Greenplum
The Data Computing Division of EMC

EMC²
where information lives®

Greenplum® Database 4.0 Installation Guide

P/N: 300-011-541
Rev: A02

Greenplum Database Installation Guide 4.0 - Contents

Preface

This guide describes the tasks you must do to install and start the Greenplum Database system.

- About This Guide
- Document Conventions
- Getting Support

About This Guide

This guide provides information and instructions for installing and initializing a Greenplum Database system. This guide is intended for system administrators responsible for building out a Greenplum Database system.

This guide assumes knowledge of Linux/Unix system administration, database management systems, database administration, and structured query language (SQL).

This guide contains the following chapters and appendices:

- Chapter 1, “Introduction to Greenplum” — Information about the Greenplum system architecture and components.
- Chapter 2, “Estimating Storage Capacity” — Guidelines for sizing a Greenplum Database system.
- Chapter 3, “Configuring your Systems for Greenplum” — Instructions about installing and configuring the Greenplum software on all hosts in your Greenplum Database array.
- Chapter 4, “Validating Your Systems” — Describes validation utilities and tests you can perform to ensure your Greenplum Database system will operate properly.
- Chapter 5, “Configuring Localization Settings” — Discusses the localization features of Greenplum Database. Locale settings must be configured prior to initializing your Greenplum Database system.
- Chapter 6, “Initializing a Greenplum Database System” — Instructions for initializing a Greenplum Database system. Each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS.
- Chapter 7, “Enabling the Performance Monitor” — Describes the (optional) performance monitoring feature that administrators can enable within Greenplum Database 4.0.
- Appendix A, “Installation Management Utilities” — Provides reference information about the command-line management utilities you use to install and initialize a Greenplum Database system,

Document Conventions

The following conventions are used throughout the Greenplum Database documentation to help you identify certain types of information.

- Text Conventions
- Command Syntax Conventions

Text Conventions

Table 0.1 Text Conventions

Text Convention	Usage	Examples
bold	Button, menu, tab, page, and field names in GUI applications	Click **Cancel** to exit the page without saving your changes.
italics	New terms where they are defined Database objects, such as schema, table, or columns names	The *master instance* is the `postgres` process that accepts client connections. Catalog information for Greenplum Database resides in the *pg_catalog* schema.
`monospace`	File names and path names Programs and executables Command names and syntax Parameter names	Edit the `postgresql.conf` file. Use `gpstart` to start Greenplum Database.

`monospace italics`	Variable information within file paths and file names
	Variable information within command syntax
		`/home/gpadmin/`*`config_file`*
		`COPY` *`tablename`* `FROM '`*`filename`*`'`

`monospace bold`	Used to call attention to a particular part of a command, parameter, or code snippet.	Change the host name, port, and database name in the JDBC connection URL: `jdbc:postgresql://`**`host:5432`**`/`**`mydb`**
`UPPERCASE`	Environment variables SQL commands Keyboard keys	Make sure that the Java `/bin` directory is in your `$PATH`. `SELECT * FROM` *`my_table`*`;` Press `CTRL+C` to escape.

Command Syntax Conventions

Table 0.2 Command Syntax Conventions

Text Convention	Usage	Examples
`{ }`	Within command syntax, curly braces group related command options. Do not type the curly braces.	`FROM { 'filename' | STDIN }`
`[ ]`	Within command syntax, square brackets denote optional arguments. Do not type the brackets.	`TRUNCATE [ TABLE ] name`
`...`	Within command syntax, an ellipsis denotes repetition of a command, variable, or option. Do not type the ellipsis.	`DROP TABLE name [, ...]`
`|`	Within command syntax, the pipe symbol denotes an "OR" relationship. Do not type the pipe symbol.	`VACUUM [ FULL | FREEZE ]`

`$ system_command`
`# root_system_command`
`=> gpdb_command`
`=# su_gpdb_command`
	Denotes a command prompt - do not type the prompt symbol. `$` and `#` denote terminal command prompts. `=>` and `=#` denote Greenplum Database interactive program command prompts (`psql` or `gpssh`, for example).	`$ createdb mydatabase`
		`# chown gpadmin -R /datadir`
		`=> SELECT * FROM mytable;`
		`=# SELECT * FROM pg_database;`

Getting Support

EMC support, product, and licensing information can be obtained as follows.

Product information

For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Powerlink website (registration required) at:

http://Powerlink.EMC.com

Technical support

For technical support, go to Powerlink and choose **Support**. On the Support page, you will see several options, including one for making a service request. Note that to open a service request, you must have a valid support agreement. Please contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

1. Introduction to Greenplum

Greenplum Database is able to handle the storage and processing of large amounts of data by distributing the load across several servers or *hosts*. A logical database in Greenplum is actually an *array* of individual PostgreSQL databases, all working together to present a single database image. The *master* is the entry point to the Greenplum Database system. It is the database instance where clients connect and submit SQL statements. The master coordinates the work across the other database instances in the system, the *segments*, which handle data processing and storage. The segments communicate with each other and with the master over the *interconnect*, which is the networking layer of Greenplum Database.

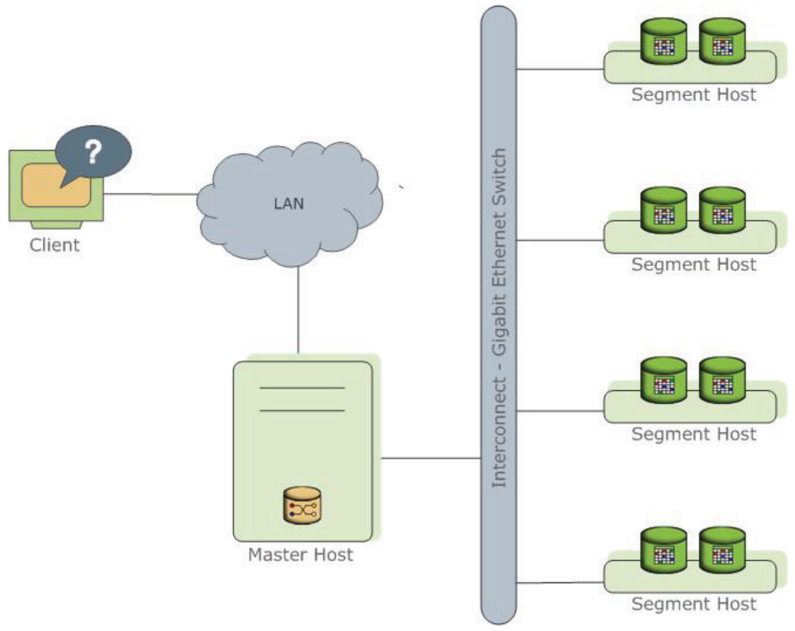

Figure 1.1

Greenplum Database is a software-only solution, meaning that it runs on a variety of commodity server platforms from Greenplum-certified hardware vendors. The hardware and database software are not coupled as with some other data warehouse appliance vendors. However, as with any database, Greenplum's performance is dependent on the hardware on which it is installed. And because the database is distributed across multiple machines in a Greenplum Database system, the selection and configuration of hardware is even more important to achieving the best performance possible.

This chapter describes the major components of a Greenplum Database system and the hardware considerations and concepts associated with each component; The Greenplum Master, The Segments and The Interconnect. Additionally, a system may also have optional ETL Hosts for Data Loading and the Greenplum Performance Monitor for monitoring query workload and performance.

The Greenplum Master

The *master* is the entry point to the Greenplum Database system. It is the database process that accepts client connections and processes the SQL commands issued by the users of the system. Users connect to Greenplum Database through the master using PostgreSQL-compatible client programs such as `psql` or ODBC.

The master maintains the *system catalog* (a set of system tables that contain metadata about the Greenplum Database system itself), however the master does not contain any user data. Data resides only on the *segments*. The master does the work of authenticating client connections, processing the incoming SQL commands, distributing the work load between the segments, coordinating the results returned by each of the segments, and presenting the final results to the client program.

Since the master does not contain any user data, it has very little disk load. However, in production environments extra space is often needed for landing load files and backup files. Customers may decide to also run ETL and reporting tools on the master, thereby requiring more disk space and processing power. The master does need a fast, dedicated CPU for data loading, connection handling and query planning.

Master Redundancy

You can also optionally deploy a *backup* or *mirror* of the master instance. A backup master host serves as a *warm standby* in the event of the primary master host becoming unoperational. You can deploy the standby master on a designated redundant master host or on one of the segment hosts.

The standby master is kept up to date by a transaction log replication process, which runs on the standby master host and keeps the data between the primary and standby master hosts synchronized. If the primary master fails, the log replication process is shutdown, and the standby master can be activated in its place. Upon activation of the standby master, the replicated logs are used to reconstruct the state of the master host at the time of the last successfully committed transaction.

Since the master does not contain any user data, only the system catalog tables need to be synchronized between the primary and backup copies. These tables are not updated frequently, but when they are, changes are automatically copied over to the standby master so that it is always kept current with the primary.

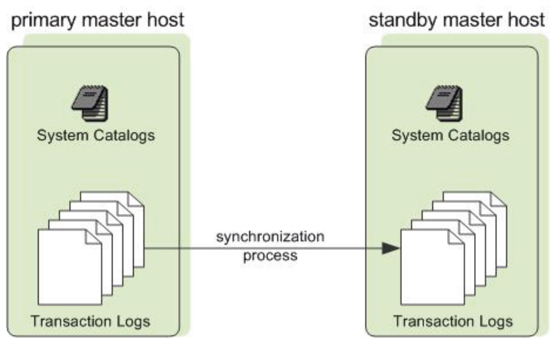

Figure 1.2 Master Mirroring in Greenplum Database

The Segments

In Greenplum Database, the *segments* are where the data is stored and where the majority of query processing takes place. User-defined tables and their indexes are distributed across the available number of segments in the Greenplum Database system, each segment containing a distinct portion of the data. Segment instances are the database server processes that serve segments. Users do not interact directly with the segments in a Greenplum Database system, but do so through the master.

In the recommended Greenplum Database hardware configuration, there is one active segment per effective CPU or CPU core. For example, if your segment hosts have two dual-core processors, you would have four primary segments per host.

Segment Redundancy

When you deploy your Greenplum Database system, you have the option to configure *mirror* segments. Mirror segments allow database queries to fail over to a backup segment if the primary segment becomes unavailable. To configure mirroring, you must have enough hosts in your Greenplum Database system so that the secondary segment always resides on a different host than its primary. Figure 1.3 shows how table data is distributed across the segments when mirroring is configured. The mirror segment always resides on a different host than its primary segment.

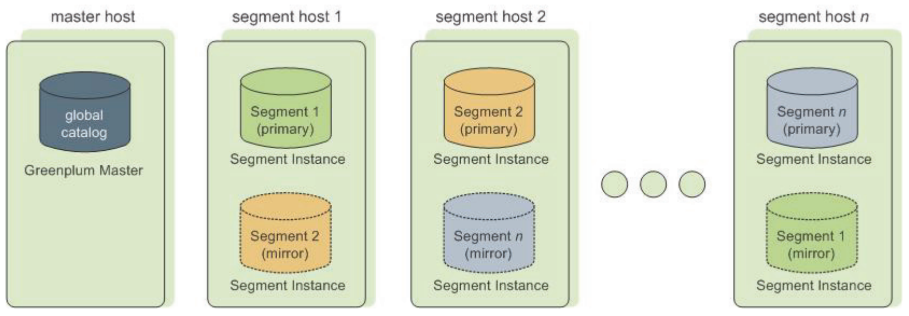

Figure 1.3 Data Mirroring in Greenplum Database

Segment Failover and Recovery

When mirroring is enabled in a Greenplum Database system, the system will automatically fail over to the mirror copy whenever a primary copy becomes unavailable. A Greenplum Database system can remain operational if a segment instance or host goes down as long as all portions of data are available on the remaining active segments.

Whenever the master cannot connect to a segment instance, it marks that segment instance as *invalid* in the Greenplum Database system catalog. The segment instance will remain invalid and out of operation until steps are taken to bring that segment back online. Once a segment is back online, the master will mark it as valid again the next time it successfully connects to the segment.

Recovery of a failed segment depends on the configured *fault operational mode*. If the system is running in *read-only* mode (the default), users will not be able to issue DDL or DML commands when there are failed segments in the system. In read-only mode, you can recover a segment with only a short interruption of service. If the system is running in *continue* mode, all operations will continue as long as there is one active segment instance alive per portion of data. In this mode, the data on the failed segment must be reconciled with the active segment before it can be brought back into operation. The system must be shutdown to recover failed segments.

If you do not have mirroring enabled, the system will automatically shutdown if a segment instance becomes invalid. You must recover all failed segments before operations can continue.

Example Segment Host Hardware Stack

Regardless of the hardware platform you choose, a production Greenplum Database processing node (a segment host) is typically configured as described in this section.

The segment hosts do the majority of database processing, so the segment host servers are configured in order to achieve the best performance possible from your Greenplum Database system. Greenplum Database's performance will be as fast as the slowest segment server in the array. Therefore, it is important to ensure that the underlying hardware and operating systems that are running Greenplum Database are all running at their optimal performance level. It is also advised that all segment hosts in a Greenplum Database array have identical hardware resources and configurations.

Segment hosts should also be dedicated to Greenplum Database operations only. To get the best query performance, you do not want Greenplum Database competing with other applications for machine or network resources.

The following diagram shows an example Greenplum Database segment host hardware stack. The number of effective CPUs on a host is the basis for determining how many primary Greenplum Database segment instances to deploy per segment host. This example shows a host with two effective CPUs (one dual-core CPU). Note that there is one primary segment instance (or primary/mirror pair if using mirroring) per CPU core.

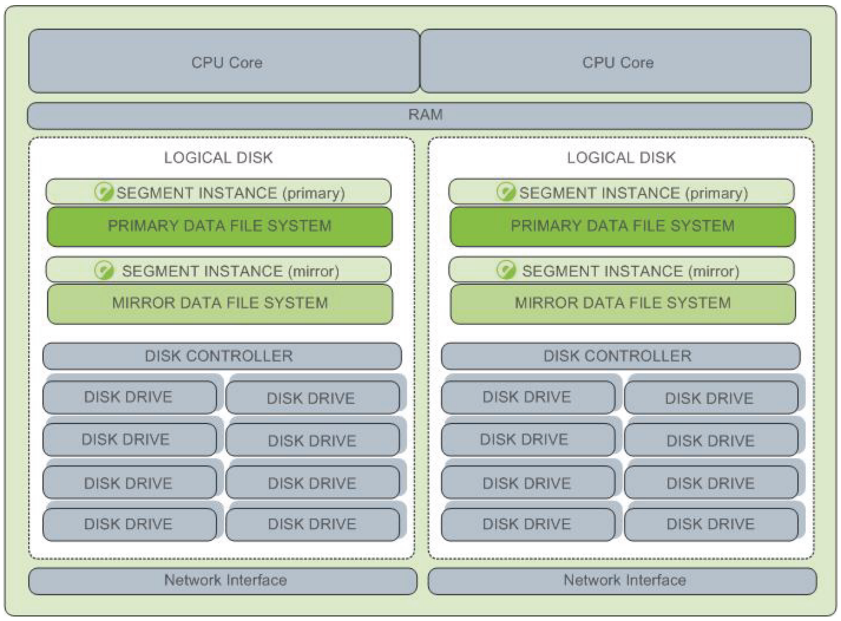

Figure 1.4 Example Greenplum Database Segment Host Configuration

Example Segment Disk Layout

Each CPU is typically mapped to a logical disk. A logical disk consists of one primary file system (and optionally a mirror file system) accessing a pool of physical disks through an I/O channel or disk controller. The logical disk and file system are provided by the operating system. Most operating systems provide the ability for a logical disk drive to use groups of physical disks arranged in RAID arrays.

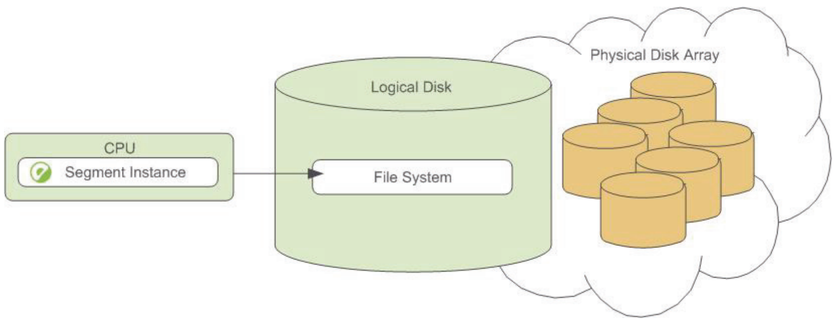

Figure 1.5 Logical Disk Layout in Greenplum Database

Depending on the hardware platform you choose, different RAID configurations offer different performance and capacity levels. Refer to the

The Interconnect

The *interconnect* is the networking layer of Greenplum Database. When a user connects to a database and issues a query, processes are created on each of the segments to handle the work of that query. The *interconnect* refers to the inter-process communication between the segments, as well as the network infrastructure on which this communication relies. The interconnect uses a standard Gigabit Ethernet switching fabric.

By default, the interconnect uses UDP (User Datagram Protocol) to send messages over the network. The Greenplum software does the additional packet verification and checking not performed by UDP, so the reliability is equivalent to TCP (Transmission Control Protocol), and the performance and scalability exceeds that of TCP.

Interconnect Redundancy

A highly available interconnect can be achieved by deploying dual Gigabit Ethernet switches on your network, and redundant Gigabit connections to the Greenplum Database master and segment host servers.

Network Interface Configuration

A segment host typically has one network interface per primary segment instance. If using mirroring, a primary/mirror pair would then share an interface. The master host would also have four network interfaces to the Greenplum Database array plus additional external network interfaces.

On each Greenplum Database segment host, you would then create separate host names for each network interface. For example, if a host has four network interfaces, then it would have four corresponding host names, each of which will map to a primary segment instance. You would also do the same for the master host, however, when you initialize your Greenplum Database array, only one master host name will be used within the array.

With this configuration, the operating system automatically selects the best path to the destination. Greenplum Database automatically balances the network destinations to maximize parallelism.

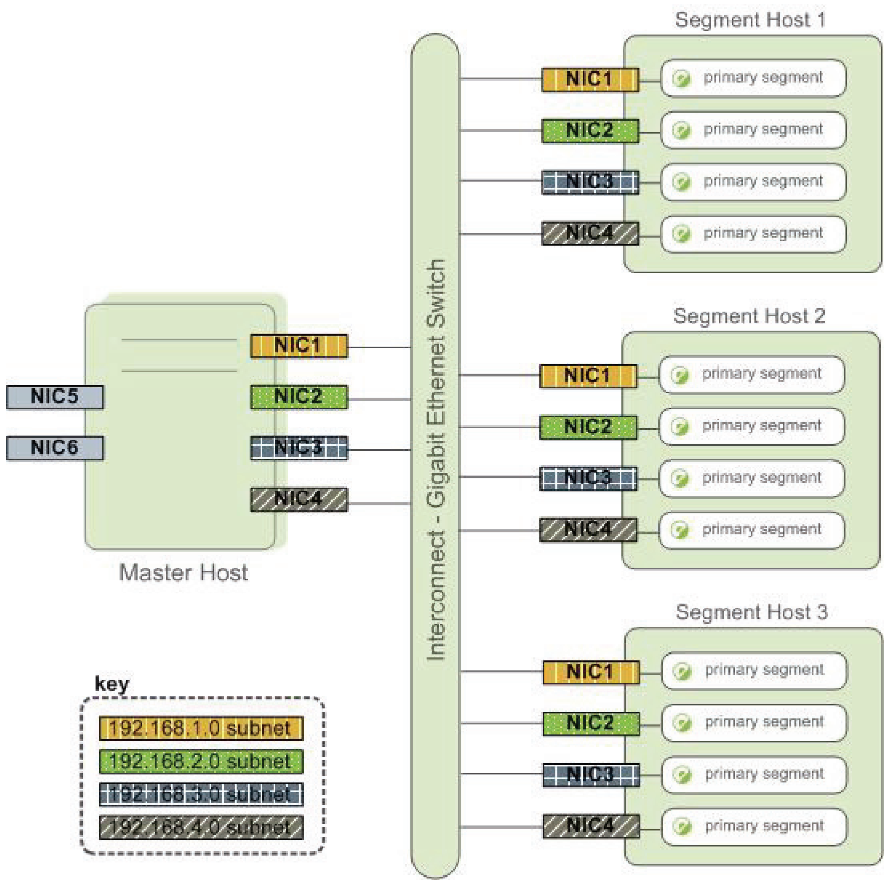

Figure 1.6 Example Network Interface Architecture

Switch Configuration

When using multiple Gigabit Ethernet switches within your Greenplum Database array, evenly divide the number of subnets between each switch. In this example configuration, if we had two switches, NICs 1 and 2 on each host would use switch 1 and NICs 3 and 4 on each host would use switch 2. For the master host, the host name bound to NIC 1 (and therefore using switch 1) is the effective master host name for the array. Therefore, if deploying a warm standby master for redundancy purposes, the standby master should map to a NIC that uses a different switch than the primary master.

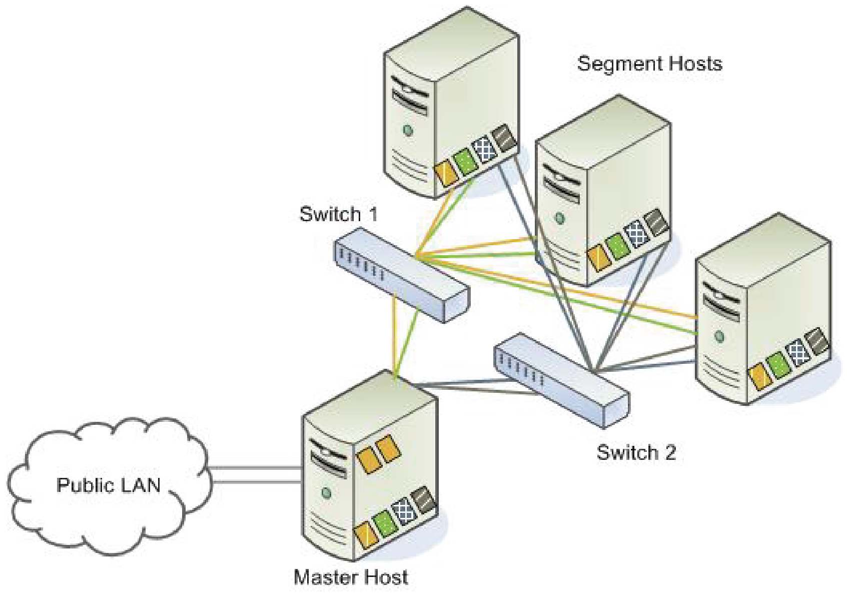

Figure 1.7 Example Switch Configuration

ETL Hosts for Data Loading

Greenplum supports fast, parallel data loading with its external tables feature. By using external tables in conjunction with Greenplum Database's parallel file server (`gpfdist`), administrators can achieve maximum parallelism and load bandwidth from their Greenplum Database system. Many production systems deploy designated ETL servers for data loading purposes. These machines run the Greenplum parallel file server (`gpfdist`), but not Greenplum Database instances.

One advantage of using the `gpfdist` file server program is that it ensures that all of the segments in your Greenplum Database system are fully utilized when reading from external table data files.

The `gpfdist` program can serve data to the segment instances at an average rate of about 350 MB/s for delimited text formatted files and 200 MB/s for CSV formatted files. Therefore, you should consider the following options when running `gpfdist` in order to maximize the network bandwidth of your ETL systems:

- If your ETL machine is configured with multiple network interface cards (NICs) as described in "Network Interface Configuration" on page 10, run one instance of `gpfdist` on your ETL host and then define your external table definition so that the host name of each NIC is declared in the `LOCATION` clause (see `CREATE EXTERNAL TABLE` in the *Greenplum Database Administrator Guide*). This allows network traffic between your Greenplum segment hosts and your ETL host to use all NICs simultaneously.

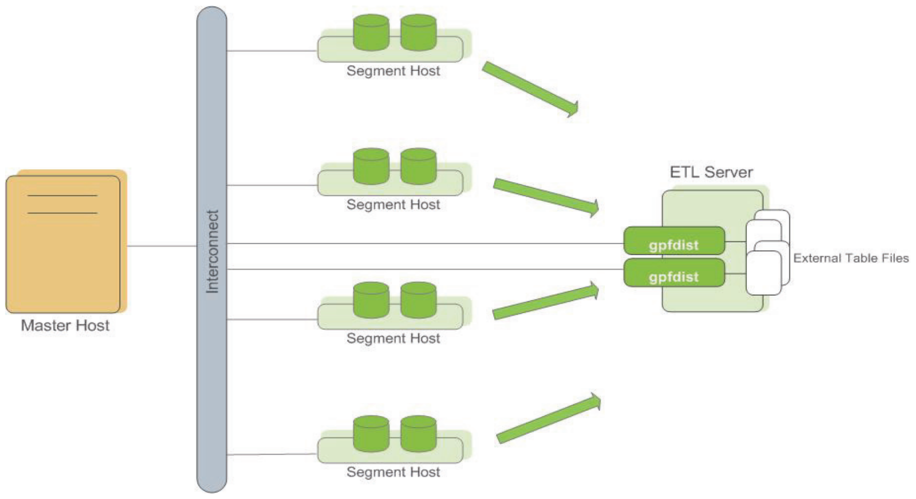

Figure 1.8 External Table Using Single gpfdist Instance with Multiple NICs

- Run multiple `gpfdist` instances on your ETL host and divide your external data files equally between each instance. For example, if you have an ETL system with two network interface cards (NICs), then you could run two `gpfdist` instances on that machine to maximize your load performance. You would then divide the external table data files evenly between the two `gpfdist` programs.

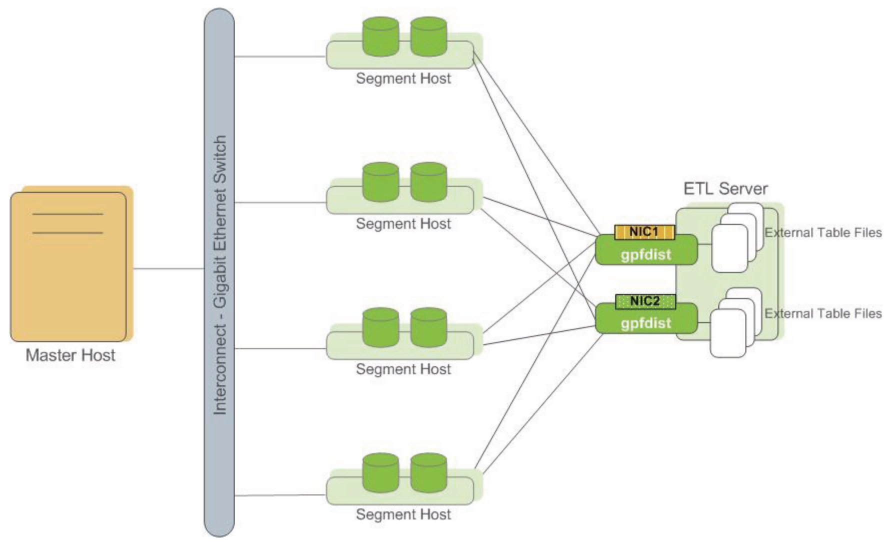

Figure 1.9 External Tables Using Multiple gpfdist Instances with Multiple NICs

Greenplum Performance Monitor

Greenplum also provides an optional performance monitoring feature that administrators can install and enable with Greenplum Database. To use the Greenplum Performance Monitor, each host in your Greenplum Database array must have a monitor agent installed. When you start the Greenplum Performance Monitor, the agents begin collecting data on queries and system utilization. Segment agents send their data to the Greenplum master at regular intervals (typically every 15 seconds). Users can query the Greenplum Performance Monitor database to see query and system performance data for both active queries and historical queries. Greenplum Performance Monitor also has a graphical web-based user interface for viewing these performance metrics.

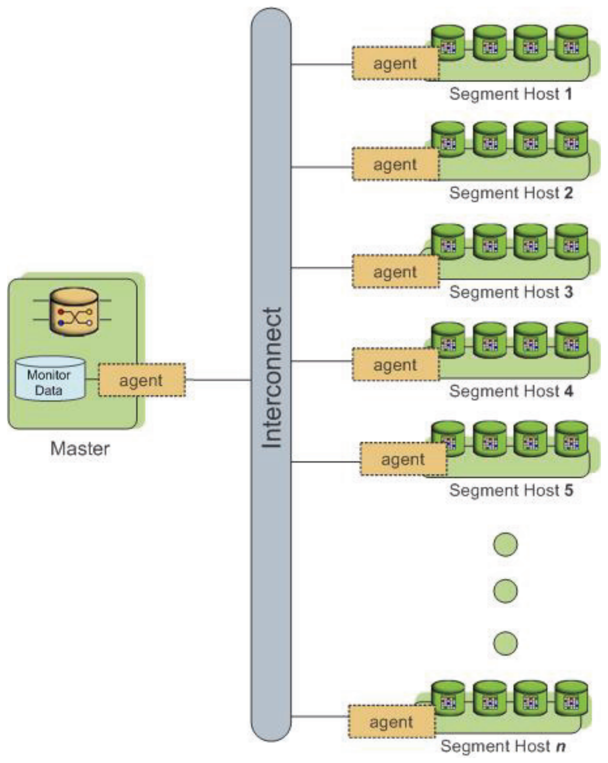

Figure 1.10 Greenplum Performance Monitor Architecture

2. Estimating Storage Capacity

To estimate how much data your Greenplum Database system can accommodate, use the following measurements as guidelines. Also keep in mind that you may want to have extra space for landing backup files and data load files on each segment host.

Calculating Usable Disk Capacity

To calculate how much data a Greenplum system can hold, you have to calculate the usable disk capacity per segment host and then multiply that by the number of segment hosts in your Greenplum array. Start with the raw capacity of the physical disks on a segment host that are available for data storage (*`raw_capacity`*), which is:

```
disk_size * number_of_disks
```

Account for file system formatting overhead (roughly 10 percent) and the RAID level you are using. For example, if using RAID-10, the calculation would be:

```
(raw_capacity * 0.9) / 2 = formatted_disk_space
```

For optimal performance, Greenplum recommends that you do not completely fill your disks to capacity, but run at 70% or lower. So with this in mind, calculate the usable disk space as follows:

```
formatted_disk_space * 0.7 = usable_disk_space
```

Once you have formatted RAID disk arrays and accounted for the maximum recommended capacity (*`usable_disk_space`*), you will need to calculate how much storage is actually available for user data (`U`). If using Greenplum mirrors for data redundancy, this would then double the size of your user data (`2 * U`). Greenplum also requires some space be reserved as a working area for active queries. The work space should be approximately one third the size of your user data (work space = `U/3`):

With mirrors: `(2 * U) + U/3 = usable_disk_space`

Without mirrors: `U + U/3 = usable_disk_space`

Calculating User Data Size

As with all databases, the size of your raw data will be slightly larger once it is loaded into the database. On average, raw data will be about 1.4 times larger on disk after it is loaded into the database, but could be smaller or larger depending on the data types you are using, table storage type, in-database compression, and so on.

- **Page Overhead -** When your data is loaded into Greenplum Database, it is divided into pages of 32KB each. Each page has 20 bytes of page overhead.

- **Row Overhead** - In a regular 'heap' storage table, each row of data has 24 bytes of row overhead. An 'append-only' storage table has only 4 bytes of row overhead.
- **Attribute Overhead** - For the data values itself, the size associated with each attribute value is dependent upon the data type chosen. As a general rule, you want to use the smallest data type possible to store your data (assuming you know the possible values a column will have).
- **Indexes** - In Greenplum Database, indexes are distributed across the segment hosts as is table data. The default index type in Greenplum Database is B-tree. Because index size depends on the number of unique values in the index and the data to be inserted, precalculating the exact size of an index is impossible. However, you can roughly estimate the size of an index using these formulas.

  B-tree: `unique_values` * (`data_type_size` + 24 bytes)

  Bitmap: (`unique_values` * `number_of_rows` * 1 bit * `compression_ratio` / 8) + (`unique_values` * 32)

Calculating Space Requirements for Metadata and Logs

On each segment host, you will also want to account for space for Greenplum Database log files and metadata:

- **System Metadata** — For each Greenplum Database segment instance (primary or mirror) or master instance running on a host, estimate approximately 20 MB for the system catalogs and metadata.
- **Write Ahead Log** — For each Greenplum Database segment (primary or mirror) or master instance running on a host, allocate space for the write ahead log (WAL). The WAL is divided into segment files of 64 MB each. At most, the number of WAL files will be: `2 * checkpoint_segments + 1`. You can use this to estimate space requirements for WAL. The default *checkpoint_segments* setting for a Greenplum Database instance is 8, meaning 1088 MB WAL space allocated for each segment or master instance on a host.
- **Greenplum Database Log Files** — Each segment instance and the master instance generates database log files, which will grow over time. Sufficient space should be allocated for these log files, and some type of log rotation facility should be used to ensure that to log files do not grow too large.
- **Performance Monitor Data** — The Greenplum Performance Monitor agents run on the same set of hosts as your Greenplum Database instance and utilize the system resources of those hosts. The resource consumption of the Greenplum Performance Monitor agent processes on these hosts is minimal and should not significantly impact database performance. Historical data collected by Greenplum Performance Monitor data is stored in its own `gpperfmon` database within your Greenplum Database system. Collected monitor data is distributed just like regular database data, so you will need to account for disk space in the data directory locations of your Greenplum segment instances. The amount of space required depends on the amount of historical data you would like to keep.

3. Configuring your Systems for Greenplum

This chapter describes how to prepare your operating system environment for Greenplum, and install the Greenplum Database software binaries on all of the hosts that will comprise your Greenplum Database system. Perform the following tasks in order:

1. Make sure your systems meet the System Requirements
1. Configuring OS Parameters for Greenplum
2. (master only) Running the Greenplum Installer
3. Creating the Greenplum Administrative User
4. (master and standby master only) Configuring Greenplum Environment Variables
5. Installing the Greenplum Binaries on Multiple Hosts
6. Creating the Data Storage Areas
7. Setting up a Trusted Host Environment
8. Synchronizing System Clocks
9. Next Steps

Unless noted, these tasks should be performed for *all* hosts in your Greenplum Database array (master, standby master and segments).

System Requirements

The following table lists minimum recommended specifications for servers intended to support Greenplum Database in a production environment. Greenplum also provides hardware build guides for its certified hardware platforms. It is recommended that you work with a Greenplum Systems Engineer to review your anticipated environment to ensure an appropriate hardware configuration for Greenplum Database.

Table 3.1 System Prerequisites for Greenplum Database 4.0

Operating System	SUSE Linux SLES 10.2 or higher CentOS 5.0 or higher RedHat Enterprise Linux 5.0 or higher Solaris x86 v10 update 7
File Systems	• xfs required for data storage on SUSE Linux and Red Hat (ext3 supported for root file system) • zfs required for data storage on Solaris (ufs supported for root file system)
Minimum CPU	Pentium Pro compatible (P3/Athlon and above)

Table 3.1 System Prerequisites for Greenplum Database 4.0

Minimum Memory	16 GB RAM per server
Disk Requirements	• 150MB per host for Greenplum installation • Approximately 300MB per segment instance for meta data • Appropriate free space for data with disks at no more than 70% capacity • High-speed, local storage
Network Requirements	Gigabit Ethernet within the array Dedicated, non-blocking switch
Software and Utilities	bash shell GNU tar GNU zip GCC runtime libraries (glibc, etc.) GNU readline (Solaris only)[1]

1. On Solaris platforms, you must have GNU Readline in your environment to support interactive Greenplum administrative utilities such as `gpssh`. Certified readline packages are available for download from the Greenplum Network.

Configuring OS Parameters for Greenplum

Greenplum requires the certain operating system (OS) parameters be set on all hosts in your Greenplum Database system (masters and segments).

- Solaris 10 x86
- Linux
- Mac OS X

In general, the following categories of system parameters need to be altered:

- **Shared Memory** - A Greenplum Database instance will not work unless the shared memory segment for your kernel is properly sized. Most default OS installations have the shared memory values set too low for Greenplum Database. On Linux systems, you must also disable the OOM (out of memory) killer.
- **Network** - On high-volume Greenplum Database systems, certain network-related tuning parameters must be set to optimize network connections made by the Greenplum interconnect.
- **User Limits** - User limits control the resources available to processes started by a user's shell. Greenplum Database requires a higher limit on the allowed number of file descriptors that a single process can have open. The default settings may cause some Greenplum Database queries to fail because they will run out of file descriptors needed to process the query.

Solaris 10 x86

Set the following parameters in `/etc/system`:

```
set rlim_fd_cur=65536
set zfs:zfs_arc_max=0x600000000
set pcplusmp:apic_panic_on_nmi=1
set nopanicdebug=1
```

Change the following line in the `/etc/project` file from:

```
default:3::::
```

to:

```
default:3:default
project:::project.max-sem-ids=(priv,1024,deny);
process.max-filedescriptor=(priv,252144,deny)
```

Add the following line to `/etc/user_attr`:

```
gpadmin::::defaultpriv=basic,dtrace_user,dtrace_proc
```

Linux

Set the following parameters in the `/etc/sysctl.conf` file and reboot:

```
kernel.sem = 250 64000 100 512
kernel.shmmax = 500000000
kernel.shmmni = 4096
kernel.shmall = 4000000000
kernel.sem = 250 64000 100 512
kernel.sysrq = 1
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
net.ipv4.tcp_syncookies = 1
net.ipv4.ip_forward = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.tcp_tw_recycle=1
net.ipv4.tcp_max_syn_backlog=4096
net.ipv4.conf.all.arp_filter = 1
net.core.netdev_max_backlog=10000
vm.overcommit_memory=2
```

Set the following parameters in the `/etc/security/limits.conf` file:

```
* soft nofile 65536
* hard nofile 65536
* soft nproc 131072
* hard nproc 131072
```

XFS is the preferred file system for data storage. Greenplum recommends the following xfs mount options:

```
noatime,nodiratim,logbufs=8
```

Also, each disk device file should have a read-ahead value of 16384.

Mac OS X

Disable IPv6 network addressing.

Add the following to `/etc/sysctl.conf`:

```
kern.sysv.shmmax=2147483648
kern.sysv.shmmin=1
kern.sysv.shmmni=64
kern.sysv.shmseg=16
kern.sysv.shmall=524288
kern.maxfiles=65535
kern.maxfilesperproc=65535
net.inet.tcp.msl=60
```

If using DHCP, add the following line to `/etc/hostconfig`:

```
HOSTNAME="your_hostname"
```

Running the Greenplum Installer

To configure your systems for Greenplum Database, you will need certain utilities found in `$GPHOME/bin` of your installation. Log in as `root` and run the Greenplum installer on the machine that will be your master host.

To install the Greenplum binaries on the master host

1. Download or copy the installer file to the machine that will be the Greenplum Database master host. Installer files are available from Greenplum for RedHat (32-bit and 64-bit), Solaris 64-bit and SuSe Linux 64-bit platforms.
2. Unzip the installer file where *`PLATFORM`* is either `RHEL5-i386` (RedHat 32-bit), `RHEL5-x86_64` (RedHat 64-bit), `SOL-x86_64` (Solaris 64-bit) or `SuSE10-x86_64` (SuSe Linux 64 bit). For example:
   ```
   # unzip greenplum-db-4.0.x.x-PLATFORM.zip
   ```
3. Launch the installer using `bash`. For example:
   ```
   # /bin/bash greenplum-db-4.0.x.x-PLATFORM.bin
   ```
4. The installer will prompt you to accept the Greenplum Database license agreement. Type `yes` to accept the license agreement.
5. The installer will prompt you to provide an installation path. Press `ENTER` to accept the default install path (`/usr/local/greenplum-db-4.0.x.x`), or enter an absolute path to an install location. You must have write permissions to the location you specify.
6. The installer will install the Greenplum software and create a `greenplum-db` symbolic link one directory level above your version-specific Greenplum installation directory. The symbolic link is used to facilitate patch maintenance and upgrades between versions. The installed location is referred to as `$GPHOME`.

About Your Greenplum Database Installation

- `greenplum_path.sh` — This file contains the environment variables for Greenplum Database. See "Configuring Greenplum Environment Variables" on page 24.
- `GPDB-LICENSE.txt` — Greenplum license agreement.
- **bin** — This directory contains the Greenplum Database management utilities. This directory also contains the PostgreSQL client and server programs, most of which are also used in Greenplum Database.
- **demo** — This directory contains the Greenplum demonstration programs.
- **docs** — The Greenplum Database documentation (PDF files).
- **etc** — Sample configuration file for OpenSSL.
- **ext** — Bundled programs (such as Python) used by some Greenplum Database utilities.
- **include** — The C header files for Greenplum Database.
- **lib** — Greenplum Database and PostgreSQL library files.
- **sbin** — Supporting/Internal scripts and programs.
- **share** — Shared files for Greenplum Database.

Creating the Greenplum Administrative User

You cannot run the Greenplum Database server as `root`. Greenplum recommends that you designate a user account that will own your Greenplum Database installation, and to always start and administer Greenplum Database as this user. For the purposes of this documentation, we will use the user name of `gpadmin`. You can choose any user name you like, but be sure to use the same user name consistently on all hosts in your Greenplum Database system.

To add a new user, for example, run the following commands as `root`:

```
# useradd gpadmin
# passwd gpadmin
# New password: password
# Retype new password: password
```

Creating the gpadmin User on Multiple Hosts at Once

Greenplum provides a utility called `gpssh` that allows you to open an ssh session and run commands on multiple hosts at once. This functionality allows you to perform administrative tasks on all of your segment hosts at the same time. `gpssh` requires a trusted-host environment (logging on to remote systems without a password prompt). In order to use `gpssh` to perform system administration tasks such as creating users, you must first exchange ssh keys as `root`. Greenplum provides a utility called `gpssh-exkeys`, which sets up a trusted host environment as the currently logged in user.

To exchange ssh keys as root

1. Log in as `root` on the master host, and source the `greenplum_path.sh` file from your Greenplum installation.

   ```
   # source /usr/local/greenplum-db/greenplum_path.sh
   ```

2. Create a host list file that has one host name per line and includes a host name for each host in your Greenplum system (master, standby master and segments). Make sure there are no blank lines or extra spaces. If a host has multiple configured host names, use only *one* host name per host. For example:

   ```
   mdw
   sdw1-1
   sdw2-1
   sdw3-1
   ```

3. Run the `gpssh-exkeys` utility referencing the host list file (*`all_hosts_file`*) you just created. For example:

   ```
   # gpssh-exkeys -f all_hosts_file
   ```

4. `gpssh-exkeys` will check the remote hosts and perform the key exchange between all hosts. Enter the `root` user password when prompted. For example:

   ```
   ***Enter password for root@hostname: root password
   ```

To create the `gpadmin` user

5. Run `gpssh` to create the `gpadmin` user on all hosts (if it does not exist already). Use the *`all_hosts_file`* you created in step 2. For example:

   ```
   # gpssh -f all_hosts_file '/usr/sbin/useradd gpadmin -d
   /home/gpadmin -s /bin/bash'
   ```

6. Set the new `gpadmin` user's password. On Linux, you can do this on all segment hosts at once using `gpssh`. For example:

   ```
   # gpssh -f all_hosts_file 'echo password | passwd gpadmin
   --stdin'
   ```

   On Solaris, you must log in to each segment host and set the `gpadmin` user's password on each host. For example:

   ```
   # ssh segment_hostname
   # passwd gpadmin
   # New password: password
   # Retype new password: password
   ```

7. After the `gpadmin` user is created, change the ownership of your Greenplum master installation directory to this user. For example:

   ```
   # chown -R gpadmin /usr/local/greenplum-db
   ```

Configuring Greenplum Environment Variables

You must configure your environment on the Greenplum Database master (and standby master). A `greenplum_path.sh` file is provided in your `$GPHOME` directory with environment variable settings for Greenplum Database. You can source this in the `gpadmin` user's startup shell profile (such as `.bashrc`).

For example, you could add a line similar to the following to your chosen profile files:

```
source /usr/local/greenplum-db/greenplum_path.sh
```

After editing the chosen profile file, source it as the correct user to make the changes active. For example:

```
$ source ~/.bashrc
```

Note: The `.bashrc` file should not produce any output. If you wish to have a message display to users upon logging in, use the `.profile` file.

Installing the Greenplum Binaries on Multiple Hosts

Greenplum provides a utility called `gpscp` that allows you to copy files to multiple hosts at once. Using `gpscp` and `gpssh`, you can install the Greenplum Database software on all of your segment hosts at once.

To install the Greenplum software on the segment hosts

1. On the master host, create a tar file of your Greenplum Database installation. For example (running as `root`):
   ```
   # su -
   # cd /usr/local
   # gtar -cvf /home/gpadmin/gp.tar greenplum-db-4.0.x.x
   ```
2. Create a host list file that has one host name per line and includes a host name for each *segment* host in your Greenplum system. Make sure there are no blank lines or extra spaces. If a host has multiple configured host names, use only *one* host name per host. For example:
   ```
   sdw1-1
   sdw2-1
   sdw3-1
   ```
3. Copy the tar file to the segment hosts using `gpscp`, where *`seg_hosts_file`* is the host list file you just created. For example:
   ```
   # gpscp -f seg_hosts_file /home/gpadmin/gp.tar =:/usr/local
   ```
4. Start an interactive session in `gpssh` using the same *`seg_hosts_file`* host list file. For example:
   ```
   # gpssh -f seg_hosts_file
   ```
5. At the `gpssh` command prompt, untar the tar file in the installation directory on the segment hosts. For example:
   ```
   => gtar --directory /usr/local -xvf /usr/local/gp.tar
   ```

6. Confirm that the Greenplum Database directory was installed in the correct location (the same location as `$GPHOME` on your master host). For example:

   ```
   => ls /usr/local/greenplum-db-4.0.x.x
   ```

7. Create a `greenplum-db` symbolic link to point to the current version directory of your Greenplum Database software. For example:

   ```
   => ln -s /usr/local/greenplum-db-4.0.x.x
   /usr/local/greenplum-db
   ```

8. Change the ownership of the Greenplum Database install directory to the `gpadmin` user. For example:

   ```
   => chown -R gpadmin /usr/local/greenplum-db
   ```

9. Remove the tar file. For example:

   ```
   => rm /usr/local/gp.tar
   ```

10. Exit `gpssh` interactive mode:

    ```
    => exit
    ```

Creating the Data Storage Areas

Every Greenplum Database master and segment instance has a designated storage area on disk that is called the *data directory* location. This is the file system location where the database data will be stored. Each master and segment instance needs its own designated data directory storage location.

To create the data directory location on the master

The data directory location on the master is different than those on the segments. The master does not store any user data, only the system catalog tables and system metadata are stored on the master instance, therefore you do not need to designate as much storage space as on the segments.

1. Create or choose a directory that will serve as your master data storage area. This directory should have sufficient disk space for your data and be owned by the `gpadmin` user and group. For example, run the following commands as `root`:

   ```
   # mkdir /gpmaster
   ```

2. Change ownership of this directory to the `gpadmin` user and group. For example:

   ```
   # chown gpadmin /gpmaster
   # chgrp gpadmin /gpmaster
   ```

To create the data directory location on a segment

On each segment host, create or choose the directories that each segment will use to store data. For example, if a segment host has two segments, run the following commands as `root`:

```
$ mkdir /gpdata1
$ mkdir /gpdata2
```

Change ownership of these directories to the `gpadmin` user and group. For example:

```
$ chown gpadmin /gpdata1
$ chgrp gpadmin /gpdata1
$ chown gpadmin /gpdata2
$ chgrp gpadmin /gpdata2
```

Setting up a Trusted Host Environment

The Greenplum Database management utilities require a trusted-host environment (logging on to remote systems without a password prompt). To perform Greenplum administration tasks using these utilities, you must first exchange ssh keys as the Greenplum administrative user (`gpadmin`). Greenplum provides a utility called `gpssh-exkeys`, which sets up a trusted host environment as the currently logged in user. When exchanging keys as `gpadmin`, you must use *all configured host names* for the hosts in your Greenplum Database system (master, standby master and segments).

To exchange SSH keys as the `gpadmin` user

1. Log in as `gpadmin`:

   ```
   $ su - gpadmin
   ```

2. Create a host list file that has one host name per line and includes all host names for all hosts in your Greenplum system. Make sure there are no blank lines or extra spaces. If a host has multiple configured host names, use *all* of the configured host names. For example:

   ```
   mdw
   mdw-1
   mdw-2
   mdw-3
   mdw-4
   sdw1-1
   sdw1-2
   sdw1-3
   sdw1-4
   sdw2-1
   sdw2-2
   sdw2-3
   sdw2-4
   sdw3-1
   sdw3-2
   sdw3-3
   sdw3-4
   ```

3. Run the `gpssh-exkeys` utility referencing the host list file (*`all_host_interfaces_file`*) you just created:

```
$ gpssh-exkeys -f all_host_interfaces_file
```

4. `gpssh-exkeys` will check the remote hosts and perform the key exchange between all hosts. Enter the `gpadmin` user password when prompted. For example:

```
***Enter password for gpadmin@hostname: gpadmin password
```

Synchronizing System Clocks

Greenplum recommends that you synchronize the system clocks on all hosts in the array using NTP (Network Time Protocol) or a similar utility. See www.ntp.org for more information about NTP. To see if the system clocks are synchronized, run the `date` command using `gpssh`. For example:

```
$ gpssh -f seg_hosts_file -v date
```

If you have the NTP daemon installed on your segment hosts, you can use it to synchronize the system clocks. For example:

```
$ gpssh -f seg_hosts_file -v ntpd
```

Next Steps

After you have configured the operating system environment and installed the Greenplum Database software on all of the hosts in the system, the next steps are:

- “Validating Your Systems” on page 29
- “Initializing a Greenplum Database System” on page 39

4. Validating Your Systems

Greenplum provides the following utilities to validate the configuration and performance of your systems:

- `gpcheck`
- `gpcheckperf`

These utilities can be found in `$GPHOME/bin` of your Greenplum installation.

The following tests should be run prior to initializing your Greenplum Database system.

- Validating OS Settings
- Validating Hardware Performance

Validating OS Settings

Greenplum provides a utility called `gpcheck` that can be used to verify that all hosts in your array have the correct OS settings for running the Greenplum Database software. To run `gpcheck`:

1. Log in on the master host as the user who will be running your Greenplum Database system (for example, `gpadmin`).
2. Create a host file that has the host names of all hosts to include in the verification test (one host name per line). Make sure there are no blank lines or extra spaces. This file should just have a *single* host name per host. For example:
   ```
   mdw
   smdw
   sdw1
   sdw2
   sdw3
   ```
3. Run the `gpcheck` utility using the host file you just created. For example:
   ```
   $ gpcheck -f host_file -m mdw -s smdw
   ```

Validating Hardware Performance

Greenplum provides a management utility called `gpcheckperf`, which can be used to identify hardware and system-level issues on the machines in your Greenplum Database array. `gpcheckperf` starts a session on the specified hosts and runs the following performance tests:

- Network Performance (`gpnetbench*`)
- Disk I/O Performance (`dd` test)
- Memory Bandwidth (`stream` test)

Before using `gpcheckperf`, you must have a trusted host setup between the hosts involved in the performance test. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already. Note that `gpcheckperf` calls to `gpssh` and `gpscp`, so these Greenplum utilities must be in your `$PATH`.

Validating Network Performance

To test network performance, run `gpcheckperf` with one of the network test run options: parallel pair test (`-r N`), serial pair test (`-r n`), or full matrix test (`-r M`). The utility runs a network benchmark program that transfers a 5 second stream of data from the current host to each remote host included in the test. By default, the data is transferred in parallel to each remote host and the minimum, maximum, average and median network transfer rates are reported in megabytes (MB) per second. If the summary transfer rate is slower than expected (less than 100 MB/s), you can run the network test serially using the `-r n` option to obtain per-host results. To run a full-matrix bandwidth test, you can specify `-r M` which will cause every host to send and receive data from every other host specified. This test is best used to validate if the switch fabric can tolerate a full-matrix workload.

Most systems in a Greenplum Database array are configured with multiple network interface cards (NICs), each NIC on its own subnet. When testing network performance, it is important to test each subnet individually. For example, considering the following standard network configuration of four NICs per host:

Table 4.1 Example Network Interface Configuration

Greenplum Host	Subnet1 NICs	Subnet2 NICs	Subnet3 NICs	Subnet4 NICs
Segment 1	sdw1-1	sdw1-2	sdw1-3	sdw1-4
Segment 2	sdw2-1	sdw2-2	sdw2-3	sdw2-4
Segment 3	sdw3-1	sdw3-2	sdw3-3	sdw3-4

You would create four distinct host files for use with the `gpcheckperf` network test:

Table 4.2 Example Network Test Host File Contents

host_file_nic1	host_file_nic2	host_file_nic3	host_file_nic4
sdw1-1	sdw1-2	sdw1-3	sdw1-4
sdw2-1	sdw2-2	sdw2-3	sdw2-4
sdw3-1	sdw3-2	sdw3-3	sdw3-4

You would then run `gpcheckperf` once per subnet. For example (if testing an *even* number of hosts, run in parallel pairs test mode):

```
$ gpcheckperf -f host_file_nic1 -r N -d /tmp > subnet1.out
$ gpcheckperf -f host_file_nic2 -r N -d /tmp > subnet2.out
$ gpcheckperf -f host_file_nic3 -r N -d /tmp > subnet3.out
$ gpcheckperf -f host_file_nic4 -r N -d /tmp > subnet4.out
```

If you have an *odd* number of hosts to test, you can run in serial test mode (`-r n`).

Validating Disk I/O and Memory Bandwidth

To test disk and memory bandwidth performance, run `gpcheckperf` with the disk and stream test run options (`-r ds`). The disk test uses the **dd** command (a standard UNIX utility) to test the sequential throughput performance of a logical disk or file system. The memory test uses the STREAM benchmark program to measure sustainable memory bandwidth. Results are reported in MB per second (MB/s).

To run the disk and stream tests

1. Create a host file that has one host name per segment host. Do not include the master host. For example:

```
sdw1-1
sdw2-1
sdw3-1
sdw4-1
```

2. Run the `gpcheckperf` utility using the host file you just created. Use the `-d` option to specify the file systems you want to test on each host (you must have write access to these directories). You will want to test your primary and mirror segment data directory locations. For example:

```
$ gpcheckperf -f seg_host_file -r ds -D -d /data/gpdb_p1 \
  -d  /data/gpdb_p2 -d /data/gpdb_p3 -d  /data/gpdb_p4 \
  -d /data/gpdb_m1 -d  /data/gpdb_m2 -d /data/gpdb_m3 \
  -d  /data/gpdb_m4
```

3. The utility may take a while to perform the tests as it is copying very large files between the hosts. When it is finished you will see the summary results for the Disk Write, Disk Read, and Stream tests.

5. Configuring Localization Settings

This chapter describes the available localization features of Greenplum Database. Greenplum Database supports localization with two approaches:

- Using the locale features of the operating system to provide locale-specific collation order, number formatting, and so on.
- Providing a number of different character sets defined in the Greenplum Database server, including multiple-byte character sets, to support storing text in all kinds of languages, and providing character set translation between client and server.

About Locale Support in Greenplum Database

Locale support refers to an application respecting cultural preferences regarding alphabets, sorting, number formatting, etc. Greenplum Database uses the standard ISO C and POSIX locale facilities provided by the server operating system. For additional information refer to the documentation of your operating system.

Locale support is automatically initialized when a Greenplum Database system is initialized. The initialization utility, `gpinitsystem`, will initialize the Greenplum array with the locale setting of its execution environment by default, so if your system is already set to use the locale that you want in your Greenplum Database system then there is nothing else you need to do. If you want to use a different locale (or you are not sure which locale your system is set to), you can instruct `gpinitsystem` exactly which locale to use by specifying the `-n` *`locale`* option. For example:

```
gpinitsystem -c gp_init_config -n sv_SE
```

This example sets the locale to Swedish (sv) as spoken in Sweden (SE). Other possibilities might be `en_US` (U.S. English) and `fr_CA` (French Canadian). If more than one character set can be useful for a locale then the specifications look like this: `cs_CZ.ISO8859-2`. What locales are available under what names on your system depends on what was provided by the operating system vendor and what was installed. On most systems, the command `locale -a` will provide a list of available locales.

Occasionally it is useful to mix rules from several locales, for example use English collation rules but Spanish messages. To support that, a set of locale subcategories exist that control only a certain aspect of the localization rules:

- `LC_COLLATE` — String sort order
- `LC_CTYPE` — Character classification (What is a letter? Its upper-case equivalent?)
- `LC_MESSAGES` — Language of messages
- `LC_MONETARY` — Formatting of currency amounts
- `LC_NUMERIC` — Formatting of numbers
- `LC_TIME` — Formatting of dates and times

If you want the system to behave as if it had no locale support, use the special locale `C` or `POSIX`.

The nature of some locale categories is that their value has to be fixed for the lifetime of a Greenplum Database system. That is, once `gpinitsystem` has run, you cannot change them anymore. `LC_COLLATE` and `LC_CTYPE` are those categories. They affect the sort order of indexes, so they must be kept fixed, or indexes on text columns will become corrupt. Greenplum Database enforces this by recording the values of `LC_COLLATE` and `LC_CTYPE` that are seen by `gpinitsystem`. The server automatically adopts those two values based on the locale that was chosen at initialization time.

The other locale categories can be changed as desired whenever the server is running by setting the server configuration parameters that have the same name as the locale categories (see the *Greenplum Database Administrator Guide* for more information on setting server configuration parameters). The defaults that are chosen by `gpinitsystem` are written into the master and segment `postgresql.conf` configuration files to serve as defaults when the Greenplum Database system is started. If you delete these assignments from the master and each segment `postgresql.conf` files then the server will inherit the settings from its execution environment.

Note that the locale behavior of the server is determined by the environment variables seen by the server, not by the environment of any client. Therefore, be careful to configure the correct locale settings on each Greenplum Database host (master and segments) before starting the system. A consequence of this is that if client and server are set up in different locales, messages may appear in different languages depending on where they originated.

Inheriting the locale from the execution environment means the following on most operating systems: For a given locale category, say the collation, the following environment variables are consulted in this order until one is found to be set: `LC_ALL`, `LC_COLLATE` (the variable corresponding to the respective category), `LANG`. If none of these environment variables are set then the locale defaults to `C`.

Some message localization libraries also look at the environment variable `LANGUAGE` which overrides all other locale settings for the purpose of setting the language of messages. If in doubt, please refer to the documentation of your operating system, in particular the documentation about `gettext`, for more information.

Native language support (NLS), which enables messages to be translated to the user’s preferred language, is not enabled in Greenplum Database for languages other than English. This is independent of the other locale support.

Locale Behavior

The locale settings influence the following SQL features:

- Sort order in queries using `ORDER BY` on textual data
- The ability to use indexes with `LIKE` clauses
- The `upper`, `lower`, and `initcap` functions
- The `to_char` family of functions

The drawback of using locales other than C or POSIX in Greenplum Database is its performance impact. It slows character handling and prevents ordinary indexes from being used by LIKE. For this reason use locales only if you actually need them.

Troubleshooting Locales

If locale support does not work as expected, check that the locale support in your operating system is correctly configured. To check what locales are installed on your system, you may use the command `locale -a` if your operating system provides it.

Check that Greenplum Database is actually using the locale that you think it is. LC_COLLATE and LC_CTYPE settings are determined at initialization time and cannot be changed without redoing gpinitsystem. Other locale settings including LC_MESSAGES and LC_MONETARY are initially determined by the operating system environment of the master and/or segment host, but can be changed after initialization by editing the postgresql.conf file of each Greenplum master and segment instance. You can check the active locale settings of the master host using the SHOW command. Note that every host in your Greenplum Database array should be using identical locale settings.

Character Set Support

The character set support in Greenplum Database allows you to store text in a variety of character sets, including single-byte character sets such as the ISO 8859 series and multiple-byte character sets such as EUC (Extended Unix Code), UTF-8, and Mule internal code. All supported character sets can be used transparently by clients, but a few are not supported for use within the server (that is, as a server-side encoding). The default character set is selected while initializing your Greenplum Database array using gpinitsystem. It can be overridden when you create a database, so you can have multiple databases each with a different character set.

Table 5.1 Greenplum Database Character Sets[1]

Name	Description	Language	Server?	Bytes/Char	Aliases
BIG5	Big Five	Traditional Chinese	No	1-2	WIN950, Windows950
EUC_CN	Extended UNIX Code-CN	Simplified Chinese	Yes	1-3	
EUC_JP	Extended UNIX Code-JP	Japanese	Yes	1-3	
EUC_KR	Extended UNIX Code-KR	Korean	Yes	1-3	
EUC_TW	Extended UNIX Code-TW	Traditional Chinese, Taiwanese	Yes	1-3	
GB18030	National Standard	Chinese	No	1-2	
GBK	Extended National Standard	Simplified Chinese	No	1-2	WIN936, Windows936
ISO_8859_5	ISO 8859-5, ECMA 113	Latin/Cyrillic	Yes	1	
ISO_8859_6	ISO 8859-6, ECMA 114	Latin/Arabic	Yes	1	

Table 5.1 Greenplum Database Character Sets[1]

Name	Description	Language	Server?	Bytes/Char	Aliases
ISO_8859_7	ISO 8859-7, ECMA 118	Latin/Greek	Yes	1	
ISO_8859_8	ISO 8859-8, ECMA 121	Latin/Hebrew	Yes	1	
JOHAB	JOHA	Korean (Hangul)	Yes	1-3	
KOI8	KOI8-R(U)	Cyrillic	Yes	1	KOI8R
LATIN1	ISO 8859-1, ECMA 94	Western European	Yes	1	ISO88591
LATIN2	ISO 8859-2, ECMA 94	Central European	Yes	1	ISO88592
LATIN3	ISO 8859-3, ECMA 94	South European	Yes	1	ISO88593
LATIN4	ISO 8859-4, ECMA 94	North European	Yes	1	ISO88594
LATIN5	ISO 8859-9, ECMA 128	Turkish	Yes	1	ISO88599
LATIN6	ISO 8859-10, ECMA 144	Nordic	Yes	1	ISO885910
LATIN7	ISO 8859-13	Baltic	Yes	1	ISO885913
LATIN8	ISO 8859-14	Celtic	Yes	1	ISO885914
LATIN9	ISO 8859-15	LATIN1 with Euro and accents	Yes	1	ISO885915
LATIN10	ISO 8859-16, ASRO SR 14111	Romanian	Yes	1	ISO885916
MULE_INTERNAL	Mule internal code	Multilingual Emacs	Yes	1-4	
SJIS	Shift JIS	Japanese	No	1-2	Mskanji, ShiftJIS, WIN932, Windows932
SQL_ASCII	unspecified[2]	any	No	1	
UHC	Unified Hangul Code	Korean	No	1-2	WIN949, Windows949
UTF8	Unicode, 8-bit	all	Yes	1-4	Unicode
WIN866	Windows CP866	Cyrillic	Yes	1	ALT
WIN874	Windows CP874	Thai	Yes	1	
WIN1250	Windows CP1250	Central European	Yes	1	
WIN1251	Windows CP1251	Cyrillic	Yes	1	WIN
WIN1252	Windows CP1252	Western European	Yes	1	
WIN1253	Windows CP1253	Greek	Yes	1	
WIN1254	Windows CP1254	Turkish	Yes	1	
WIN1255	Windows CP1255	Hebrew	Yes	1	
WIN1256	Windows CP1256	Arabic	Yes	1	
WIN1257	Windows CP1257	Baltic	Yes	1	
WIN1258	Windows CP1258	Vietnamese	Yes	1	ABC, TCVN, TCVN5712, VSCII

1. Not all APIs support all the listed character sets. For example, the JDBC driver does not support MULE_INTERNAL, LATIN6, LATIN8, and LATIN10.
2. The SQL_ASCII setting behaves considerably differently from the other settings. Byte values 0-127 are interpreted according to the ASCII standard, while byte values 128-255 are taken as uninterpreted characters. If you are working with any non-ASCII data, it is unwise to use the SQL_ASCII setting as a client encoding. SQL_ASCII is not supported as a server encoding.

Setting the Character Set

`gpinitsystem` defines the default character set for a Greenplum Database system by reading the setting of the `ENCODING` parameter in the `gp_init_config` file at initialization time. The default character set is `UNICODE` or `UTF8`.

You can create a database with a different character set besides what is used as the system-wide default. For example:

```
=> CREATE DATABASE korean WITH ENCODING 'EUC_KR';
```

Important: Although you can specify any encoding you want for a database, it is unwise to choose an encoding that is not what is expected by the locale you have selected. The `LC_COLLATE` and `LC_CTYPE` settings imply a particular encoding, and locale-dependent operations (such as sorting) are likely to misinterpret data that is in an incompatible encoding.

Since these locale settings are frozen by `gpinitsystem`, the apparent flexibility to use different encodings in different databases is more theoretical than real.

One way to use multiple encodings safely is to set the locale to `C` or `POSIX` during initialization time, thus disabling any real locale awareness.

Character Set Conversion Between Server and Client

Greenplum Database supports automatic character set conversion between server and client for certain character set combinations. The conversion information is stored in the master *pg_conversion* system catalog table. Greenplum Database comes with some predefined conversions or you can create a new conversion using the SQL command `CREATE CONVERSION`.

Table 5.2 Client/Server Character Set Conversions

Server Character Set	Available Client Character Sets
BIG5	not supported as a server encoding
EUC_CN	EUC_CN, MULE_INTERNAL, UTF8
EUC_JP	EUC_JP, MULE_INTERNAL, SJIS, UTF8
EUC_KR	EUC_KR, MULE_INTERNAL, UTF8
EUC_TW	EUC_TW, BIG5, MULE_INTERNAL, UTF8
GB18030	not supported as a server encoding
GBK	not supported as a server encoding
ISO_8859_5	ISO_8859_5, KOI8, MULE_INTERNAL, UTF8, WIN866, WIN1251
ISO_8859_6	ISO_8859_6, UTF8

Table 5.2 Client/Server Character Set Conversions

Server Character Set	Available Client Character Sets
ISO_8859_7	ISO_8859_7, UTF8
ISO_8859_8	ISO_8859_8, UTF8
JOHAB	JOHAB, UTF8
KOI8	KOI8, ISO_8859_5, MULE_INTERNAL, UTF8, WIN866, WIN1251
LATIN1	LATIN1, MULE_INTERNAL, UTF8
LATIN2	LATIN2, MULE_INTERNAL, UTF8, WIN1250
LATIN3	LATIN3, MULE_INTERNAL, UTF8
LATIN4	LATIN4, MULE_INTERNAL, UTF8
LATIN5	LATIN5, UTF8
LATIN6	LATIN6, UTF8
LATIN7	LATIN7, UTF8
LATIN8	LATIN8, UTF8
LATIN9	LATIN9, UTF8
LATIN10	LATIN10, UTF8
MULE_INTERNAL	MULE_INTERNAL, BIG5, EUC_CN, EUC_JP, EUC_KR, EUC_TW, ISO_8859_5, KOI8, LATIN1 to LATIN4, SJIS, WIN866, WIN1250, WIN1251
SJIS	not supported as a server encoding
SQL_ASCII	not supported as a server encoding
UHC	not supported as a server encoding
UTF8	all supported encodings
WIN866	WIN866
ISO_8859_5	KOI8, MULE_INTERNAL, UTF8, WIN1251
WIN874	WIN874, UTF8
WIN1250	WIN1250, LATIN2, MULE_INTERNAL, UTF8
WIN1251	WIN1251, ISO_8859_5, KOI8, MULE_INTERNAL, UTF8, WIN866
WIN1252	WIN1252, UTF8
WIN1253	WIN1253, UTF8
WIN1254	WIN1254, UTF8
WIN1255	WIN1255, UTF8
WIN1256	WIN1256, UTF8
WIN1257	WIN1257, UTF8
WIN1258	WIN1258, UTF8

To enable automatic character set conversion, you have to tell Greenplum Database the character set (encoding) you would like to use in the client. There are several ways to accomplish this:

- Using the `\encoding` command in `psql`, which allows you to change client encoding on the fly.
- Using `SET client_encoding TO`. Setting the client encoding can be done with this SQL command:

  ```
  => SET CLIENT_ENCODING TO 'value';
  ```

  To query the current client encoding:

  ```
  => SHOW client_encoding;
  ```

  To return to the default encoding:

  ```
  => RESET client_encoding;
  ```

- Using the `PGCLIENTENCODING` environment variable. When `PGCLIENTENCODING` is defined in the client's environment, that client encoding is automatically selected when a connection to the server is made. (This can subsequently be overridden using any of the other methods mentioned above.)
- Setting the configuration parameter `client_encoding`. If `client_encoding` is set in the master `postgresql.conf` file, that client encoding is automatically selected when a connection to Greenplum Database is made. (This can subsequently be overridden using any of the other methods mentioned above.)

If the conversion of a particular character is not possible — suppose you chose `EUC_JP` for the server and `LATIN1` for the client, then some Japanese characters do not have a representation in `LATIN1` — then an error is reported.

If the client character set is defined as `SQL_ASCII`, encoding conversion is disabled, regardless of the server's character set. The use of `SQL_ASCII` is unwise unless you are working with all-ASCII data. `SQL_ASCII` is not supported as a server encoding.

6. Initializing a Greenplum Database System

This chapter describes how to initialize a Greenplum Database database system. The instructions in this chapter assume you have already installed the Greenplum Database software on all of the hosts in the system according to the instructions in Chapter 3, "Configuring your Systems for Greenplum".

This chapter contains the following topics:

- Overview
- Initializing Greenplum Database
- Next Steps

Overview

Because a Greenplum Database database is distributed across many machines, the process for initializing the database is different than in PostgreSQL. With a regular PostgreSQL DBMS, you run a utility called `initdb` which creates the data storage directories, generates the shared catalog tables and configuration files, and creates the *template1* database, which is the template used to create other databases.

In a Greenplum Database DBMS, each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS. Greenplum provides its own version of `initdb` called `gpinitsystem`, which takes care of initializing the database on the master and on each segment instance, and starting each instance in the correct order.

After the Greenplum Database database system has been initialized and started, you can then create and manage databases as you would in a regular PostgreSQL DBMS by connecting to the Greenplum master.

Initializing Greenplum Database

These are the high-level tasks for initializing Greenplum Database:

1. Make sure you have completed all of the installation tasks described in Chapter 3, "Configuring your Systems for Greenplum".
2. (optional) If you want a standby master host, make sure that host is installed and configured before you initialize.
3. Create a host file that contains the host name of each *segment* host. See "Creating a Host List File" on page 40.
4. Create your Greenplum Database system configuration file. See "Creating the Greenplum Database Configuration File" on page 41.

5. By default, Greenplum Database will be initialized using the locale of the master host system. Make sure this is the correct locale you want to use, as some locale options cannot be changed after initialization. See "Configuring Localization Settings" on page 32 for more information.

6. Run the Greenplum Database initialization utility on the master host. See "Running the Initialization Utility" on page 42.

Creating a Host List File

The host list file is used by the `gpinitsystem` system initialization utility to determine the hosts on which to create the Greenplum Database segment instances. This file has just the host names of the segment hosts. If using a a multi-NIC configuration, this file should have *all* per-interface host names for each segment host — one host name per line.

To create the host list file

Note: If available, you can reuse the *`seg_hosts_file`* you created in Chapter 3, "Configuring your Systems for Greenplum".

1. Create a file in any location you like. The examples in this documentation call this file *`seg_hosts_file`*. For example:

   ```
   $ vi seg_hosts_file
   ```

2. In this file add the host name of each *segment* host, one name per line, no extra lines or spaces. If using multiple network interfaces per segment host, you must initialize the array using *all* configured host names of each segment host. For example:

   ```
   sdw1-1
   sdw1-2
   sdw1-3
   sdw1-4
   sdw2-1
   sdw2-2
   sdw2-3
   sdw2-4
   sdw3-1
   sdw3-2
   sdw3-3
   sdw3-4
   ```

3. Save and close the file.

4. If you created this file as `root`, make sure to change the ownership to the `gpadmin` user or group. For example:

   ```
   # chown gpadmin seg_hosts_file
   ```

```
# chgrp gpadmin seg_hosts_file
```

5. Note the location where this file resides, as you will need to specify it for the `MACHINE_LIST_FILE` parameter in the next task, "Creating the Greenplum Database Configuration File" on page 41.

Creating the Greenplum Database Configuration File

Your Greenplum Database configuration file tells the `gpinitsystem` initialization utility how you want to configure your Greenplum Database system. An example configuration file can be found in `$GPHOME/docs/cli_help/gp_init_config_example`. Also see "Initialization Configuration File Format" on page 77 for a detailed description of each parameter.

To create a gp_init_config file

1. Make a copy of the `gp_init_config_example` document to use as a starting point. For example:

   ```
   $ cp $GPHOME/docs/cli_help/gp_init_config_example
   /home/gpadmin/gp_init_config
   ```

2. Open the file you just copied in a text editor. For example:

   ```
   $ vi gp_init_config
   ```

3. Set all of the required parameters according to your environment. See "Initialization Configuration File Format" on page 77 for more information. A Greenplum Database system must contain a master instance and *at least two* segment instances (even if setting up a demo system on a single host).

   When creating the configuration file used by `gpinitsystem`, make sure you specify the correct number of segment instance data directories per host. Here is an example of the required parameters in the `gpdb_init_config` file:

   ```
   ARRAY_NAME="Greenplum"
   MACHINE_LIST_FILE=/home/gpadmin/multi_seg_hosts_file
   SEG_PREFIX=gpseg
   PORT_BASE=50000
   declare -a DATA_DIRECTORY=(/gpdata1 /gpdata2
   /gpdata3 /gpdata4)
   MASTER_HOSTNAME=mdw1
   MASTER_DIRECTORY=/gpmaster
   MASTER_PORT=5432
   TRUSTED SHELL=ssh
   CHECK_POINT_SEGMENT=8
   ENCODING=UNICODE
   ```

4. (optional) If you want to deploy mirror segments, set the mirroring parameters according to your environment. See "Initialization Configuration File Format" on page 77 for more information.

   You can also deploy mirrors later using the `gpaddmirrors` utility.

5. Save and close the file.

Running the Initialization Utility

The `gpinitsystem` utility will create a Greenplum Database system using the values defined in the configuration file.

To run the initialization utility

1. Run the following command referencing the path and file name of your initialization configuration file (`gp_init_config`). For example:

   ```
   $ gpinitsystem -c /home/gpadmin/gp_init_config
   ```

   If deploying an optional standby master, you would run:

   ```
   $ gpinitsystem -c /home/gpadmin/gp_init_config -s
   standby_master_hostname
   ```

2. The utility will verify your setup information and make sure it can connect to each host and access the data directories specified in your configuration. If all of the pre-checks are successful, the utility will prompt you to confirm your configuration. For example:

   ```
   => Continue with Greenplum creation? y
   ```

3. The utility will then begin setup and initialization of the master instance and each segment instance in the system. Each segment instance is set up in parallel. Depending on the number of segments, this process can take a while.

4. At the end of a successful setup, the utility will start your Greenplum Database system. You should see:

   ```
   => Greenplum Database instance successfully created.
   ```

Troubleshooting Initialization Problems

If the utility encounters any errors while setting up an instance, the entire process will fail, and could possibly leave you with a partially created system. Refer to the error messages and logs to determine the cause of the failure and where in the process the failure occurred. Log files are created in `~/gpAdminLogs`.

Depending on when the error occurred in the process, you may need to clean up and then try the `gpinitsystem` utility again. For example, if some segment instances were created and some failed, you may need to stop `postgres` processes and remove any utility-created data directories from your data storage area(s). A backout script is created to help with this cleanup if necessary.

Using the Backout Script

If the `gpinitsystem` utility fails, it will create the following backout script if it has left your system in a partially installed state:

```
~/gpAdminLogs/backout_gpinitsystem_<user>_<timestamp>
```

You can use this script to clean up a partially created Greenplum Database system. This backout script will remove any utility-created data directories, `postgres` processes, and log files. After correcting the error that caused `gpinitsystem` to fail and running the backout script, you should be ready to retry initializing your Greenplum Database array.

The following example shows how to run the backout script:

```
$ sh backout_gpinitsystem_gpadmin_20071031_121053
```

Setting the Master Data Directory Environment Variable

The Greenplum Database management utilities require that the `MASTER_DATA_DIRECTORY` environment variable be set. This should point to the directory created by the `gpinitsystem` utility in the master data directory location.

For example, you could add a line similar to the following to the `gpadmin` user's profile file (such as `.bashrc`):

```
MASTER_DATA_DIRECTORY=/gpmaster/gp-1

export MASTER_DATA_DIRECTORY
```

After editing the chosen profile file, source it as the correct user to make the changes active. For example:

```
$ source ~/.bashrc
```

Next Steps

After your system is up and running, the next steps are:

- Allowing Client Connections
- Creating Databases and Loading Data

Allowing Client Connections

After a Greenplum Database is first initialized it will only allow local connections to the database from the `gpadmin` role (or whatever system user ran `gpinitsystem`). If you would like other users or client machines to be able to connect to Greenplum Database, you must give them access. See the *Greenplum Database Administrator Guide* for more information.

Creating Databases and Loading Data

After verifying your installation, you may want to begin creating databases and loading data. See the *Greenplum Database Administrator Guide* for more information about creating databases, schemas, tables, and other database objects in Greenplum Database and loading your data.

7. Enabling the Performance Monitor

Greenplum provides an optional performance monitoring feature that administrators can enable within Greenplum Database 4.0.

Enabling Greenplum Performance Monitor is a two part process. First you must enable the Greenplum Database server to collect and store query performance data and system metrics. After data collection is enabled on the server side, the next step is to install and configure the Greenplum Performance Monitor console (a Web application used to view the performance monitor data stored in Greenplum Database). The Greenplum Performance Monitor console is shipped separately from your Greenplum Database 4.0 installation. You can download the Greenplum Performance Monitor console package and documentation from Greenplum Network. See the *Greenplum Database Performance Monitor Administrator Guide* for more information on installing and using the Greenplum Performance Monitor console.

This section describes how to create the Performance Monitor database and enable the Performance Monitor data collection agents. When the data collection agents are enabled, their processes are started and stopped along with the Greenplum Database server processes (using `gpstart` and `gpstop`).

Greenplum provides a `gpperfmon_install` utility that performs the following tasks:

- Creates the Performance Monitor database (`gpperfmon`).
- Creates the Performance Monitor superuser role (`gpmon`).
- Configures Greenplum Database to accept connections from the Performance Monitor superuser role (edits the `pg_hba.conf` and `.pgpass` files).
- Sets the Performance Monitor server configuration parameters in the Greenplum Database `postgresql.conf` files.

Install the Performance Monitor database and enable Performance Monitor agents

1. Log in to the Greenplum Database master as the `gpadmin` user.
   ```
   $ su - gpadmin
   ```
2. Run the `gpperfmon_install` utility with the `--enable` option. You must supply the connection port of the Greenplum Database master, and set the password for the `gpmon` superuser that will be created. For example:
   ```
   $ gpperfmon_install --enable --password p@$$word --port 5432
   ```
3. When the utility completes, restart Greenplum Database. The data collection agents will not start until the database is restarted:
   ```
   $ gpstop -r
   ```
4. Using the `ps` command, verify that the data collection process is running on the Greenplum master. For example:
   ```
   $ ps -ef | grep gpmmon
   ```

5. Run the following command to verify that the data collection processes are writing to the Performance Monitor database. If all of the segment data collection agents are running, you should see one row per segment host.

```
$ psql gpperfmon -c 'SELECT * FROM system_now;'
```

The data collection agents are now running, and your Greenplum Database system now has a `gpperfmon` database installed. This is the database where performance data is stored. You can connect to it as follows:

```
$ psql gpperfmon
```

To configure a standby master host (if enabled)

6. Copy the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file from your primary master host to your standby master host. This ensures that the required connection options are also set on the standby master.

7. Copy your `~/.pgpass` file from your primary master host to your standby master host. This file usually resides in the `gpadmin` user's home directory.

Greenplum Performance Monitor Console

The Performance Monitor Console provides a graphical interface for viewing performance data and for administering certain aspects of monitoring. Normally installed on the Greenplum Database master host, the console is a web-based user interface accessed through a supported browser.

The Performance Monitor Console is typically installed on the Greenplum Database master host. However, you have the option to install the console on a host different from the master host. Note that this setup incurs a performance penalty due to the numerous database connections the console must open over the network.

If you have multiple Greenplum Database instances, you can create separate monitor instances for each of them. Each separate console instance operates on a unique port and has its own unique configuration options.

The Performance Monitor Console is supported for the following browsers with Adobe Flash 9.0 or higher:

- Internet Explorer for Windows XP and Vista
- Mozilla Firefox for Windows and Linux
- Apple's Safari browser for Macintosh

The Performance Monitor Console runs on a lighttpd web server. The default web server port is 28080. For more information about the web server, see "Web Server Administration" on page 21.

Installing the Performance Monitor Console involves the following high-level tasks:

- Install Performance Monitor Console — Create the software installation directory.
- Set up Greenplum Performance Monitor Console — Create and configure a Performance Monitor Console instance and its supporting web services.

Install Performance Monitor Console

1. Download the installer file from Greenplum Network. Installer files are available for RedHat (32-bit and 64-bit), Solaris 64-bit or SuSE Linux 64 bit platforms.
2. Unzip the installer file where *PLATFORM* is either `RHEL5-i386` (RedHat 32-bit), `RHEL5-x86_64` (RedHat 64-bit), `SOL-x86_64` (Solaris 64-bit), or `SuSE10-x86_64` (SuSe Linux 64 bit). For example:
   ```
   # unzip greenplum-perfmon-web-4.0.x.x-PLATFORM.zip
   ```
3. Launch the installer using `bash`. For example:
   ```
   # /bin/bash greenplum-perfmon-web-4.0.x.x-PLATFORM.bin
   ```
4. Type `yes` to accept the license agreement.
5. The installer will prompt you to provide an installation path. Press `ENTER` to accept the default install path:
   (`/usr/local/greenplum-perfmon-web-4.0.x.x`)
   or enter an absolute path to an install location. You must have write permissions to the location you specify. This location is referred to as `$GPPERFMONHOME`.
6. If you ran the installer as `root`, change the ownership of the Console installation directory to the `gpadmin` user. For example:
   ```
   # chown -R gpadmin /usr/local/greenplum-perfmon-web-4.0.x.x
   ```
7. The installation directory contains a `gpperfmon_path.sh` file with path and environment settings for the Console. You must source this file in your `gpadmin` user's startup shell profile (such as `.bashrc`).

   For example, you could add a line similar to the following to your chosen profile files:
   ```
   source /usr/local/greenplum-perfmon-web-4.0.x.x/ \
   gpperfmon_path.sh
   ```
   After editing the chosen profile file, source it as the correct user to make the changes active. For example:
   ```
   $ source ~/.bashrc
   ```
8. Configure the Console as described in "Setting up the Performance Monitor Console".

Setting up the Performance Monitor Console

The `gpperfmon` utility sets up the Performance Monitor Console on the current host. On hosts other than the Greenplum Database master host, the console experiences slower performance due to frequent connections to the `gpperfmon` database.

During the setup process, the utility prompts you for values to configure Console connections to a single Greenplum Database instance. To configure connections to multiple Greenplum Database instances, run the setup routine multiple times. To accept the displayed default values for any of these parameters at configuration time, hit the `ENTER` key.

Set up Greenplum Performance Monitor Console

1. Log in as the Greenplum administrator (`gpadmin`).
2. With the Greenplum Database running, launch the setup utility. For example:

   ```
   $ gpperfmon --setup
   ```

3. Provide an instance name for the Greenplum Database instance monitored by this Console. To monitor multiple instances, you must run the setup utility separately for each instance.
4. Select `y` or `n` to specify if the Greenplum Database master for this instance is on a remote host. Console performance is better when the Console and Greenplum Database master are on the same host.

   If the master host is remote, enter `y` and enter the hostname of the master at the prompt.

Note: Verify that you can connect to the master host from the host on which you are installing the Performance Monitor Console. Enter:

```
psql -h master_host_name -p port -U gpmon
```

If you cannot establish a connection, make sure you create a `.pgpass` file on the host running the Performance Monitor Console as described in "Configuring Authentication for the Monitor Console" on page 49.

5. Provide a display name for the instance. This name is shown in the Console user interface. This prompt does not display if the master host is remote.
6. Provide the port for the Greenplum Database master instance.
7. Provide a port number for the Performance Monitor Console web server. The default is `28080`.
8. (Optional) Enter `y` or `n` to set up SSL connections for the Performance Monitor Console. If you enter `Y`, you are prompted for the following distinguished name (DN) information used to create an unsigned certificate for the Performance Monitor Console. For example:

   ```
   Country Name (2 letter code) [GB]:US
   State or Province Name (full name) [Berkshire]:California
   Locality Name (eg, city) [Newbury]:San Mateo
   Organization Name (eg, company) [My Company Ltd]:Greenplum
   Organizational Unit Name (eg, section) []:Engineering
   Common Name (eg, your name or your server's hostname) []:mdw1
   Email Address []:gpadmin@greenplum.com
   ```

Note: Because database login information is sent over the network, Greenplum recommends using SSL to encrypt these communications.

9. Start and log into the Console. See "Connecting to the Greenplum Performance Monitor Console".

You can also configure authentication so that other database users can log into the Console, as described in "Configuring Authentication for the Monitor Console" on page 49.

Connecting to the Greenplum Performance Monitor Console

Start the Greenplum Performance Monitor Console by entering:

```
gpperfmon --start
```

If you do not specify an instance name, all Performance Monitor Console instances are started. To start a particular instance, you can specify the name of the instance. For example:

```
gpperfmon --start "instance_name"
```

See "Starting and Stopping Greenplum Performance Monitor" on page 19 for a complete list of commands.

After the instance is running, you can open the Performance Monitor Console in a supported browser using the correct hostname and port. For example, to open a monitor instance running on port 28080 on the local host with SSL, use the following web address:

```
https://localhost:28080/
```

If connecting to a monitor instance using SSL on port 28080, enter:

```
https://monitor_console_hostname:28080/
```

At the login prompt, enter the user name and password of a Greenplum Database role that has been properly configured to allow authentication to Greenplum Performance Monitor (for example, `gpmon`), then click **Login**. This opens the **Dashboard** page of the Performance Monitor Console, which provides a graphical system snapshot and a summary view of active queries. See the Performance Monitor Console online help for more information.

You must be a Greenplum Database administrator to fully use the Greenplum Performance Monitor Console. Administrators can view information for all queries as well as system metrics, while regular database users can only monitor their own queries.

Configuring Authentication for the Monitor Console

When you installed Greenplum Performance Monitor database and enabled the data collection agents, a `gpmon` superuser was created by the installation utility. This is the database role used to manage the Performance Monitor components and data within Greenplum Database. The `gpmon` role is configured to use md5-encrypted password authentication to connect to Greenplum Database.

Typically, you will not be connecting to the Performance Monitor Console as `gpmon`, and instead connect as another database user (such as `gpadmin`). The Performance Monitor Console is configured by default to require md5-encrypted password authentication for all database users who want to log into the Performance Monitor Console.

To set up md5-encrypted password authentication for a database user who will login to the Console, do the following steps:

1. Make sure the database role has an md5-encrypted password set. Passwords can be assigned using `CREATE ROLE` or `ALTER ROLE`. For example:

   ```
   =# ALTER ROLE gpadmin WITH ENCRYPTED PASSWORD
   'password1234';
   ```

2. Edit `$MASTER_DATA_DIRECTORY/pg_hba.conf` to specify `md5` authentication for connections made by any role to the `gpperfmon` database. For example:

   ```
   host      gpperfmon     all    0.0.0.0/0    md5
   ```

3. In the `gpadmin` user's home directory on the master host, add an entry to the `.pgpass` file that has the following connection information:

   `master_hostname``:`*`master_port`*`:`*`dbname`*`:`*`rolename`*`:`*`password`*

   For example (where * means any configured master hostname):

   ```
   *:5432:gpperfmon:gpadmin:password1234
   ```

4. Save and close the `.pgpass` file.

5. Use `gpstop` to reload the `pg_hba.conf` file:

   ```
   $ gpstop -u
   ```

Using Trust Authentication

You can bypass password authentication by using `trust` authentication. Trust authentication allows database users to login to the Performance Monitor Console without authenticating, and is recommended only for testing purposes. It should not be used in production Greenplum Database environments. Greenplum recommends using password authentication with SSL for production installations.

To allow trust access

1. Set `allow_trust_logon` to `yes` in the `gpperfmonui.conf` file (Greenplum Performance Monitor configuration file) located in:

   `$GPPERFMONHOME/instances/`*`instance_name`*

   where *`instance_name`* is the name of the instance that you specified when you ran `gpperfmon --setup`.

2. Edit `$MASTER_DATA_DIRECTORY/pg_hba.conf` to specify `trust` authentication for connections made by any role to the `gpperfmon` database. For example:

   ```
   host      gpperfmon     all    0.0.0.0/0    trust
   ```

3. Use `gpstop` to reload the `pg_hba.conf` file:

   ```
   $ gpstop -u
   ```

Using SSL Encryption

If you enable SSL at setup time, the installer creates a self-signed certificate and uses OpenSSL encryption for all connections to the monitor web server.

Because this is a self-signed certificate, supported browsers may need to be configured to accept or allow it as a security exception. This does not detract from the added security gained from encrypting communications between the browser client and the web server.

Optionally, you can obtain and use a certificate signed by a trusted certificate authority such as Verisign. If you use a trusted certificate, edit the `lighttpd.conf` file (located in `$GPPERFMONHOME/instances/`*`instance_name`*`/conf`), and set the `ssl.pemfile` to point to the location of the certificate. For example:

```
ssl.pemfile = "$GPPERFMONHOME/instances/instance_name/conf/cert.pem"
```

For more information on the `lighttpd.conf` file, see "Web Server Administration" on page 21.

Setting up Performance Monitor Console on a Standby Master

1. Install the Performance Monitor Console software on the standby master host using the instructions in "Installing the Greenplum Performance Monitor Console" on page 9.
2. Set up a new Performance Monitor Console instance on the master host using the instructions in "Setting up the Performance Monitor Console" on page 47. During the setup process, make sure to specify the following options specific to the standby master:
   - Specify a *different* instance name than the console instance running on the primary master host.
   - When prompted for the Greenplum Performance Monitor port, specify the *same* value as the primary master (the standby master must always use the same port as the primary).
   - If you configured the Greenplum Performance Monitor Console for SSL connections, make sure the `ssl.pemfile` parameter in the web server configuration file points to a valid certificate for the standby master host. You cannot use the same certificate file as the primary master host.
3. If you fail over to your standby master host, the data collection agents are restarted automatically upon activation of the standby. However, you must manually start the Performance Monitor Console using `gpperfmon --start`.

About the Performance Monitor Installation

The installation and setup procedures create a software installation directory and a directory containing files and folders to support each Greenplum Performance Monitor Console instance.

Software Installation Directory

The following files and first-level subdirectories are copied into the installation folder that you specified when you installed Greenplum Performance Monitor Console. This location is referred to as `$GPPERFMONHOME`.

- `gpperfmon_path.sh` — file containing environment variables for the monitor
- **bin** — program files for Greenplum Performance Monitor
- **docs** — documentation, including administration guide and release notes
- **etc** — contains `openssl.conf` file
- **ext** — Python directory and files
- **instances** — contains a subdirectory of resources for each Greenplum Database instance monitored by the console
- **lib** — library files for Greenplum Performance Monitor
- **www** — web service and user interface files

Instances Directory

The `$GPPERFMONHOME/instances` directory contains subdirectories named for each instance created during console setup. The `conf` subdirectory contains configuration files that you can edit. Other files and folders are used by the web services for the instance, and should not be modified or deleted.

Each subdirectory contains the following files and first-level subdirectories:

- `lighttpd.pid` -- file containing an updated process ID for the web server process for the instance
- **conf** — console and web server configuration files, `gpperfmonui.conf` and `lighttpd.conf`
- **logs** — logs for the web server for this instance
- `perfmon.fastcgi.socket-0` — dynamically updated socket information, which cannot be viewed or updated
- **sessions** — files storing information about session state
- **tmp** — temporary files used by the web server
- **web** — symbolic links to web server files in the installation directory

A. Installation Management Utilities

This appendix provides references for the command-line management utilities used to install and initialize a Greenplum Database system. For a full reference of all Greenplum Database utilities, see the *Greenplum Database Administrator Guide*.

The following Greenplum Database management utilities are located in `$GPHOME/bin`:

- gpactivatestandby
- gpaddmirrors
- gpcheck
- gpcheckperf
- gpdeletesystem
- gpinitstandby
- gpinitsystem
- gpscp
- gpssh
- gpssh-exkeys
- gpstart
- gpstop

gpactivatestandby

Activates a standby master host and makes it the active master for the Greenplum Database system.

Synopsis

```
gpactivatestandby -d standby_master_datadir
[-c new_standby_master] [-f] [-a] [-q] [-l logfile_directory]

gpactivatestandby -? | -h | --help

gpactivatestandby -v
```

Description

The `gpactivatestandby` utility activates a backup master host and brings it into operation as the active master instance for a Greenplum Database system. The activated standby master effectively becomes the Greenplum Database master, accepting client connections on the master port (which must be set to the same port number on the master host and the backup master host).

You must run this utility from the master host you are activating, not the failed master host you are disabling. Running this utility assumes you have a backup master host configured for the system (see `gpinitstandby`).

The utility will perform the following steps:

- Stop the synchronization process (`gpsyncagent`) on the backup master
- Update the system catalog tables of the backup master using the logs
- Activate the backup master to be the new active master for the system
- (optional) Make the host specified with the `-c` option the new standby master host
- Restart the Greenplum Database system with the new master host

A backup Greenplum master host serves as a 'warm standby' in the event of the primary Greenplum master host becoming unoperational. The backup master is kept up to date by a transaction log replication process (`gpsyncagent`), which runs on the backup master host and keeps the data between the primary and backup master hosts synchronized.

If the primary master fails, the log replication process is shutdown, and the backup master can be activated in its place by using the `gpactivatestandby` utility. Upon activation of the backup master, the replicated logs are used to reconstruct the state of the Greenplum master host at the time of the last successfully committed transaction. To specify a new standby master host after making your current standby master active, use the `-c` option.

In order to use `gpactivatestandby` to activate a new primary master host, the master host that was previously serving as the primary master cannot be running. The utility checks for a `postmaster.pid` file in the data directory of the disabled master host, and if it finds it there, it will assume the old master host is still active. In some cases, you may need to remove the `postmaster.pid` file from the disabled master host data directory before running `gpactivatestandby` (for example, if the disabled master host process was terminated unexpectedly).

After activating a standby master, run `ANALYZE` to update the database query statistics. For example:

```
psql dbname -c 'ANALYZE;'
```

Options

`-a (do not prompt)`

Do not prompt the user for confirmation.

`-c` ***`new_standby_master_hostname`***

Optional. After you activate your standby master you may want to specify another host to be the new standby, otherwise your Greenplum Database system will no longer have a standby master configured. Use this option to specify the hostname of the new standby master host. You can also use `gpinitstandby` at a later time to configure a new standby master host.

`-d` ***`standby_master_datadir`***

Required. The absolute path of the data directory for the master host you are activating.

`-f (force activation)`

Use this option to force activation of the backup master host when the synchronization process (`gpsyncagent`) is not running. Only use this option if you are sure that the backup and primary master hosts are consistent, and you know the `gpsyncagent` process is not running on the backup master host. This option may be useful if you have just initialized a new backup master using `gpinitstandby`, and want to activate it immediately.

`-l` ***`logfile_directory`***

The directory to write the log file. Defaults to `~/gpAdminLogs`.

`-q (no screen output)`

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

`-v (show utility version)`

Displays the version, status, last updated date, and check sum of this utility.

`-? | -h | --help (help)`

Displays the online help.

Examples

Activate the backup master host and make it the active master instance for a Greenplum Database system (run from backup master host you are activating):

```
gpactivatestandby -d /gpdata
```

Activate the backup master host and at the same time configure another host to be your new standby master:

```
gpactivatestandby -d /gpdata -c new_standby_hostname
```

See Also

gpinitsystem, gpinitstandby

gpaddmirrors

Adds mirror segments to a Greenplum Database system that was initially configured without mirroring.

Synopsis

gpaddmirrors [**-p** *port_offset*] [**-m** *datadir_config_file* [**-a**]] [**-s**] [**-d** *master_data_directory*] [**-B** *parallel_processes*] [**-l** *logfile_directory*] [**-v**]

gpaddmirrors -i *mirror_config_file* [**-s**] [**-a**] [**-d** *master_data_directory*] [**-B** *parallel_processes*] [**-l** *logfile_directory*] [**-v**]

gpaddmirrors -o *output_sample_mirror_config* [**-m** *datadir_config_file*]

gpaddmirrors -?

gpaddmirrors --version

Description

The `gpaddmirrors` utility configures mirror segment instances for an existing Greenplum Database system that was initially configured with primary segment instances only. The utility will create the mirror instances and begin the online replication process between the primary and mirror segment instances. Once all mirrors are synchronized with their primaries, your Greenplum Database system is fully data redundant.

By default, the utility will prompt you for the file system location(s) where it will create the mirror segment data directories. If you do not want to be prompted, you can pass in a file containing the file system locations using the `-m` option.

The mirror locations and ports must be different than your primary segment data locations and ports. If you have created additional filespaces, you will also be prompted for mirror locations for each of your filespaces.

The utility will create a unique data directory for each mirror segment instance in the specified location using the predefined naming convention. There must be the same number of file system locations declared for mirror segment instances as for primary segment instances. It is OK to specify the same directory name multiple times if you want your mirror data directories created in the same location, or you can enter a different data location for each mirror. Enter the absolute path. For example:

```
Enter mirror segment data directory location 1 of 2 > /gpdb/mirror
Enter mirror segment data directory location 2 of 2 > /gpdb/mirror
```

OR

```
Enter mirror segment data directory location 1 of 2 > /gpdb/m1
Enter mirror segment data directory location 2 of 2 > /gpdb/m2
```

Alternatively, you can run the `gpaddmirrors` utility and supply a detailed configuration file using the `-i` option. This is useful if you want your mirror segments on a completely different set of hosts than your primary segments. The format of the mirror configuration file is:

```
filespaceOrder=[filespace1_fsname[:filespace2_fsname:...]
mirror[content]=content:address:port:mir_replication_port:pri_
replication_port:fselocation[:fselocation:...]
```

For example (if you do not have additional filespaces configured besides the default *pg_system* filespace):

```
filespaceOrder=
mirror0=0:sdw1-1:60000:61000:62000:/gpdata/mir1/gp0
mirror1=1:sdw1-1:60001:61001:62001:/gpdata/mir2/gp1
```

The *gp_segment_configuration*, *pg_filespace*, and *pg_filespace_entry* system catalog tables can help you determine your current primary segment configuration so that you can plan your mirror segment configuration. For example, run the following query:

```
=# SELECT dbid, content, address as host_address, port,
   replication_port, fselocation as datadir
   FROM gp_segment_configuration, pg_filespace_entry
   WHERE dbid=fsedbid
   ORDER BY dbid;
```

If creating your mirrors on alternate mirror hosts, the new mirror segment hosts must be pre-installed with the Greenplum Database software and configured exactly the same as the existing primary segment hosts.

You must make sure that the user who runs `gpaddmirrors` (the `gpadmin` user) has permissions to write to the data directory locations specified. You may want to create these directories on the segment hosts and `chown` them to the appropriate user before running `gpaddmirrors`.

Options

-a (do not prompt)

Run in quiet mode - do not prompt for information. Must supply a configuration file with either `-m` or `-i` if this option is used.

-B *parallel_processes*

The number of mirror setup processes to start in parallel. If not specified, the utility will start up to 10 parallel processes depending on how many mirror segment instances it needs to set up.

-d *master_data_directory*

The master data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-i *mirror_config_file*

A configuration file containing one line for each mirror segment you want to create. You must have one mirror segment listed for each primary segment in the system. The format of this file is as follows (as per attributes in the *gp_segment_configuration*, *pg_filespace*, and *pg_filespace_entry* catalog tables):

```
filespaceOrder=[filespace1_fsname[:filespace2_fsname:...]
mirror[content]=content:address:port:mir_replication_port:pri_
replication_port:fselocation[:fselocation:...]
```

Note that you only need to specify an name for `filespaceOrder` if your system has multiple filespaces configured. If your system does not have additional filespaces configured besides the default *pg_system* filespace, this file will only have one location (for the default data directory filespace, *pg_system*). *pg_system* does not need to be listed in the `filespaceOrder` line. It will always be the first *`fselocation`* listed after *`replication_port`*.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m *datadir_config_file*

A configuration file containing a list of file system locations where the mirror data directories will be created. If not supplied, the utility will prompt you for locations. Each line in the file specifies a mirror data directory location. For example:

```
/gpdata/m1
/gpdata/m2
/gpdata/m3
/gpdata/m4
```

If your system has additional filespaces configured in addition to the default *pg_system* filespace, you must also list file system locations for each filespace as follows:

```
filespace filespace1
/gpfs1/m1
/gpfs1/m2
/gpfs1/m3
/gpfs1/m4
```

-o *output_sample_mirror_config*

If you are not sure how to lay out the mirror configuration file used by the **`-i`** option, you can run `gpaddmirrors` with this option to generate a sample mirror configuration file based on your primary segment configuration. The utility will prompt you for your mirror segment data directory locations (unless you provide these in a file using `-m`). You can then edit this file to change the host names to alternate mirror hosts if necessary.

-p *port_offset*

Optional. This number is used to calculate the database ports and replication ports used for mirror segments. The default offset is 1000. Mirror port assignments are calculated as follows:

primary port + offset = mirror database port

primary port + (2 * offset) = mirror replication port

primary port + (3 * offset) = primary replication port

For example, if a primary segment has port 50001, then its mirror will use a database port of 51001, a mirror replication port of 52001, and a primary replication port of 53001 by default.

-s (spread mirrors)

Spreads the mirror segments across the available hosts. The default is to group a set of mirror segments together on an alternate host from their primary segment set. Mirror spreading will place each mirror on a different host within the Greenplum Database array. Spreading is only allowed if there is a sufficient number of hosts in the array (number of hosts is greater than or equal to the number of segment instances per host).

-v (verbose)

Sets logging output to verbose.

--version (show utility version)

Displays the version of this utility.

-? (help)

Displays the online help.

Examples

Add mirroring to an existing Greenplum Database system using the same set of hosts as your primary data. Calculate the mirror database and replication ports by adding 100 to the current primary segment port numbers:

```
$ gpaddmirrors -p 100
```

Add mirroring to an existing Greenplum Database system using a different set of hosts from your primary data:

```
$ gpaddmirrors -i mirror_config_file
```

Where the *mirror_config_file* looks something like this (if you do not have additional filespaces configured besides the default *pg_system* filespace):

```
filespaceOrder=
mirror0=0:sdw1-1:52001:53001:54001:/gpdata/mir1/gp0
mirror1=1:sdw1-2:52002:53002:54002:/gpdata/mir2/gp1
mirror2=2:sdw2-1:52001:53001:54001:/gpdata/mir1/gp2
mirror3=3:sdw2-2:52002:53002:54002:/gpdata/mir2/gp3
```

Output a sample mirror configuration file to use with `gpaddmirrors -i`:

```
$ gpaddmirrors -o /home/gpadmin/sample_mirror_config
```

See Also

gpinitsystem, gpinitstandby, gpactivatestandby

gpcheck

Verifies and validates Greenplum Database platform settings.

Synopsis

```
gpcheck -f host_file [-m master_host] [-s standy_master_host]
          [--stdout | --zipout]

gpcheck --zipin file

gpcheck -?

gpcheck --version
```

Description

The `gpcheck` utility determines the platform on which you are running Greenplum Database and validates various platform-specific configuration settings. `gpcheck` can use a host list file or a file previously created with the `--zipout` option to validate platform settings. At the end of a successful validation process, `GPCHECK_NORMAL` message displays. If `GPCHECK_ERROR` displays, one or more validation checks failed. You can use also `gpcheck` to gather and view platform settings on hosts without running validation checks.

Greenplum recommends that you run `gpcheck` as `root`. If you do not run `gpcheck` as `root`, the utility displays a warning message and will not be able to validate all configuration settings; Only some of these settings will be validated.

Options

-f *host_file*

The name of a file that contains a list of hosts that `gpcheck` uses to validate platform-specific settings. This file should contain a single host name for all hosts in your Greenplum Database system (master, standby master, and segments).

-m *master_host*

Perform special master host-specific validation tasks on this host.

-s *standy_master_host*

Perform special standby master host-specific validation tasks on this host.

--stdout

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

--zipout

Save all collected data to a `.zip` file in the current working directory. `gpcheck` automatically creates the `.zip` file and names it `gpcheck_`*`timestamp`*`.tar.gz`. No checks or validations are performed.

--zipin *file*

Use this option to decompress and check a `.zip` file created with the `--zipout` option. `gpcheck` performs validation tasks against the file you specify in this option.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Verify and validate the Greenplum Database platform settings by entering a host file and specifying the master host and the standby master host.

```
# gpcheck -f myhostfile -m mdw -s smdw
```

Save Greenplum Database platform settings to a zip file.

```
# gpcheck -f myhostfile -m mdw -s smdw --zipout
```

Verify and validate the Greenplum Database platform settings using a zip file created with the `--zipout` option:

```
# gpcheck --zipin gpcheck_timestamp.tar.gz
```

View collected Greenplum Database platform settings:

```
# gpcheck -f myhostfile -m mdw -s smdw --stdout
```

See Also

gpssh, gpscp, gpcheckperf

gpcheckperf

Verifies the baseline hardware performance of the specified hosts.

Synopsis

```
gpcheckperf -d test_directory [-d test_directory ...]
           {-f host_file | - h hostname [-h hostname ...]}
           [-r ds{ n|N|M [--duration time]
                         [--netperf] }]
           [-B block_size] [-S file_size] [-D] [-v|-V]
```

```
gpcheckperf -?
```

```
gpcheckperf --version
```

Description

The `gpcheckperf` utility starts a session on the specified hosts and runs the following performance tests:

- **Disk I/O Test (`dd` test)** — To test the sequential throughput performance of a logical disk or file system, the utility uses the **`dd`** command, which is a standard UNIX utility. It times how long it takes to write and read a large file to and from disk and calculates your disk I/O performance in megabytes (MB) per second. By default, the file size that is used for the test is calculated at two times the total random access memory (RAM) on the host. This ensures that the test is truly testing disk I/O and not using the memory cache.
- **Memory Bandwidth Test (`stream`)** — To test memory bandwidth, the utility uses the STREAM benchmark program to measure sustainable memory bandwidth (in MB/s). This tests that your system is not limited in performance by the memory bandwidth of the system in relation to the computational performance of the CPU. In applications where the data set is large (as in Greenplum Database), low memory bandwidth is a major performance issue. If memory bandwidth is significantly lower than the theoretical bandwidth of the CPU, then it can cause the CPU to spend significant amounts of time waiting for data to arrive from system memory.
- **Network Performance Test (`gpnetbench*`)** — To test network performance (and thereby the performance of the Greenplum Database interconnect), the utility runs a network benchmark program that transfers a 5 second stream of data from the current host to each remote host included in the test. The data is transferred in parallel to each remote host and the minimum, maximum, average and median network transfer rates are reported in megabytes (MB) per second. If the summary transfer rate is slower than expected (less than 100 MB/s), you can run the network test serially using the `-r n` option to obtain per-host results. To run a full-matrix bandwidth test, you can specify `-r M` which will cause every host to send and receive data from every other host specified. This test is best used to validate if the switch fabric can tolerate a full-matrix workload.

To specify the hosts to test, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. If running the network performance test, all entries in the host file must be for network interfaces within the same subnet. If your segment hosts have multiple network interfaces configured on different subnets, run the network test once for each subnet.

You must also specify at least one test directory (with `-d`). The user who runs `gpcheckperf` must have write access to the specified test directories on all remote hosts. For the disk I/O test, the test directories should correspond to your segment data directories (primary and/or mirrors). For the memory bandwidth and network tests, one test directory is still required to copy over the test program files.

Before using `gpcheckperf`, you must have a trusted host setup between the hosts involved in the performance test. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already. Note that `gpcheckperf` calls to `gpssh` and `gpscp`, so these Greenplum utilities must also be in your `$PATH`.

Options

-d *test_directory*

Specifies a single directory to test on the host(s). You must have write access to the test directory on all hosts involved in the performance test. You can use the `-d` option multiple times to specify multiple test directories (for example, to test disk I/O of your primary and mirror data directories).

-f *host_file*

Specifies the name of a file that contains a list of hosts that will participate in the performance test. The host name is required, and you can optionally specify an alternate user name and/or SSH port number per host. The syntax of the host file is one host per line as follows:

```
[username@]hostname[:ssh_port]
```

If running the network performance test, all entries in the host file must be for network interfaces within the same subnet. If your segment hosts have multiple network interfaces configured on different subnets, run the network test once for each subnet. For example (a host file containing segment interface host names on subnet 1):

```
sdw1-1
sdw2-1
sdw3-1
```

-h *hostname*

Specifies a single host name that will participate in the performance test. You can use the `-h` option multiple times to specify multiple host names.

-p *ssh_port*

Specifies the default SSH port to use when logging into a remote host. If you do not specify this option, the utility defaults to port 22 (or the port specified in the host file, if one is specified).

-r ds{n|N|M}

Specifies which performance tests to run. The default is `dsn`:

- Disk I/O test (`d`)
- Stream test (`s`)
- Network performance test in sequential (`n`), parallel (`N`), or full-matrix (`M`) mode. The optional `--duration` option specifies how long (in seconds) to run the network test. To use the parallel (`N`) mode, you must run the test on an *even* number of hosts.

  If you would rather use `netperf` (www.netperf.org) instead of the Greenplum network test, you can download it and install it into `$GPHOME/bin/lib` on all Greenplum hosts (master and segments). You would then specify the optional `--netperf` option to use the `netperf` binary instead of the default `gpnetbench*` utilities.

--duration

Specifies the duration of the network test in seconds (the default), minutes (m), hours (h), or days (d). The default is 5 seconds.

--netperf

Specifies that the `netperf` binary should be used to perform the network test instead of the Greenplum network test. To use this option, you must download `netperf` from www.netperf.org and install it into `$GPHOME/bin/lib` on all Greenplum hosts (master and segments).

-B *block_size*

Specifies the block size to be used for disk I/O test. The default is 32KB, which is the same as the Greenplum Database page size. You can specify sizing in KB, MB, or GB.

-S *file_size*

Specifies the total file size to be used for the disk I/O test for all directories specified with `-d`. *file_size* should equal two times total RAM on the host. If not specified, the default is calculated at two times the total RAM on the host where `gpcheckperf` is executed. This ensures that the test is truly testing disk I/O and not using the memory cache. You can specify sizing in KB, MB, or GB.

-D (display per-host results)

Reports performance results for each host for the disk I/O tests. The default is to report results for just the hosts with the minimum and maximum performance, as well as the total and average performance of all hosts.

-v (verbose) | -V (very verbose)

Verbose mode shows progress and status messages of the performance tests as they are run. Very verbose mode shows all output messages generated by this utility.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Run the disk I/O and memory bandwidth tests on all the hosts in the file *host_file* using the test directory of */data1* and */data2*:

```
$ gpcheckperf -f host_file -d /data1 -d /data2 -r ds
```

Run only the disk I/O test on the hosts named *sdw1* and *sdw2* using the test directory of */data1*. Show individual host results and run in verbose mode:

```
$ gpcheckperf -h sdw1 -h sdw2 -d /data1 -r d -D -v
```

Run the parallel network test using the test directory of */tmp,* where *host_file_subnet** specifies all network interface host names within the same subnet:

```
$ gpcheckperf -f host_file_subnet1 -r N -d /tmp
$ gpcheckperf -f host_file_subnet2 -r N -d /tmp
```

Run the same test as above, but use `netperf` instead of the Greenplum network test (note that `netperf` must be installed in `$GPHOME/bin/lib` on all Greenplum hosts):

```
$ gpcheckperf -f host_file_subnet1 -r N --netperf -d /tmp
$ gpcheckperf -f host_file_subnet2 -r N --netperf -d /tmp
```

See Also

gpssh, gpscp, gpcheck

gpdeletesystem

Deletes a Greenplum Database system that was initialized using `gpinitsystem`.

Synopsis

gpdeletesystem -d *master_data_directory* [**-B** *parallel_processes*]
[**-f**] [**-l** *logfile_directory*] [**-D**]

gpdeletesystem -?

gpdeletesystem -v

Description

The `gpdeletesystem` utility will perform the following actions:

- Stop all `postgres` processes (the segment instances and master instance).
- Deletes all data directories.

Before running `gpdeletesystem`:

- Move any backup files out of the master and segment data directories.
- Make sure that Greenplum Database is running.
- If you are currently in a segment data directory, change directory to another location. The utility fails with an error when run from within a segment data directory.

This utility will not uninstall the Greenplum Database software.

Options

-d *data_directory*

Required. The master host data directory.

-B *parallel_processes*

The number of segments to delete in parallel. If not specified, the utility will start up to 60 parallel processes depending on how many segment instances it needs to delete.

-f (force)

Force a delete even if backup files are found in the data directories. The default is to not delete Greenplum Database instances if backup files are present.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-D (debug)

Sets logging level to debug.

-? (help)

Displays the online help.

-v (show utility version)

Displays the version, status, last updated date, and check sum of this utility.

Examples

Delete a Greenplum Database system:

```
gpdeletesystem -d /gpdata/gp-1
```

Delete a Greenplum Database system even if backup files are present:

```
gpdeletesystem -d /gpdata/gp-1 -f
```

See Also

gpinitsystem, gp_dump

gpinitstandby

Adds and/or initializes a standby master host for a Greenplum Database system.

Synopsis

```
gpinitstandby { -s standby_hostname | -r }
[-M smart | -M fast] [-n] [-a] [-q] [-l logfile_directory] [-L]
[-B parallel_processes] [-D]

gpinitstandby -? | -v
```

Description

The `gpinitstandby` utility adds a backup master host to your Greenplum Database system. If your system has an existing backup master host configured, use the `-r` option to remove it before adding the new standby master host.

Before running this utility, make sure that the Greenplum Database software is installed on the backup master host and that you have exchanged SSH keys between hosts. Also make sure that the master port is set to the same port number on the master host and the backup master host.

See the *Greenplum Database Installation Guide* for instructions. This utility should be run on the currently active *primary* master host.

The utility will perform the following steps:

- Shutdown your Greenplum Database system
- Update the Greenplum Database system catalog to remove the existing backup master host information (if the `-r` option is supplied)
- Update the Greenplum Database system catalog to add the new backup master host information (use the `-n` option to skip this step)
- Edit the `pg_hba.conf` files of the segment instances to allow access from the newly added standby master.
- Setup the backup master instance on the alternate master host
- Start the synchronization process
- Restart your Greenplum Database system

A backup master host serves as a 'warm standby' in the event of the primary master host becoming unoperational. The backup master is kept up to date by a transaction log replication process (`gpsyncagent`), which runs on the backup master host and keeps the data between the primary and backup master hosts synchronized. If the primary master fails, the log replication process is shut down, and the backup master can be activated in its place by using the utility. Upon activation of the backup master, the replicated logs are used to reconstruct the state of the master host at the time of the last successfully committed transaction.

The activated standby master effectively becomes the Greenplum Database master, accepting client connections on the master port and performing normal master operations such as SQL command processing and workload management.

Options

-s *standby_hostname*

Required. The host name of the standby master host.

-r (remove standby master)

Removes the currently configured standby master host from your Greenplum Database system.

-M fast (fast shutdown - rollback)

Use fast shut down when stopping Greenplum Database at the beginning of the standby initialization process. Any transactions in progress are interrupted and rolled back.

-M smart (smart shutdown - warn)

Use smart shut down when stopping Greenplum Database at the beginning of the standby initialization process. If there are active connections, this command fails with a warning. This is the default shutdown mode.

-n (resynchronize)

Use this option if you already have a standby master configured, and just want to resynchronize the data between the primary and backup master host. The Greenplum system catalog tables will not be updated.

-a (do not prompt)

Do not prompt the user for confirmation.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-L (leave database stopped)

Leave Greenplum Database in a stopped state after removing the warm standby master.

-B *parallel_processes*

The number of segment `pg_hba.conf` files to edit in parallel. If not specified, the utility will start up to 40 parallel processes depending on how many segment instances it needs to alter.

-D (debug)

Sets logging level to debug.

-? (help)

Displays the online help.

-v (show utility version)

Displays the version, status, last updated date, and check sum of this utility.

Examples

Add a backup master host to your Greenplum Database system and start the synchronization process:

```
gpinitstandby -s host09
```

Remove the existing backup master from your Greenplum system configuration:

```
gpinitstandby -r
```

Start an existing backup master host and synchronize the data with the primary master host - do not add a new Greenplum backup master host to the system catalog:

```
gpinitstandby -s host09 -n
```

See Also

gpinitsystem, gpaddmirrors, gpactivatestandby

gpinitsystem

Initializes a Greenplum Database system by using configuration parameters specified in the `gp_init_config` file.

Synopsis

```
gpinitsystem -c gp_init_config_file [-h seghost_address_file] [-o]
             [-B parallel_processes]
             [-p postgresql_conf_param_file]
             [-s standby_master_host]
             [--max_connections number] [--shared_buffers size]
             [--locale locale] [--lc-collate locale]
             [--lc-ctype locale] [--lc-messages locale]
             [--lc-monetary locale] [--lc-numeric locale]
             [--lc-time locale] [--su_password password]
             [-S] [-a] [-q] [-l logfile_directory] [-D]
```

```
gpinitsystem -?
```

```
gpinitsystem -v
```

Description

The `gpinitsystem` utility will create a Greenplum Database instance using the values defined in a Greenplum Database configuration file. See "Initialization Configuration File Format" on page 77 for more information about this configuration file. Before running this utility, make sure that you have installed the Greenplum Database software on all the hosts in the array.

The functionality of `gpinitsystem` is analogous to PostgreSQL's `initdb` utility, which creates the data storage directories, generates the system catalog tables and configuration files, and creates a *template* database (a database template used to create other databases).

In a Greenplum Database DBMS, each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS. The `gpinitsystem` utility takes care of initializing the Greenplum master and each segment instance, and configuring the system as a whole.

Before running `gpinitsystem`, you must set the `$GPHOME` environment variable to point to the location of your Greenplum Database server installation on the master host and exchange SSH keys between all host addresses in the array using `gpssh-exkeys`.

This utility performs the following tasks:

- Verifies that the parameters in the configuration file are correct.
- Ensures that a connection can be established to each host address. If a host address cannot be reached, the utility will exit.
- Verifies that the locale settings.

- Displays the configuration that will be used and prompts the user for confirmation.
- Initializes the master instance.
- Initializes the standby master instance (if specified).
- Initializes the primary segment instances.
- Initializes the mirror segment instances (if mirroring is configured).
- Configures the Greenplum Database system and checks for errors.
- Starts the Greenplum Database system.

Options

`-a (do not prompt)`

Do not prompt the user for confirmation.

`-B` ***`parallel_processes`***

The number of segments to create in parallel. If not specified, the utility will start up to 4 parallel processes at a time.

`-c` ***`gp_init_config_file`***

Required. The full path and filename of the configuration file, which contains all of the defined parameters to configure and initialize a new array. See "Initialization Configuration File Format" on page 77 for a description of this file.

`-D (debug)`

Sets log output level to debug.

`-h` ***`host_file`***

Optional. The full path and filename of a file that contains the host names of all the segment hosts in the array. If not specified on the command line, you can specify the host file using the `MACHINE_LIST_FILE` parameter in the `gp_init_config` file.

`--locale | -n` ***`locale`***

Sets the default locale used by Greenplum Database. If not specified, the `LC_ALL`, `LC_COLLATE`, or `LANG` environment variable of the master host determines the locale. If these are not set, the default locale is `C` (`POSIX`). A locale identifier consists of a language identifier and a region identifier, and optionally a character set encoding. For example, `sv_SE` is Swedish as spoken in Sweden, `en_US` is U.S. English, and `fr_CA` is French Canadian. If more than one character set can be useful for a locale, then the specifications look like this: `en_US.UTF-8` (locale specification and character set encoding). On most systems, the command `locale` will show the locale environment settings and `locale -a` will show a list of all available locales.

--lc-collate *locale*

Similar to `--locale`, but sets the locale used for collation (sorting data). The sort order cannot be changed after Greenplum Database is initialized, so it is important to choose a collation locale that is compatible with the character set encodings that you plan to use for your data. There is a special collation name of `C` or `POSIX` (byte-order sorting as opposed to dictionary-order sorting). The `C` collation can be used with any character encoding.

--lc-ctype *locale*

Similar to `--locale`, but sets the locale used for character classification (what character sequences are valid and how they are interpreted). This cannot be changed after Greenplum Database is initialized, so it is important to choose a character classification locale that is compatible with the data you plan to store in Greenplum Database.

--lc-messages *locale*

Similar to `--locale`, but sets the locale used for messages output by Greenplum Database. The current version of Greenplum Database does not support multiple locales for output messages (all messages are in English), so changing this setting will not have any effect.

--lc-monetary *locale*

Similar to `--locale`, but sets the locale used for formatting currency amounts.

--lc-numeric *locale*

Similar to `--locale`, but sets the locale used for formatting numbers.

--lc-time *locale*

Similar to `--locale`, but sets the locale used for formatting dates and times.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

--max_connections | -m *number*

Sets the maximum number of client connections allowed to the master. The default is 25.

-o (create master only)

Creates a master instance only. Does not configure any segment hosts.

-p *postgresql_conf_param_file*

Optional. The name of a file that contains `postgresql.conf` parameter settings that you want to set for Greenplum Database. These settings will be used when the individual master and segment instances are initialized. You can also set parameters after initialization using the `gpconfig` utility.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

--shared_buffers | -b *size*

Sets the amount of memory a Greenplum server instance uses for shared memory buffers. You can specify sizing in kilobytes (kB), megabytes (MB) or gigabytes (GB). The default is 125MB.

-s *standby_master_host*

Optional. If you wish to configure a backup master host, specify the host name using this option. You must have the Greenplum Database software installed and configured on the backup master host.

--su_password | -e *superuser_password*

The password to set for the Greenplum Database superuser. Defaults to `gparray`. You can always change the superuser password at a later time using the `ALTER ROLE` command. Client connections over the network require a password login for the database superuser account (for example, the `gpadmin` user).

-S (spread mirror configuration)

If mirroring parameters specified, spreads the mirror segments across the available hosts. The default is to group the set of mirror segments together on an alternate host from their primary segment set. Mirror spreading will place each mirror on a different host within the Greenplum Database array. Spreading is only allowed if there is a sufficient number of hosts in the array (number of hosts is greater than the number of segment instances).

-v (show utility version)

Displays the version of this utility.

-? (help)

Displays the online help.

Initialization Configuration File Format

`gpinitsystem` requires a configuration file with the following parameters defined. An example initialization configuration file can be found in `$GPHOME/docs/cli_help/gp_init_config_example`.

ARRAY_NAME

Required. A name for the array you are configuring. You can use any name you like. Enclose the name in quotes if the name contains spaces.

MACHINE_LIST_FILE

Optional. Can be used in place of the `-h` option. This specifies the file that contains the list of segment host address names that comprise the Greenplum system. The master host is assumed to be the host from which you are running the utility and should not be included in this file. If your segment hosts have multiple network interfaces, then this file would include all addresses for the host. Give the absolute path to the file.

SEG_PREFIX

Required. This specifies a prefix that will be used to name the data directories on the master and segment instances. The naming convention for data directories in a Greenplum Database system is *`SEG_PREFIXnumber`* where *`number`* starts with 0 for segment instances (the master is always -1). So for example, if you choose the prefix `gpseg`, your master instance data directory would be named `gp-1`, and the segment instances would be named `gpseg0`, `gpseg1`, `gpseg2`, `gpseg3`, and so on.

PORT_BASE

Required. This specifies the base number by which primary segment port numbers are calculated. The first primary segment port on a host is set as `PORT_BASE`, and then incremented by one for each additional primary segment on that host. Valid values range from 1 through 65535.

DATA_DIRECTORY

Required. This specifies the data storage location(s) where the utility will create the primary segment data directories. The utility creates a unique data directory for each segment instance. If you want multiple segment instances per host, list a data storage area for each primary segment you want created. The recommended number is one primary segment per CPU. It is OK to list the same data storage area multiple times if you want your data directories created in the same location. The number of data directory locations specified will determine the number of primary segment instances per host. You must make sure that the user who runs `gpinitsystem` (for example, the `gpadmin` user) has permissions to write to these directories. You may want to create these directories on the segment hosts before running `gpinitsystem` and `chown` them to the appropriate user. For example:

```
declare -a DATA_DIRECTORY=(/data1/primary /data1/primary
/data1/primary /data2/primary /data2/primary /data2/primary)
```

MASTER_HOSTNAME

Required. The host name of the master instance. This host name must exactly match the configured host name of the machine (run the `hostname` command to determine the correct hostname).

MASTER_DIRECTORY

Required. This specifies the location where the data directory will be created on the master host. You must make sure that the user who runs `gpinitsystem` (for example, the `gpadmin` user) has permissions to write to this directory. You may want to create this directory on the master host before running `gpinitsystem` and `chown` it to the appropriate user.

MASTER_PORT

Required. The port number for the master instance. This is the port number that users and client connections will use when accessing the Greenplum Database system.

TRUSTED_SHELL

Required. The shell the `gpinitsystem` utility uses to execute commands on remote hosts. Allowed values are `ssh`. You must set up your trusted host environment before running the `gpinitsystem` utility (you can use `gpssh-exkeys` to do this).

CHECK_POINT_SEGMENTS

Required. Maximum distance between automatic write ahead log (WAL) checkpoints, in log file segments (each segment is normally 16 megabytes). This will set the `checkpoint_segments` parameter in the `postgresql.conf` file for each segment instance in the Greenplum Database system.

ENCODING

Required. The character set encoding to use. This character set must be compatible with the `--locale` settings used, especially `--lc-collate` and `--lc-ctype`. Greenplum Database supports the same character sets as PostgreSQL.

DATABASE_NAME

Optional. The name of a Greenplum Database database to create after the system is initialized. You can always create a database later using the `CREATE DATABASE` command or the `createdb` utility.

MIRROR_PORT_BASE

Optional. This specifies the base number by which mirror segment port numbers are calculated. The first mirror segment port on a host is set as `MIRROR_PORT_BASE`, and then incremented by one for each additional mirror segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE`.

REPLICATION_PORT_BASE

Optional. This specifies the base number by which the port numbers for the primary file replication process are calculated. The first replication port on a host is set as `REPLICATION_PORT_BASE`, and then incremented by one for each additional primary segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE` or `MIRROR_PORT_BASE`.

MIRROR_REPLICATION_PORT_BASE

Optional. This specifies the base number by which the port numbers for the mirror file replication process are calculated. The first mirror replication port on a host is set as `MIRROR_REPLICATION_PORT_BASE`, and then incremented by one for each additional mirror segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE`, `MIRROR_PORT_BASE`, or `REPLICATION_PORT_BASE`.

MIRROR_DATA_DIRECTORY

Optional. This specifies the location where the data directory will be created on a host for a mirror segment instance. There must be the same number of data directories declared for mirror segment instances as for primary segment instances (see the `DATA_DIRECTORY` parameter). You must make sure that the user who runs `gpinitsystem` (for example, the `gpadmin` user) has permissions to write to these directories. You may want to create these directories on the segment hosts before running `gpinitsystem` and `chown` them to the appropriate user. For example:

```
declare -a MIRROR_DATA_DIRECTORY=(/data1/mirror
/data1/mirror /data1/mirror /data2/mirror /data2/mirror
/data2/mirror)
```

Examples

Initialize a Greenplum Database array by supplying a configuration file and a segment host address file:

```
$ gpinitsystem -c gp_init_config -h seg_host_addresses
```

Initialize a Greenplum Database array and set the superuser remote password:

```
$ gpinitsystem -c gp_init_config --su-password mypassword
```

Initialize a Greenplum Database array with an optional standby master host:

```
$ gpinitsystem -c gp_init_config -s host09
```

See Also

gpbuildsystem, gpdeletesystem

gpscp

Copies files between multiple hosts at once.

Synopsis

```
gpscp { -f host_file | - h hostname [-h hostname ...] }
[-J character] [-v] [[user@]hostname:]file_to_copy [...]
[[user@]hostname:]copy_to_path
```

```
gpscp -?
```

```
gpscp --version
```

Description

The `gpscp` utility allows you to copy one or more files from the specified hosts to other specified hosts in one command using SCP (secure copy). For example, you can copy a file from the Greenplum Database master host to all of the segment hosts at the same time.

To specify the hosts involved in the SCP session, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file (`-f`) is required. The `-J` option allows you to specify a single character to substitute for the *hostname* in the copy from and to destination strings. If `-J` is not specified, the default substitution character is an equal sign (`=`). For example, the following command will copy *.bashrc* from the local host to `/home/gpadmin` on all hosts named in *host_file*:

```
gpscp -f host_file .bashrc =:/home/gpadmin
```

If a user name is not specified in the host list or with *`user@`* in the file path, `gpscp` will copy files as the currently logged in user. To determine the currently logged in user, do a `whoami` command. By default, `gpscp` goes to `$HOME` of the session user on the remote hosts after login. To ensure the file is copied to the correct location on the remote hosts, it is recommended that you use absolute paths.

Before using `gpscp`, you must have a trusted host setup between the hosts involved in the SCP session. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already.

Options

-f *host_file*

Specifies the name of a file that contains a list of hosts that will participate in this SCP session. The host name is required, and you can optionally specify an alternate user name and/or SCP port number per host. The syntax of the host file is one host per line as follows:

```
[username@]hostname[:ssh_port]
```

-h *hostname*

Specifies a single host name that will participate in this SCP session. You can use the `-h` option multiple times to specify multiple host names.

-J *character*

The `-J` option allows you to specify a single character to substitute for the *hostname* in the copy from and to destination strings. If `-J` is not specified, the default substitution character is an equal sign (`=`).

-v (verbose mode)

Optional. Reports additional messages in addition to the SCP command output.

file_to_copy

Required. The file name (or absolute path) of a file that you want to copy to other hosts (or file locations). This can be either a file on the local host or on another named host.

copy_to_path

Required. The path where you want the file(s) to be copied on the named hosts. If an absolute path is not used, the file will be copied relative to `$HOME` of the session user. You can also use the equal sign '`=`' (or another character that you specify with the `-J` option) in place of a *hostname*. This will then substitute in each host name as specified in the supplied host file (`-f`) or with the `-h` option.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Copy the file named *installer.tar* to `/` on all the hosts in the file *host_file*.

```
gpscp -f host_file installer.tar =:/
```

Copy the file named *myfuncs.so* to the specified location on the hosts named *seg1* and *seg2*:

```
gpscp -h seg1 -h seg2 myfuncs.so \
=:/usr/local/greenplum-db-4.0.x.x/lib
```

See Also

gpssh-exkeys, gpssh

gpseginstall

Installs Greenplum Database on segment hosts.

Synopsis

gpseginstall -f *host_file* [**-u** *user*] [**-p** *password*]
[**-c** [**u**|**p**|**c**|**s**|**E**|**e**|**l**|**v**]]

gpseginstall --help

Description

The `gpseginstall` utility provides a simple way to quickly install Greenplum Database on segment hosts that you specify in a host list file. The utility does not install or update Greenplum Database on the master host. You can run `gpseginstall` as `root` or as a non-root user. `gpseginstall` does not perform database initialization. See `gpinitsystem` for more information about initializing Greenplum Database.

When run as `root`, `gpseginstall` default actions are to add a system user (default is `gpadmin`), create a password (default is `changeme`), and deploy and install Greenplum Database on segment hosts. To do this, `gpseginstall` locates the current Greenplum Database binaries on the master from the installation path in the current user's environment variables (`$GPHOME`). It compresses Greenplum Database software into a tar.gz file and performs an MD5 checksum to verify file integrity.

Then, it copies Greenplum Database to the segment hosts, installs (decompresses) Greenplum Database, and changes the ownership of the Greenplum Database installation to the system user you specify with the `-u` option. Lastly, it exchanges keys between all Greenplum Database hosts as both `root` and as the system user you specify with the `-u` option. `gpseginstall` also perform a user limit check and verifies the version number of Greenplum Database on all the segments.

If you run `gpseginstall` as a non-root user, `gpseginstall` only compresses, copies, and installs Greenplum Database on segment hosts. It can also exchanges keys between Greenplum Database hosts for the current system user, and verifies the version number of Greenplum Database on all the segments.

Options

-c | --commands ***command_option(s)***

This allows you to customize `gpseginstall` actions. Note that these command options are executed by default if you do not specify the `-c` option in the `gpseginstall` syntax.

- **u**: Adds a system user. (`root` only)
- **p**: Changes the password for a system user. (`root` only)
- **s**: Compresses, copies, decompresses (installs) Greenplum Database on all segments.

- **c**: Changes the ownership of the Greenplum Database installation directory on the segment hosts. (`root` only)
- **E**: Exchange keys between Greenplum Database master and segment hosts for the `root` user. (`root` only)
- **e**: Exchange keys between Greenplum Database master and segment hosts for the non-root system user.
- **l**: (Linux only) Checks and modifies the user limits configuration file (`/etc/security/limits.conf` file) when adding a new user to segment hosts. (`root` only)
- **v**: Verifies the version of Greenplum Database running on all segments. `gpseginstall` checks the version number of the Greenplum Database installation referenced by the `$GPHOME` environment variable and symbolic link to the installation directory. An error occurs if there is a version number mismatch or the Greenplum Database installation directory cannot be found.

-f | --file *host_file*

This option is required. This specifies the file that lists the segment hosts onto which you want to install Greenplum Database.

The host list file must have one host name per line and includes a host name for each segment host in your Greenplum system. Make sure there are no blank lines or extra spaces. If a host has multiple configured host names, use only *one* host name per host. For example:

```
sdw1-1
sdw2-1
sdw3-1
sdw4-1
```

If available, you can use the same `gpssh-exkeys` host list file you used to exchange keys between Greenplum Database hosts.

-p | --password *password*

This sets the password for the user you specify with the `-u` option. The default password is `changeme`. This option is only available when you run `gpseginstall` as `root`.

-u | --user *user*

This specifies the system user. This user is also the Greenplum Database administrative user. This user owns Greenplum Database installation and administers the database. This is also the user under which Greenplum Database is started/initialized. This option is only available when you run `gpseginstall` as `root`. The default is `gpadmin`.

--help (help)

Displays the online help.

Examples

As `root`, install a Greenplum Database on all segments, leave the system user as the default (`gpadmin`) and set the `gpadmin` password to `secret123`:

```
$ gpseginstall -f my_host_list_file -p secret123
```

As a non-root user, compress and copy Greenplum Database binaries to all segments (as `gpadmin`)

```
$ gpseginstall -f host_file
```

As `root`, add a user (`gpadmin2`), set the password for the user (`secret1234`), exchange keys between hosts as the new user, check user limits, and verify version numbers, but do not change ownership of Greenplum binaries, compress/copy/ install Greenplum Database on segments, or exchange keys as `root`.

```
$ gpseginstall -f host_file -u gpadmin2 -p secret1234 -c upelv
```

See Also

gpinitsystem, gpssh-exkeys

gpssh-exkeys

Exchanges SSH public keys between hosts.

Synopsis

gpssh-exkeys { **-f** *host_file* | **- h** *hostname* [**-h** *hostname* ...] | **-e** *host_file* **-x** *host_file*}

gpssh-exkeys -?

gpssh-exkeys --version

Description

The `gpssh-exkeys` utility exchanges SSH keys between the specified hosts. This allows SSH connections between hosts without a password prompt. The utility is used to initially prepare a Greenplum Database system for password-free SSH access, and also to add hosts to an existing Greenplum Database system.

To specify the hosts involved in an initial SSH key exchange, use the `-f` option to specify a file containing a list of host names (recommended), or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file is required. Note that the local host is included in the key exchange by default.

To specify new expansion hosts to be added to an existing Greenplum Database system, use the `-e` and `-x` options. The `-e` option specifies a file containing a list of existing hosts in the system that already have SSH keys. The `-x` option specifies a file containing a list of new hosts that need to participate in the SSH key exchange.

Keys are exchanged as the currently logged in user. Greenplum recommends performing the key exchange process twice: once as `root` and once as the `gpadmin` user (the user designated to own your Greenplum Database installation). The Greenplum Database management utilities require that the same non-root user be created on all hosts in the Greenplum Database system, and the utilities must be able to connect as that user to all hosts without a password prompt.

The `gpssh-exkeys` utility performs key exchange using the following steps:

- Creates an RSA identification key pair for the current user if one does not already exist. The public key of this pair is added to the current user's `authorized_keys` file.
- Updates the current user's `known_hosts` file with the host key of each host specified using the `-h`, `-f`, `-e`, and `-x` options.
- Connects to each host using `ssh` and obtains the `authorized_keys`, `known_hosts`, and `id_rsa.pub` files to set up password-free access.
- Adds keys from the `id_rsa.pub` files obtained from each host to the current user's `authorized_keys` file.
- Updates the `authorized_keys`, `known_hosts`, and `id_rsa.pub` files on all hosts with new host information (if any).

Options

-f *host_file*

Specifies the name of a file that contains a list of hosts that will participate in the SSH key exchange. Each line of the file must contain a single host name.

-h *hostname*

Specifies a single host name that will participate in the SSH key exchange. You can use the `-h` option multiple times to specify multiple host names.

-e *existing_hosts_file*

Specifies the name of a file containing a list of hosts participating in an expansion SSH key exchange. Each host is presumed an existing host previously prepared using `gpssh-exkeys`. Each line of the file must contain a single host name. Hosts specified in this file cannot be specified in the host file used with `-x`.

-x *new_hosts_file*

Specifies the name of a file containing a list of new hosts participating in an expansion SSH key exchange. Each host is considered a new host that was not previously prepared using `gpssh-exkeys`. Each line of the file must contain a single host name. Hosts specified in this file cannot be specified in the host file used with `-e`.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Exchange SSH keys between all hosts listed in the file *host_file*:

```
$ gpssh-exkeys -f host_file
```

Exchange SSH keys between the hosts *sdw1*, *sdw2*, and *sdw3*:

```
$ gpssh-exkeys -h sdw1 -h sdw2 -h sdw3
```

Exchange SSH keys between existing hosts *sdw1*, *sdw2* and *sdw3,* and new hosts *sdw4* and *sdw5* as part of a system expansion operation:

```
$ cat existing_hosts_file
   sdw1
   sdw2
   sdw3
$ cat new_hosts_file
   sdw4
   sdw5
```

```
$ gpssh-exkeys -e existing_hosts_file -x new_hosts_file
```

See Also

gpssh, gpscp

gpssh

Provides ssh access to multiple hosts at once.

Synopsis

```
gpssh { -f host_file | - h hostname [-h hostname ...] } [-v] [-e]
[bash_command]
```

```
gpssh -?
```

```
gpssh --version
```

Description

The `gpssh` utility allows you to run bash shell commands on multiple hosts at once using SSH (secure shell). You can execute a single command by specifying it on the command-line, or omit the command to enter into an interactive command-line session.

To specify the hosts involved in the SSH session, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file (`-f`) is required. Note that the current host is *not* included in the session by default — to include the local host, you must explicitly declare it in the list of hosts involved in the session.

Before using `gpssh`, you must have a trusted host setup between the hosts involved in the SSH session. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already.

If you do not specify a command on the command-line, `gpssh` will go into interactive mode. At the `gpssh` command prompt (`=>`), you can enter a command as you would in a regular bash terminal command-line, and the command will be executed on all hosts involved in the session. To end an interactive session, press `CTRL+D` on the keyboard or type `exit` or `quit`.

If a user name is not specified in the host file, `gpssh` will execute commands as the currently logged in user. To determine the currently logged in user, do a `whoami` command. By default, `gpssh` goes to `$HOME` of the session user on the remote hosts after login. To ensure commands are executed correctly on all remote hosts, you should always enter absolute paths.

Options

-f *host_file*

Specifies the name of a file that contains a list of hosts that will participate in this SSH session. The host name is required, and you can optionally specify an alternate user name and/or SSH port number per host. The syntax of the host file is one host per line as follows:

```
[username@]hostname[:ssh_port]
```

-h *hostname*

Specifies a single host name that will participate in this SSH session. You can use the `-h` option multiple times to specify multiple host names.

-v (verbose mode)

Optional. Reports additional messages in addition to the command output when running in non-interactive mode.

-e (echo)

Optional. Echoes the commands passed to each host and their resulting output while running in non-interactive mode.

bash_command

A bash shell command to execute on all hosts involved in this session (optionally enclosed in quotes). If not specified, `gpssh` will start an interactive session.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Start an interactive group SSH session with all hosts listed in the file *host_file*:

```
gpssh -f host_file
```

At the `gpssh` interactive command prompt, run a shell command on all the hosts involved in this session.

```
=> ls -a /data/p1
```

Exit an interactive session:

```
=> exit
=> quit
```

Start a non-interactive group SSH session with the hosts named *dw1* and *dw2* and pass a file containing several commands named *command_file* to `gpssh`:

```
gpssh -h dw1 -h dw2 -v -e < command_file
```

Execute single commands in non-interactive mode on hosts *dw2* and *localhost*:

```
gpssh -h dw2 -h localhost -v -e 'ls -a /dbfast1/gpdb-1'
gpssh -h dw2 -h localhost -v -e 'echo $GPHOME'
gpssh -h dw2 -h localhost -v -e 'ls -1 | wc -l'
```

See Also

gpssh-exkeys, gpscp

gpstart

Starts a Greenplum Database system.

Synopsis

gpstart [**-d** *master_data_directory*] [**-B** *parallel_processes*] [**-R**] [**-m**] [**-y**] [**-a**] [**-t** *timeout_seconds*] [**-l** *logfile_directory*] [**-v** | **-q**]

gpstart -? | **-h** | **--help**

gpstart --version

Description

The `gpstart` utility is used to start the Greenplum Database server processes. When you start a Greenplum Database system, you are actually starting several `postgres` database server listener processes at once (the master and all of the segment instances). The `gpstart` utility handles the startup of the individual instances. Each instance is started in parallel.

The first time an administrator runs `gpstart`, the utility creates a hosts cache file named `.gphostcache` in the user's home directory. Subsequently, the utility uses this list of hosts to start the system more efficiently. If new hosts are added to the system, you must manually remove this file from the `gpadmin` user's home directory. The utility will create a new hosts cache file at the next startup.

Before you can start a Greenplum Database system, you must have initialized the system using `gpinitsystem` first.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-B *parallel_processes*

The number of segments to start in parallel. If not specified, the utility will start up to 60 parallel processes depending on how many segment instances it needs to start.

-d *master_data_directory*

Optional. The master host data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m (master only)

Optional. Starts the master instance only, which may be useful for maintenance tasks. This mode only allows connections to the master in utility mode. For example:

```
PGOPTIONS='-c gp_session_role=utility' psql
```

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-R (restricted mode)

Starts Greenplum Database in restricted mode (only database superusers are allowed to connect).

-t *timeout_seconds*

Specifies a timeout in seconds to wait for a segment instance to start up. If a segment instance was shutdown abnormally (due to power failure or killing its `postgres` database listener process, for example), it may take longer to start up due to the database recovery and validation process. If not specified, the default timeout is 60 seconds.

-v (verbose output)

Displays detailed status, progress and error messages output by the utility.

-y (do not start standby master)

Optional. Do not start the standby master host. The default is to start the standby master host and synchronization process.

-? | -h | --help (help)

Displays the online help.

--version (show utility version)

Displays the version of this utility.

Examples

Start a Greenplum Database system:

```
gpstart
```

Start a Greenplum Database system in restricted mode (only allow superuser connections):

```
gpstart -R
```

Start the Greenplum master instance only and connect in utility mode:

```
gpstart -m
PGOPTIONS='-c gp_session_role=utility' psql
```

Display the online help for the `gpstart` utility:

```
gpstart -?
```

See Also

gpinitsystem, gpstop

gpstop

Stops or restarts a Greenplum Database system.

Synopsis

```
gpstop [-d master_data_directory] [-B parallel_processes]
[-M smart | fast | immediate] [-r] [-y] [-a]
[-l logfile_directory] [-v | -q]

gpstop -m [-d master_data_directory] [-y] [-l logfile_directory]
[-v | -q]

gpstop -u [-d master_data_directory] [-l logfile_directory] [-v |
-q]

gpstop --version

gpstop -? | -h | --help
```

Description

The `gpstop` utility is used to stop the database servers that comprise a Greenplum Database system. When you stop a Greenplum Database system, you are actually stopping several `postgres` database server processes at once (the master and all of the segment instances). The `gpstop` utility handles the shutdown of the individual instances. Each instance is shutdown in parallel.

By default, you are not allowed to shut down Greenplum Database if there are any client connections to the database. Use the `-M fast` option to roll back all in progress transactions and terminate any connections before shutting down. If there are any transactions in progress, the default behavior is to wait for them to commit before shutting down.

With the `-u` option, the utility uploads changes made to the master `pg_hba.conf` file or to *runtime* configuration parameters in the master `postgresql.conf` file without interruption of service. Note that any active sessions will not pickup the changes until they reconnect to the database.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-B *parallel_processes*

The number of segments to stop in parallel. If not specified, the utility will start up to 60 parallel processes depending on how many segment instances it needs to stop.

-d *master_data_directory*

Optional. The master host data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m (master only)

Optional. Shuts down a Greenplum master instance that was started in maintenance mode.

-M fast (fast shutdown - rollback)

Fast shut down. Any transactions in progress are interrupted and rolled back.

-M immediate (immediate shutdown - abort)

Immediate shut down. Any transactions in progress are aborted. This shutdown mode is not recommended. This mode kills all `postgres` processes without allowing the database server to complete transaction processing or clean up any temporary or in-process work files.

-M smart (smart shutdown - warn)

Smart shut down. If there are active connections, this command fails with a warning. This is the default shutdown mode.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-r (restart)

Restart after shutdown is complete.

-u (reload pg_hba.conf and postgresql.conf files only)

This option reloads the `pg_hba.conf` files of the master and segments and the runtime parameters of the `postgresql.conf` files but does not shutdown the Greenplum Database array. Use this option to make new configuration settings active after editing `postgresql.conf` or `pg_hba.conf`. Note that this only applies to configuration parameters that are designated as *runtime* parameters.

-v (verbose output)

Displays detailed status, progress and error messages output by the utility.

--version (show utility version)

Displays the version of this utility.

-y (do not stop standby master)

Do not stop the standby master process. The default is to stop the standby master.

-? | -h | --help (help)

Displays the online help.

Examples

Stop a Greenplum Database system in smart mode:

```
gpstop
```

Stop a Greenplum Database system in fast mode:

```
gpstop -M fast
```

Stop all segment instances and then restart the system:

```
gpstop -r
```

Stop a master instance that was started in maintenance mode:

```
gpstop -m
```

Reload the `postgresql.conf` and `pg_hba.conf` files after making configuration changes but do not shutdown the Greenplum Database array:

```
gpstop -u
```

See Also

gpstart

Index