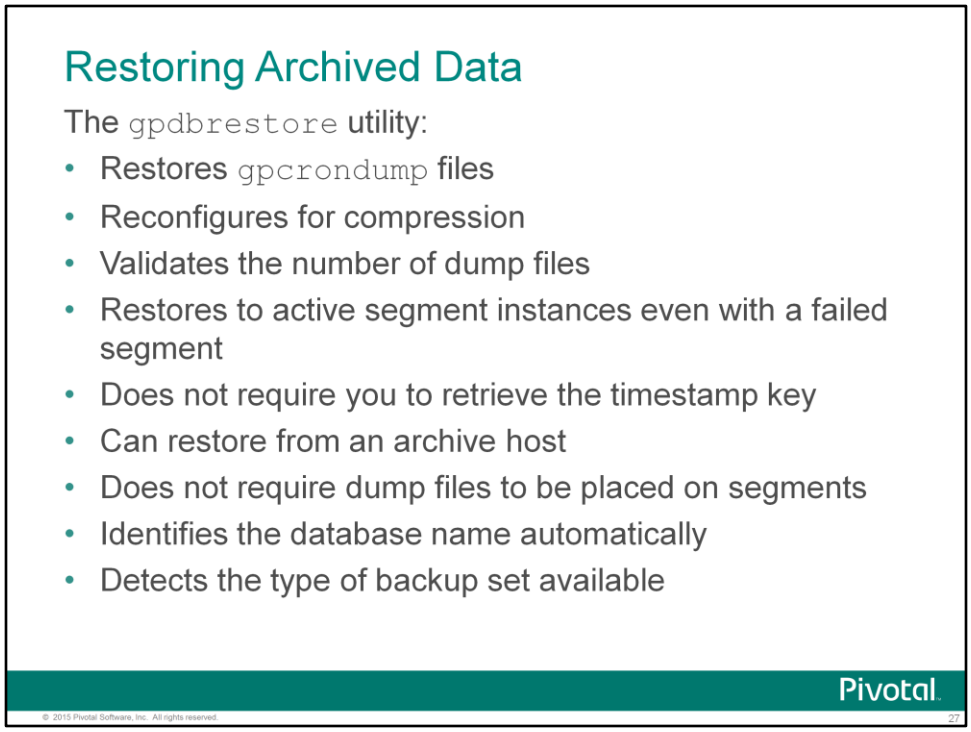
A parallel restore takes all the files associated with a backupset and restores the database objects and data to the distributed database in parallel. The unique timestamp key generated during the parallel backup is used to define the backupset. The data is validated and restored to the master and primary segments for the database captured in the backupset.



Scheduling routine backups is a recommended practice to implement for disaster recovery sites. This includes automating nightly backups, moving backup files to a disaster recovery site, and providing a quick restoration option if needed.

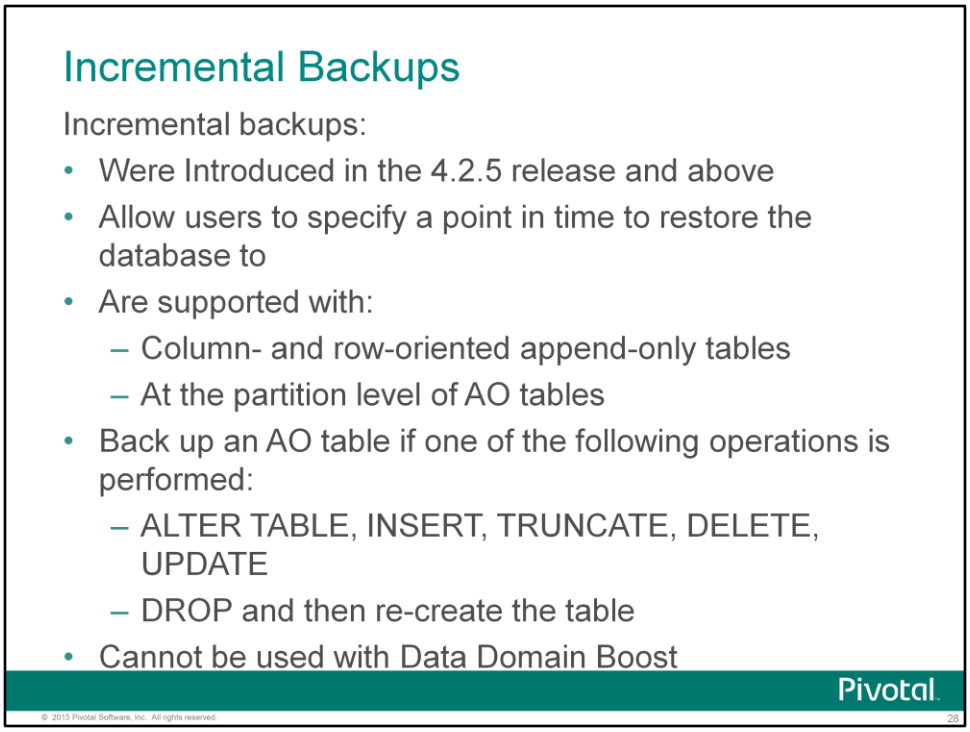
The `gpcrondump` utility:

- Allows you to schedule routine backups of a Greenplum database using `cron`, a scheduling utility for Unix operating systems.
- Should be scheduled on the master host.
- Sends email notifications on the status.
- Provides a multitude of flexible dump options and checks.
- Copies the configuration files.
- Can dump only the system catalog.
- Can dump only the global objects.
- Can include a post-dump script for rebuilding the data.



The `gpdbrestore` utility provides the following additional functionality on top of `gp_restore`:

- Restores from backup files created by `gpcrondump`.
- Automatically reconfigures for compression.
- Validates the number of dump files are correct for primary only, mirror only, primary and mirror, or a subset consisting of some mirror and primary segment dump files.
- If a failed segment is detected and if `gp_fault_action=continue`, restores to active segment instances.
- Does not need to know the complete timestamp key of the backup set to restore. Additional options are provided to instead give a date, directory, or database name to restore.
- Lets you restore from a backup set located on a host outside of the Greenplum Database array (archive host). To do this, it ensures that the correct dump file goes to the correct segment instance.
- Identifies the database name automatically from the backup set.
- Allows you to drop the target database before a restore in a single operation.



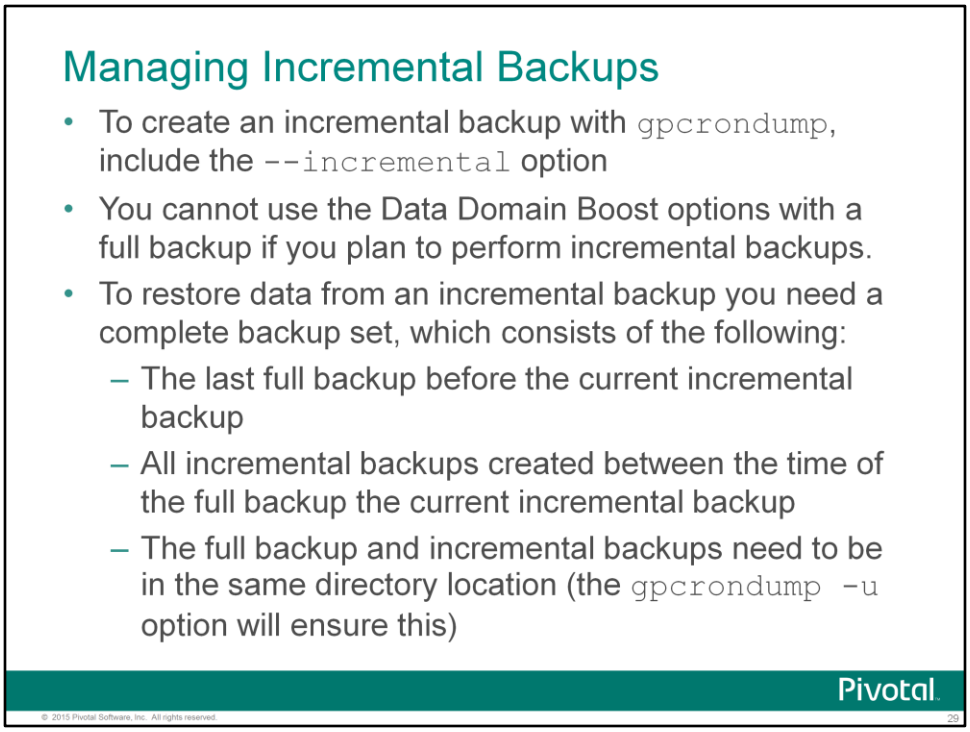
Starting in GP 4.2.5, options for incremental backup for append-optimized (AO) tables have been added to your support strategy. All other tables and database objects are backed up in a full backup mode.

Incremental backups are supported on both column-oriented and row-oriented AO tables. Support for incremental backups is also provided on AO tables that are partitioned.

When an incremental backup is executed on an AO partitioned table, only the partitions that have been modified will be captured. Changes to the table that are captured by incremental backups are tables that have been modified with one of the following:

- Alter table
- Insert
- Table truncate
- Dropping and recreating the AO table.
- Updating the table
- Deleting the table

If using Data Domain for the backup, the complete backupset must be on the Data Domain system. The complete backupset consists of the full backup along with the incremental backups. Note that Data Domain Boost is not supported with incremental backups.

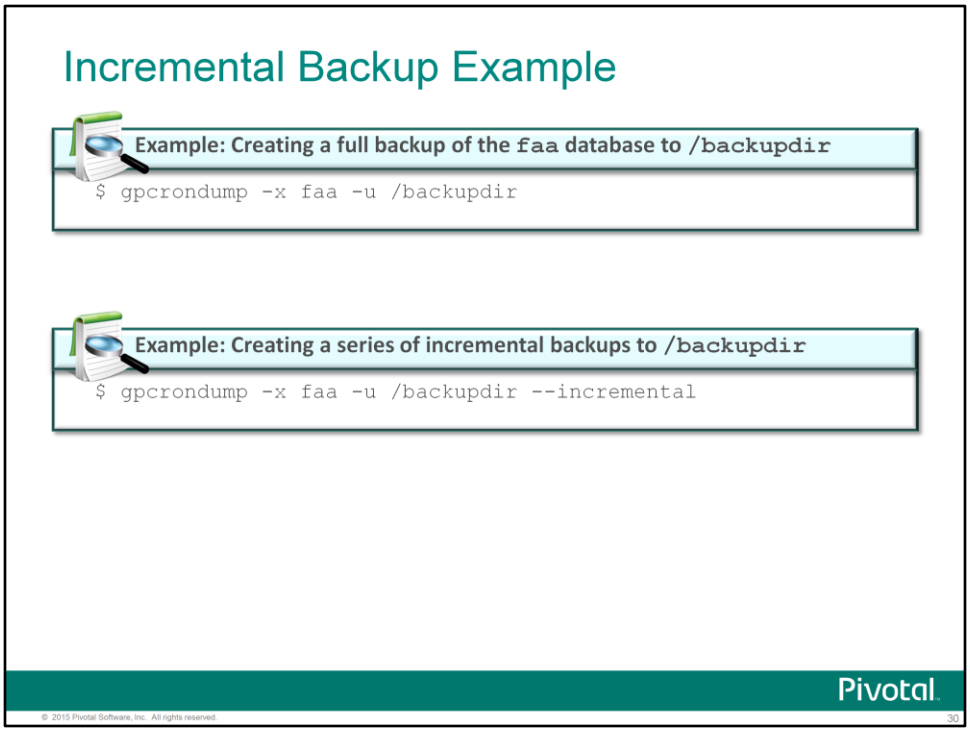


To run an incremental backup using the gpcrondump utility you must specify the option --incremental. As stated before incremental backups are not supported with DDBoost.

NOTE that if you are performing incremental backups, you have to ensure that your full backups are not executed with any of the DDBoost option found with the `gpcrondump` utility.

Restoring from an incremental backup requires the following

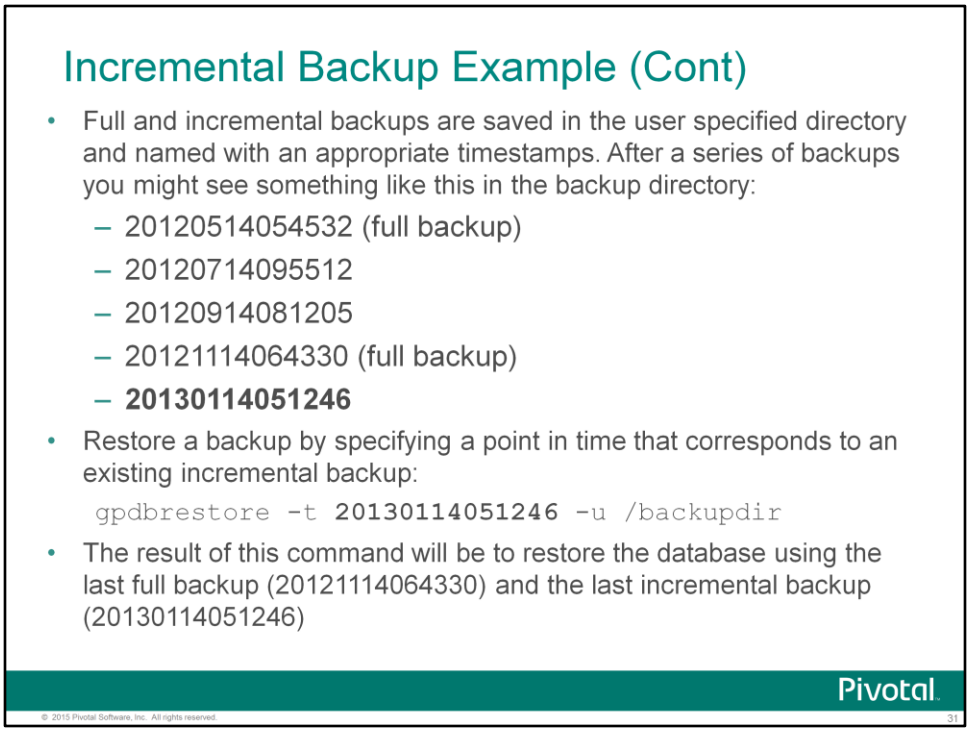
1. A Full backup has been perform (exists) before the current incremental that you want to restore from.
2. All incremental backups created between the time of the full backup the current incremental backup that you want to restore from exist.
3. All backups (full and incremental) are stored in the same directory location. A -u option can be used with gpcrondump to insure this.



The first command shown on the slide will perform a full backup of the `faa` database found in Greenplum. It will store all archive files in the directory `backupdir`.

The second command will perform an incremental backup on the faa database in Greenplum and place the archive files in the directory backupdir.

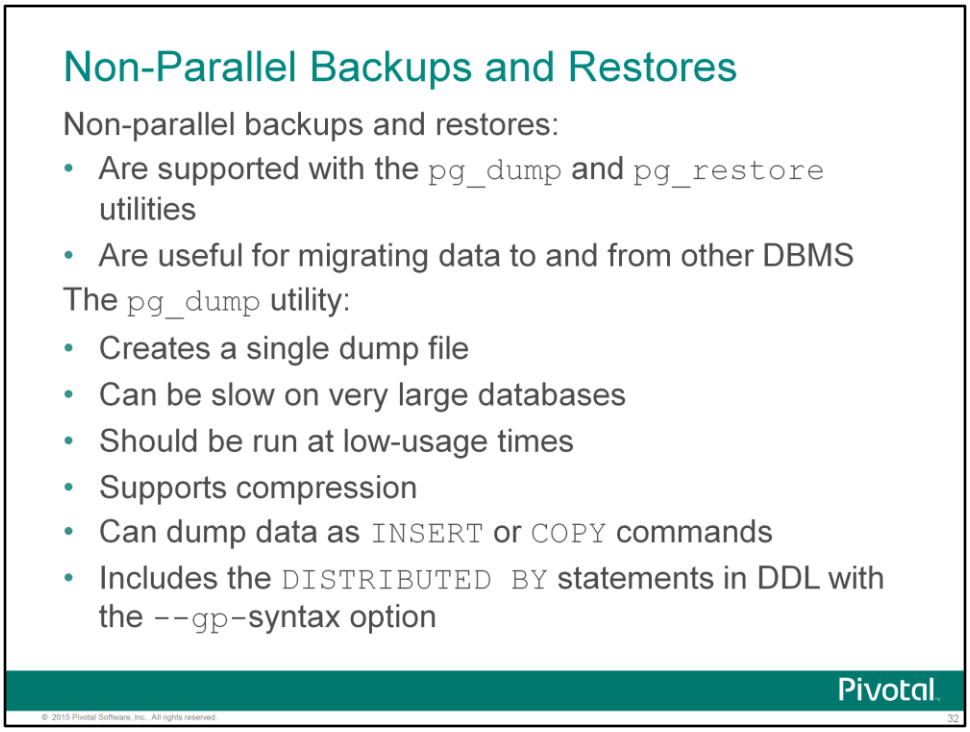
Note that the incremental backup only backs up changes performed on append-only tables found in the `faa` database. If no AO tables exist, or no AO tables have changed, a full backup will be performed.



We can see that a number of full and incremental backup have occurred. Archive files are named using the following timestamp format (YYYYMMDDHHMMSS) . So you can see that the archive files ending in 4532 and 4330 are full backups, and the other backups are incrementals.

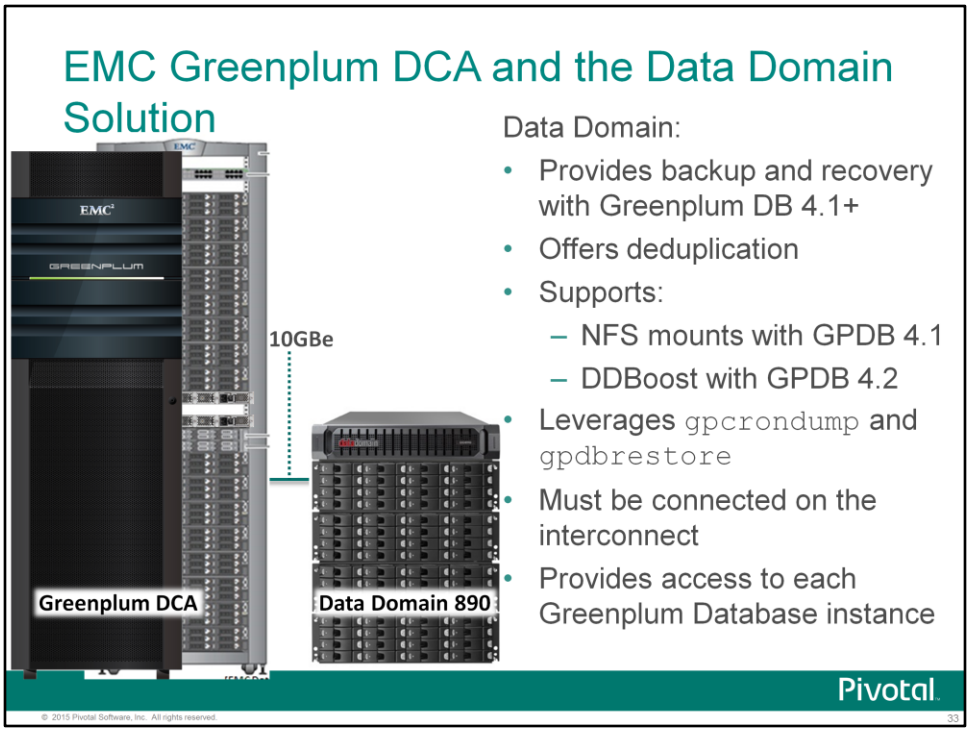
Running the gpdbrestore utility and specifying the archive file ending in 1246 will restore the database back to a time of the full backup ending in 4330 and the incremental backup ending in 1246.

NOTE: the other archive files in this example are not needed and will not be used with this restore command.



The `pg_dump` utility creates a single dump file to the master server, instead of to the individual segments.

To use this command, you must have adequate file system space on the master to store the resulting backup file. This should be used when the system is quiet, as it has the potential to impact the overall performance, especially on very large databases.



Starting with Greenplum Database 4.1 and 4.2, NFS mounted and native integration with Data Domain 880 and 890 was introduced to support backup and recovery services for the Greenplum Database. Data Domain offers a comprehensive backup management, offloading many of the duties a database administrator has with backup and recovery services to the device.

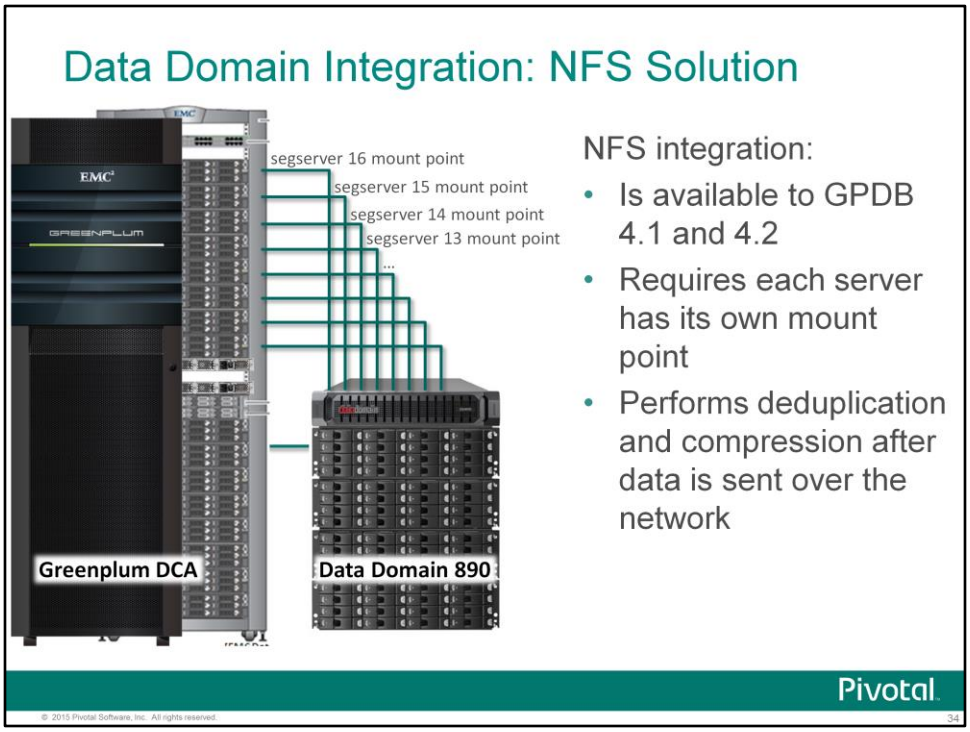
Deduplication offers a form of incremental backup not offered with the native Greenplum Database backup and recovery utilities.

There are two integration options offered for Greenplum Database with Data Domain:

- Backup and recovery with NFS mounted directories.
- Backup and recovery with DDBoost, a client library integrated with the database.

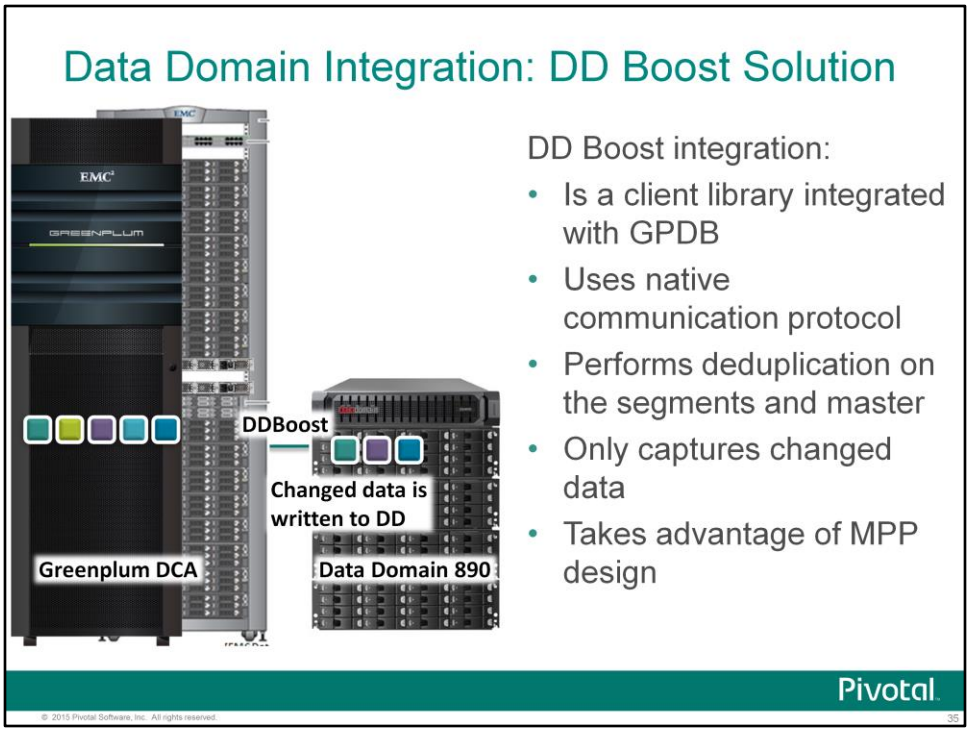
EMC Data Domain:

- Leverages the Greenplum backup utility, gpcrondump, to perform the backup and gpdbrestore for restores.
- Must be connected to the DCA on the interconnect switches. This enables 10 GbE connectivity between every DCA server and the Data Domain Appliance.
- Provides access to the Greenplum DCA using internal 10GbE network.



The NFS integration solution is available for both Greenplum Database 4.1 and 4.2. Once connected to the Data Domain, each segment server and the master server has its own mount point to the Data Domain appliance.

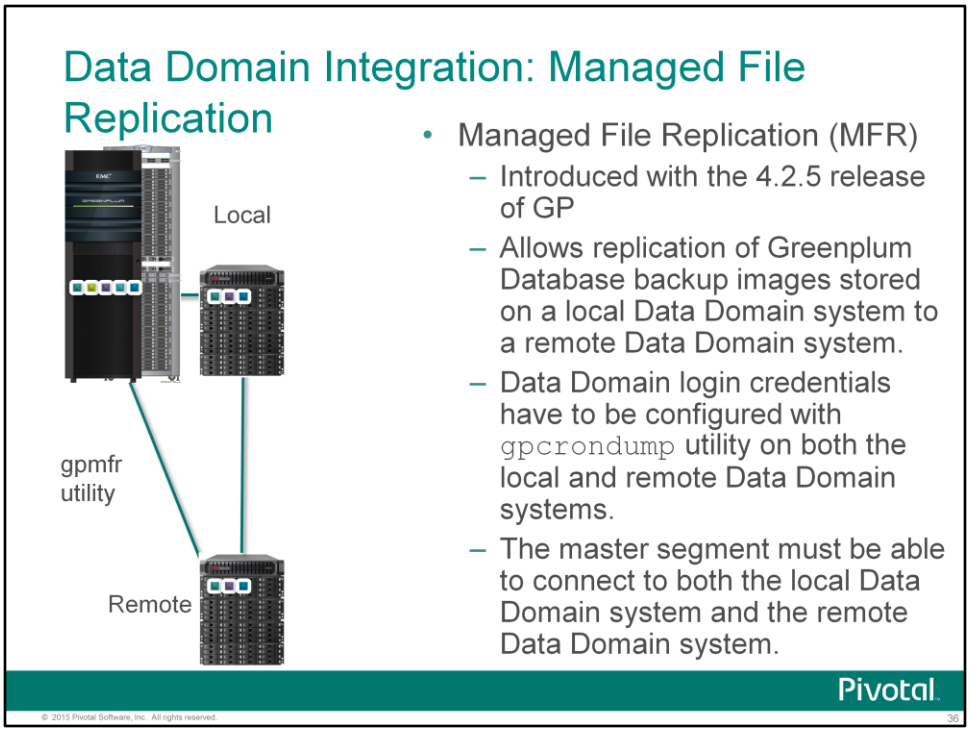
When performing a backup, data is sent over the network to the Data Domain appliance over the mount point. All deduplication and compression is applied after the information has been sent to the Data Domain appliance. While this is transparent to you, it can incur high costs for network bandwidth as all data is sent. At this point, there is no sense of incremental backup or deduplication until the data is fully sent to the Data Domain appliance.



DD Boost is a client library integrated into the Greenplum Database, fully integrated with the Greenplum Database utilities for backup and recovery, and runs on all segments.

It is a native communication protocol which then performs deduplication on the segments and master themselves, reducing network traffic. The only data that is transferred is data that has changed. This immediately takes advantage of deduplication and provides a form of incremental backup. In addition to taking advantage of deduplication, the data is compressed before it is sent over the network.

DD Boost takes advantage of the Greenplum Database cluster by pushing deduplication processing to the segment hosts and utilizing the CPU cores available to perform deduplication and compression. The more CPU cores are available, the better the performance. This enables faster backup as the Greenplum Database scales.



Managed File Replication is a feature found in Data Domain software. Collection Replication is the other type of replication supported.

Managed file replication directly transfers a backup image from one Data Domain system to another, one at a time on request from the backup software.

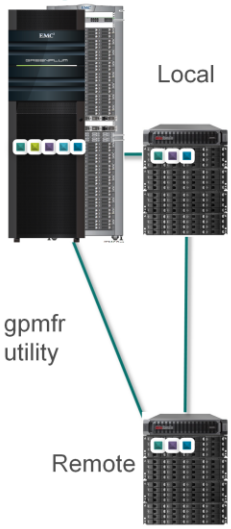
Collection replication performs whole system mirroring in a one-to-one topology continuously transferring changes in the underlying collection, including logical directories and files of the Data Domain filesystem.

Prior to Greenplum 4.2.5 only collection replication was supported. In the 4.2.5 and above releases of Greenplum you know can perform a Managed File Replication (MFR).

MFR allows you to take backup data sets (a set of archive files) and copy them over to a remote Data Domain system (a disaster recover site). The remote site data sets can then be used to perform restore operations of the Greenplum Database.

The Master server must be able to communicate with both Data Domain systems (local and remote) and do require additional setup steps. These additional steps require running the gpcrondump utility with certain `--ddboost` options to establish user credentials on the Data Domain systems.

Data Domain Integration: Managed File Replication

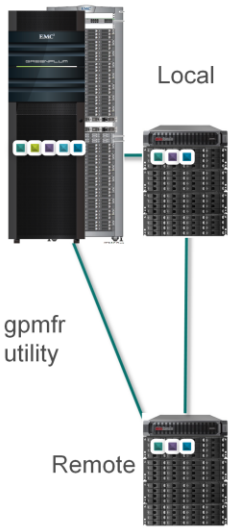


- The `gpmfr` utility manages the Greenplum Database backup sets on the local and remote Data Domain systems.
- The `gpmfr` utility provides these capabilities:
  - Lists the backup data sets that are on the local or the remote Data Domain system.
  - Replicates a backup data set that is on the local Data Domain system to the remote system.
  - Recovers a backup data set that is on the remote Data Domain system to the local system.
  - Deletes a backup data set that is on the local or the remote Data Domain system.

Pivotal

A new utility called `gpmfr` has been created that lets you manage your backup data sets on the local and remote Data Domain systems. gpmfr allows you to list, replicate, recover, and delete your backup sets on both Data Domain systems.

Data Domain Integration: MFR Example



The following example replicates the latest backup set on the local Data Domain sever to the remote server. The maximum number of I/O streams that can be used for the replication is 30.

```
gpmfr --replicate
LATEST --max-streams
30
```

Pivotal

38

The following example replicates the latest backup set on the local Data Domain sever to the remote server. The maximum number of I/O streams that can be used for the replication is 30.

Comparing the EMC Greenplum DCA Integration Solutions

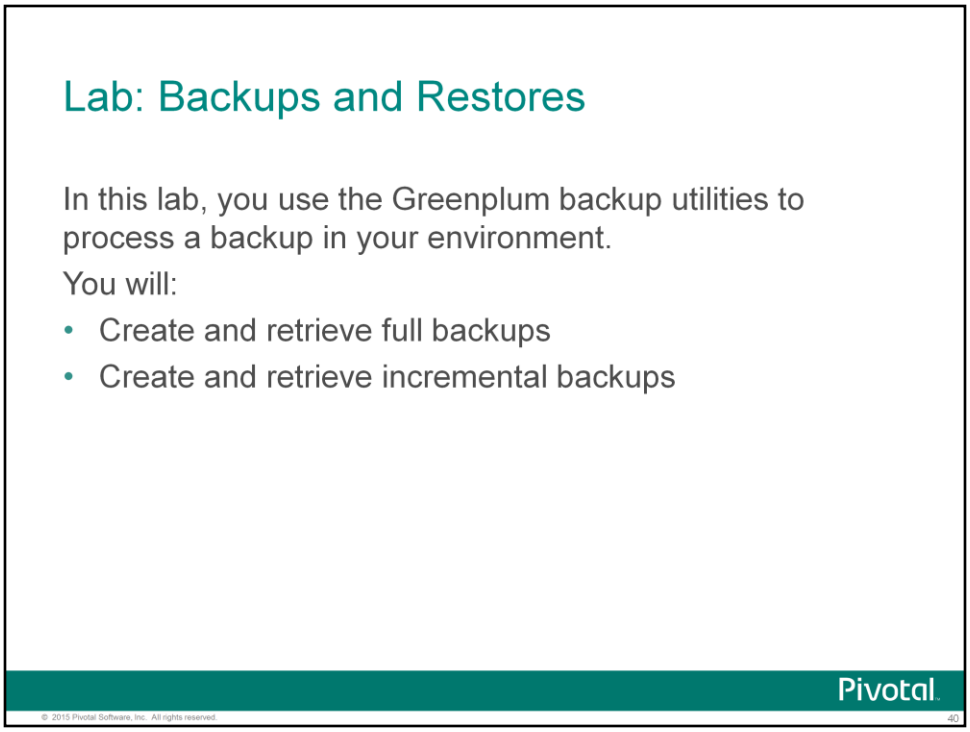
Feature	NFS	DD Boost
Deduplication	Deduplication on Data Domain appliance	Deduplication on Greenplum DB segment and master instance
CPU Usage on Segments	As needed for NFS	Increased CPU usage on GPDB due to de-duplication and compression
Network Utilization	All data is sent over the network	Only changed, deduplication data is sent over the network
Scalability	Increasing the number of racks can result in saturation of DD appliance or network	Minimal data transfer
Management	Each segment server and master server requires its own mount point	Integrated native solution with no static system configuration
Backup Performance	Full backup	Initial backup is full; follow-on backups are incremental
Data Domain Replication	Directory level	Collection and Managed File

Pivotal

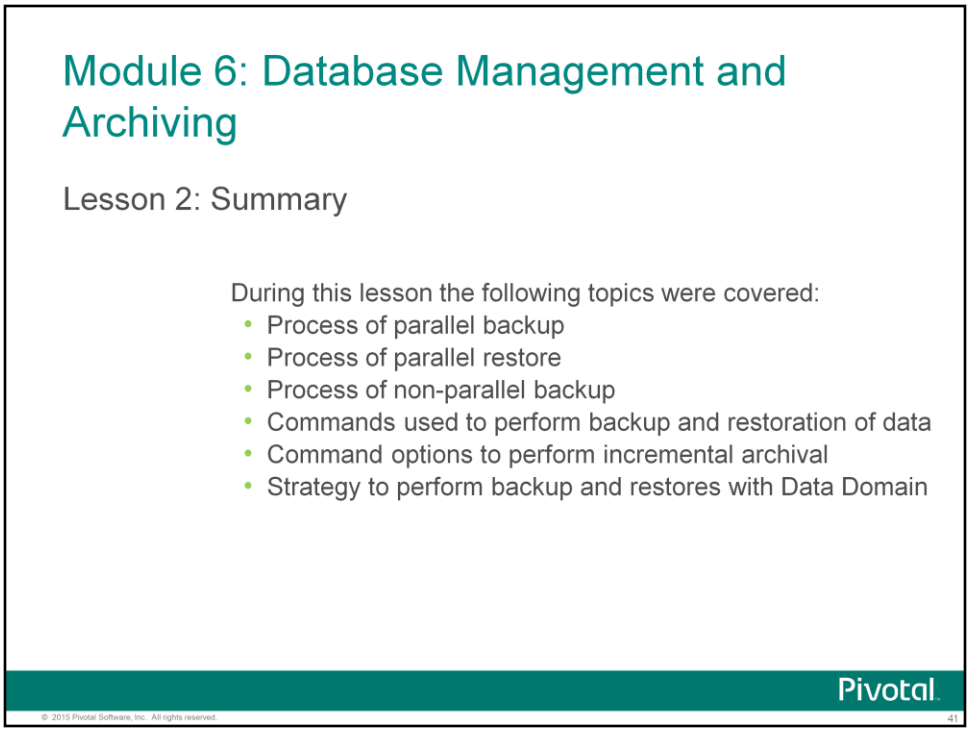
To use the Data Domain for backup and restore:

- All servers in the DCA must have NFS access to the Data Domain. Data Domain provides a dedicated NFS export to each of the servers in the DCA. For example, Data Domain will export the directory, `/backup/DCA-01/mdw`, which is available to the master server as `/backup`. Each segment will have its own directory available as well.
- The superuser account, `gpadmin`, must have full read and write access to the backup directory.

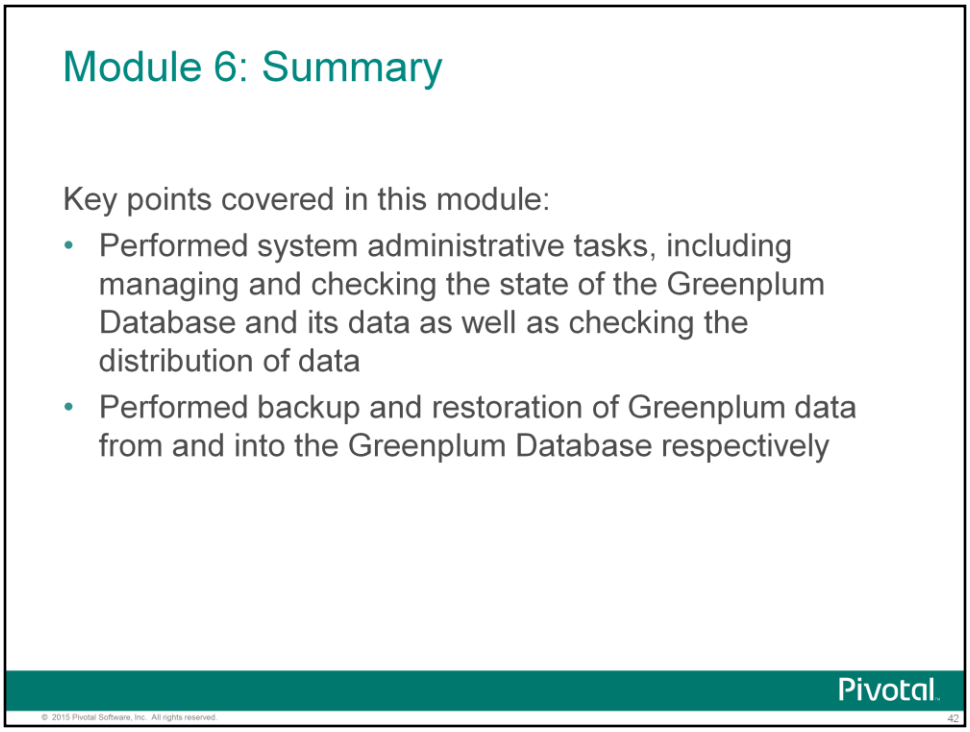
Once the NFS configuration has been completed, the Greenplum backup and restore utilities can be used to write to or read from the mount points.



In this lab, you use the Greenplum backup utilities to backup databases in your environment. You will perform both incremental and full backups of your data.



This lesson covered various archival topics, including the difference between parallel and non-parallel backup and restore and how to perform each of these using the Greenplum backup and restore utilities. Greenplum backup utilities include options both for incremental support for append-only storage as well as storing and restoring data to a Data Domain.



Listed are the key points covered in this module. You should have learned to:

- Perform system administrative tasks, including managing and checking the state of the Greenplum Database and its data as well as checking the distribution of data.
- Perform backup and restoration of Greenplum data from and into the Greenplum Database respectively.