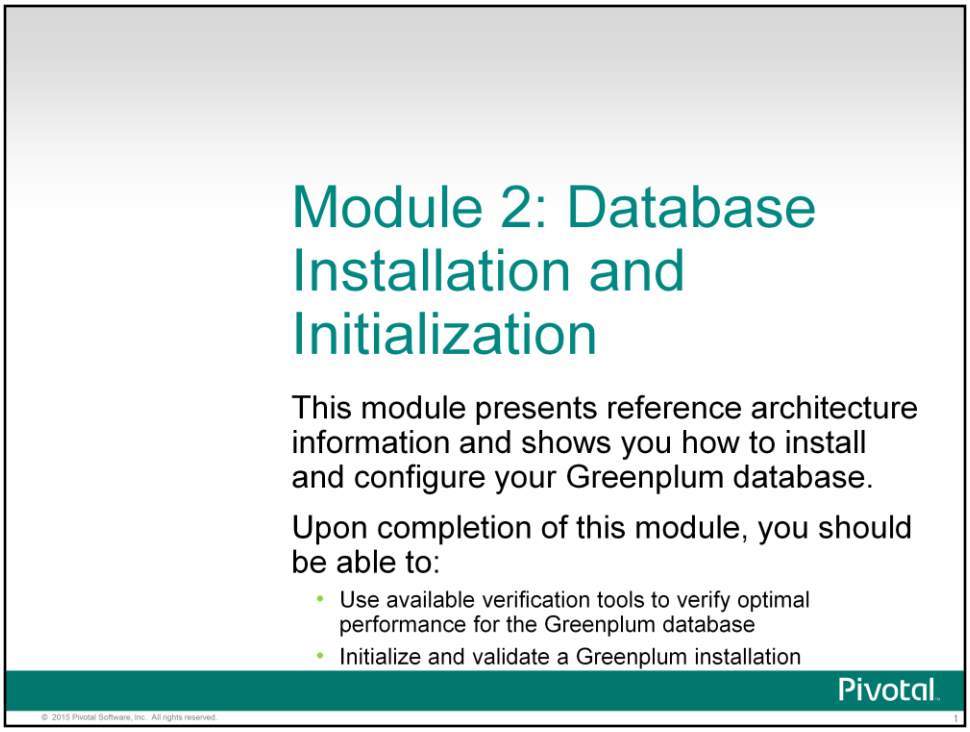
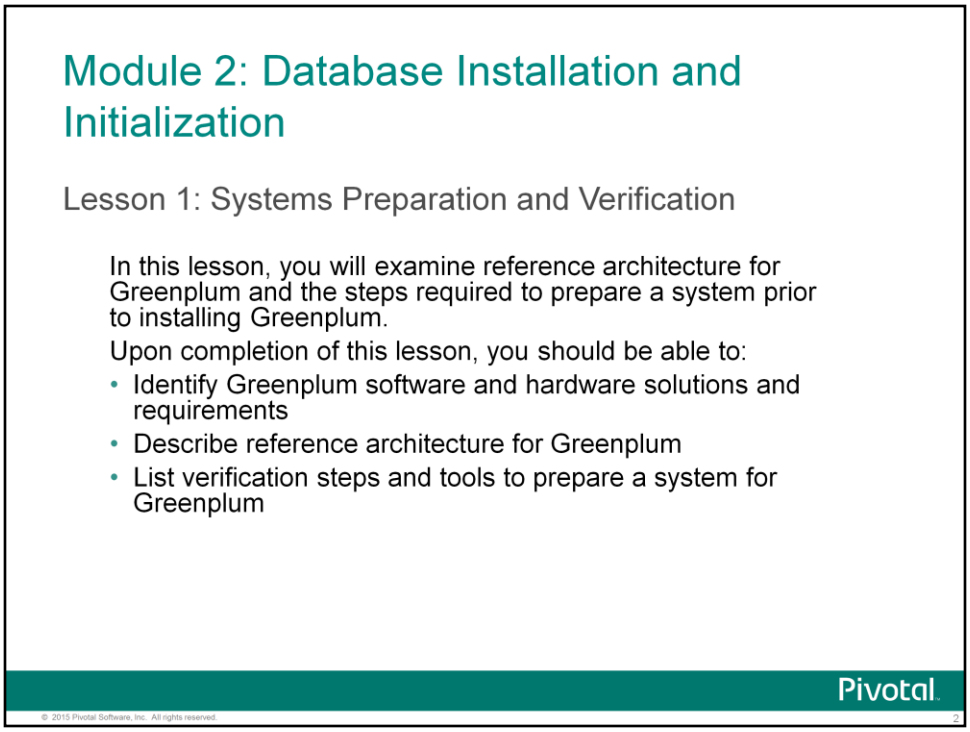
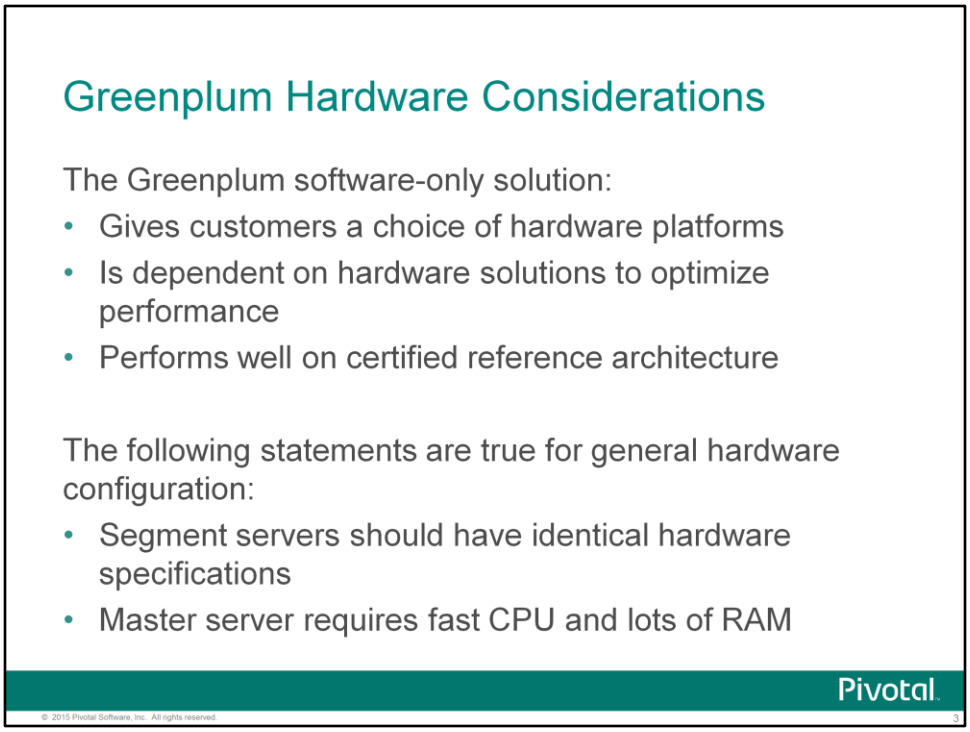


In this module, you install and initialize the Greenplum Database and Greenplum Database instance. You will configure the database and begin the process of loading data.

You will install Greenplum and use the available verification tools to ensure that the environment has been properly configured to optimize data movement, storage, and retrieval. Once verification is completed, a Greenplum Database instance can be installed and initialized to prepare it for use.



In this lesson, you will examine the supported operating systems and EMC hardware available for your Greenplum environment. You will also examine reference architecture for Greenplum, and list the steps and tools you can use to prepare a system for Greenplum installation and initialization.



Greenplum Database is a software only solution, meaning that can theoretically run on any hardware platform. The hardware and software are not coupled as in appliance vendors like Netezza and Terradata. However, as with any database, Greenplum's performance is dependent on the hardware on which it is installed. As the database is distributed across multiple machines in a Greenplum Database system, the selection and configuration of hardware is even more important to achieve the best performance possible from your database. Using SSD disk drives, for example, can greatly improve the performance of the database from the recommended specifications provided for Greenplum DB.

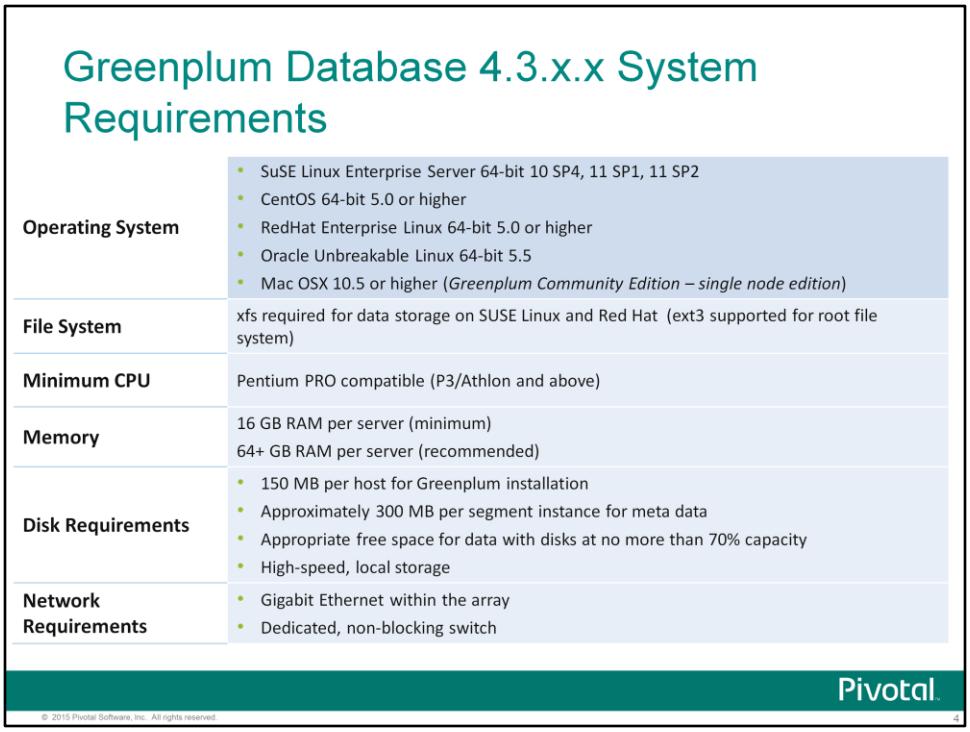
Greenplum provides reference architecture that you can use in building a white-box solution.

In general, when configuring the hardware environment for your Greenplum software-only solution, remember that:

- Segment servers should have identical hardware specifications. As they work in parallel, it is important that the components of the architecture are the same to provide optimal performance and throughput.
- The master server needs fast CPU and lots of RAM to handle incoming queries, create the query plans, and distribute these out to the segment servers.

We will examine the recommended hardware configurations for:

- The segment hosts, which store the user data and do the majority of the data processing.
- The master host, which handles user connections.
- Interconnect, which is the data transport layer between the hosts in the array.



The minimum recommended system specifications for installing Greenplum Database 4.3 are as shown on the slide. Greenplum is supported on major UNIX-based systems.

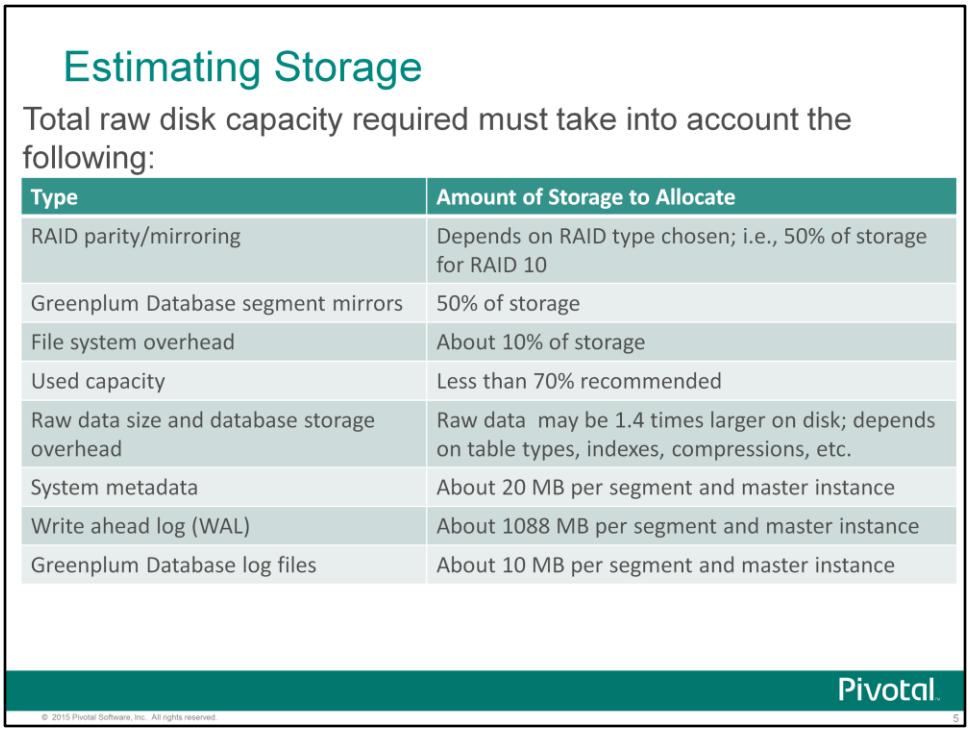
For each server in the array, the minimum memory is 16 GB of RAM. 64 GB of RAM or more is recommended for all servers. The master and standby server may require additional memory if other tools or databases are included, including GPTEXT and Pivotal HD.

Disk requirements vary based on your needs, but a minimum 150MBytes for the installation and 300MBytes per segment for meta data is required. The disk capacity should be no more than 70% to maintain optimal performance. We will discuss estimating disk capacity for user data later in the course.

EMC provides a free Greenplum Database software called Greenplum Database Community Edition to be used in single node test environments. This single node edition shares much of the same requirements as the production software. It differs in the following areas:

- It can be installed on Mac OSX 10.5 or higher. This platform is not supported for production installation of Greenplum Database.
- Minimum memory requirements are 1GB. However, for optimal performance, 4GB per CPU core is recommended.
- It does not share the same networking requirements as the multi-node production database software.
- `xfs` is recommended for optimal performance, but is not required. Either `xfs` or `ext3` can be used for file system support.

Note: Starting with Greenplum Database 4.3.0.0, Solaris is no longer a supported operating system.



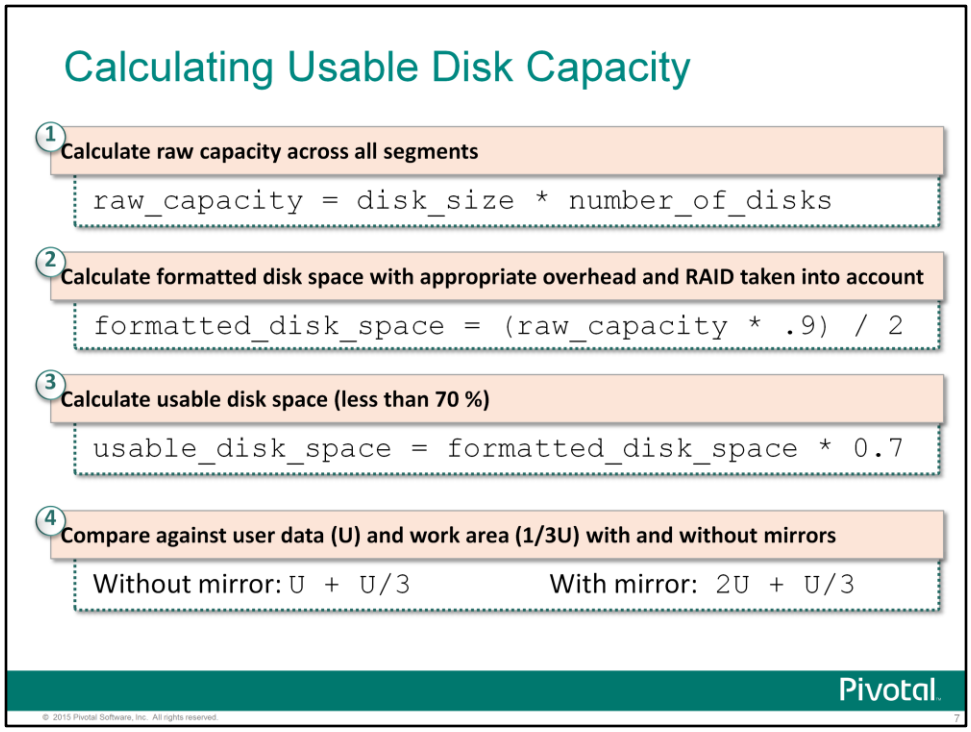
To estimate your storage needs, you need to calculate the raw disk capacity available for data storage on all of your segment hosts and consider the following:

- **RAID Parity and Mirroring** – Your overall disk space capacity will be reduced according to the RAID level you choose. For example, if using RAID 5, subtract one disk drive's worth of storage capacity per each RAID 5 set. If using RAID 10, cut your total disk drives' storage capacity in half.
- **Greenplum Database Mirroring** – Doubles the storage requirements of your user data on segment hosts.
- **Less than 70% Capacity** – For optimal performance, disks should not be more than 70% full. Performance drops off precipitously after 70%. If you do start hitting 70%, you should look at options including extending the appliance or hardware.
- **File system overhead** – The file system overhead can be anywhere from 5-15% (checksumming, file permissions, etc) depending on the file system you are using.
- **Raw Data Size and DB overhead** – 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, size of the rows versus page size, whether you create indexes, etc. The installation guide has detailed sizing info if you want to calculate DB overhead for your data more accurately.

Estimating Storage (continued)

- **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** – Greenplum Command Center 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 Greenplum Command Center agent processes on these hosts is minimal and should not significantly impact database performance. Historical data collected by Greenplum Command Center monitoring 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.

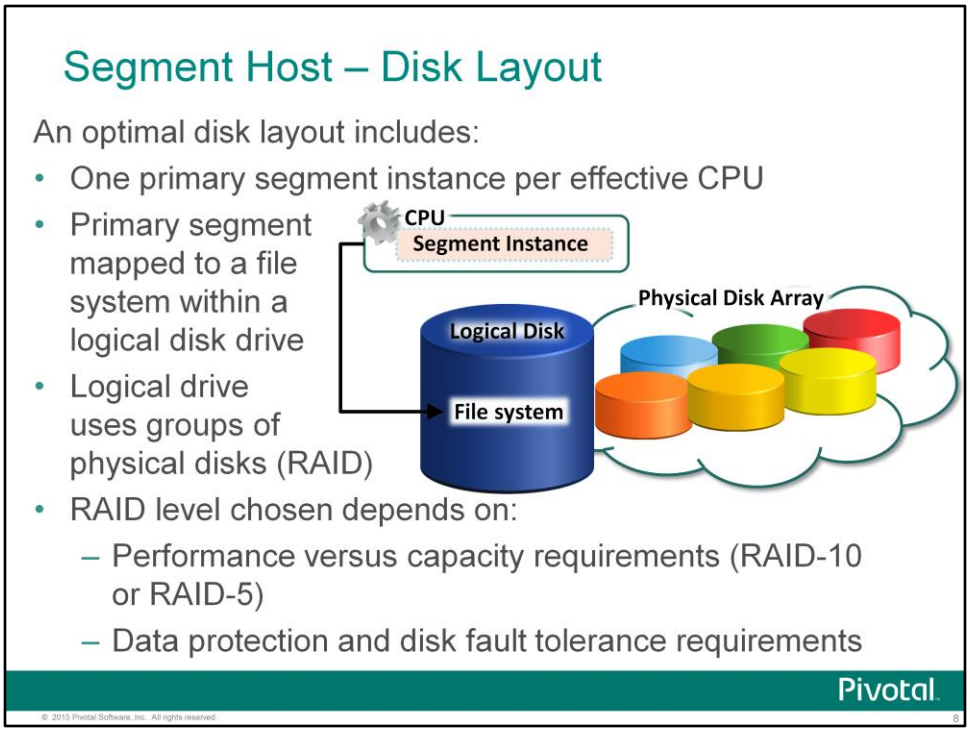
The next slide shows the calculations used for determining the disk space requirements. You will still need to account for the user data size, based on the type of data, storage, and compression methods used to name a few. You will also still need to account for system logs and metadata.



Taking into account all of the afore mentioned items you must consider when calculating required storage size, here are some measurements that will guide you in calculating how much storage you will require for your environment.

Usable disk space is calculated first for each segment host, taking into account RAID configurations, and finally mirroring. To do this:

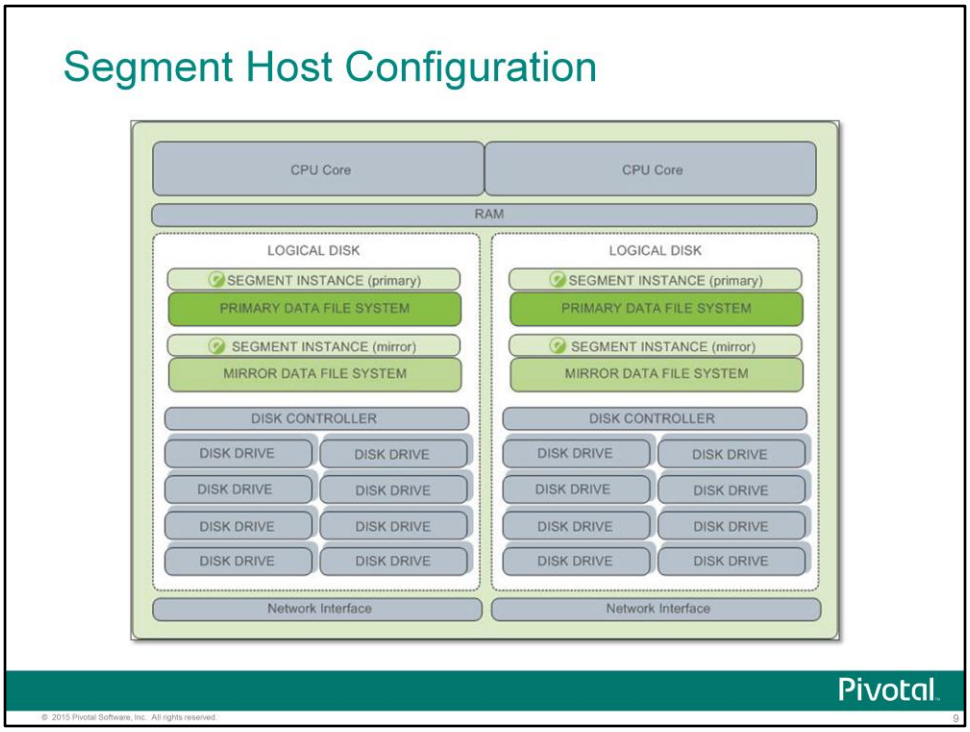
- First, calculate the raw capacity for the entire system by multiplying the number of available disks across the entire cluster by the disk size. The calculation is
  `raw_capacity = disk_size * number_of_disks`
- Next, account for both RAID and any file system overhead, which may on average be 10%. The formatted disk space would be calculated as follows if RAID 10 is being used with 10% file system overhead:
  `formatted_disk_space = (raw_capacity * .9)/2`
  Remember, RAID 10 consumes half of the available disk drive. Use the appropriate calculation for your implementation of RAID.
- Ensure you are not using more than 70% of the disk drive space to maintain optimal disk drive performance:
  `usable_disk_space = formatted_disk_space * .70`
- Once you have calculated the usable disk space, examine the user data size, represented by `U`, to ensure it corresponds to the usable disk space calculated here. If you are using mirrors, you will require twice the disk space (`U * 2`). Additionally, a work area for active queries needs to be set aside. The work area should be 1/3 the size of the user data size, calculated as `U/3`.



An optimal disk layout for Greenplum Database maps a single primary Segment Instance to a filesystem running within a logical disk drive. The logical disk and filesystem are provided by the operating system, and can be chosen from many available options. Most operating systems provide the ability for a logical disk drive to use groups of physical disks arranged in RAID arrays.

The RAID level you choose should be chosen based on:

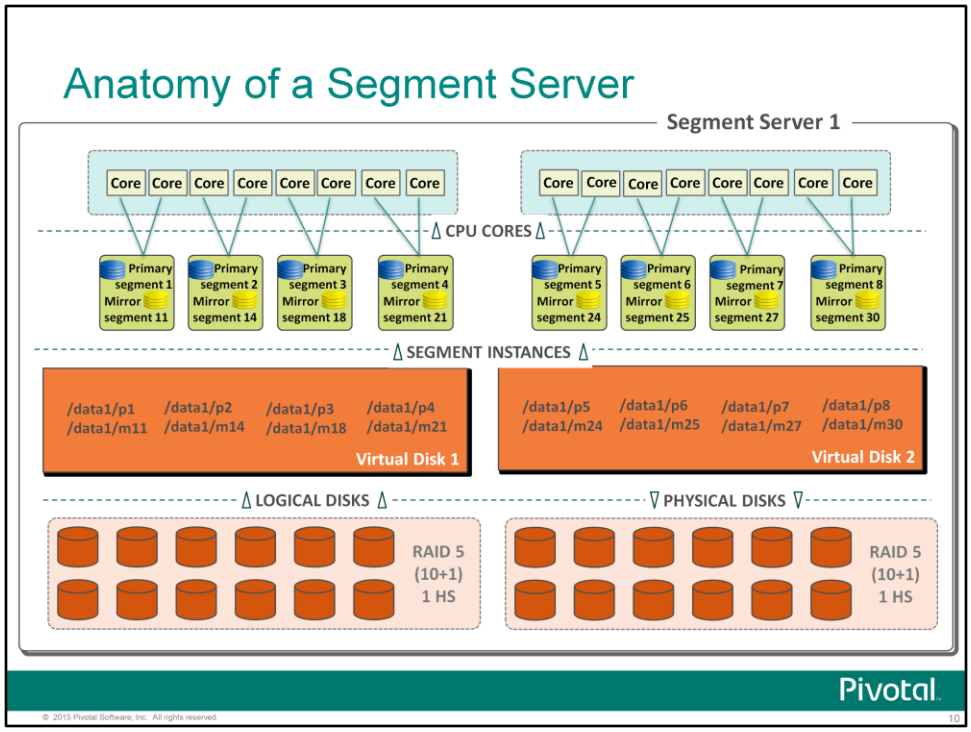
- **Performance over capacity** – For example, a RAID 10 (mirrored) disk configuration offers the best performance while a RAID 5 or RAID 1 configuration has about 30% slower performance but more capacity.
- **Data protection** – RAID 10 reserves half of the disks for data parity while RAID 5 reserves one disk for data parity.



This is an example of the hardware stack configuration of a segment host:

- One PRIMARY Greenplum Segment Instance per CPU core (or primary/mirror pair)
- A minimum of 64GB for a production environment to 256GB of RAM, depending on additional tool usage or production requirements
- One logical disk drive per CPU with 2 file systems – one for primary segments and one for mirror segments
- Hardware or software RAID disk controllers (one per set of drives)
- A number of disk drives divided evenly across primary segments
- One 10 GigE Network Interface per primary segment instance where each NIC is on its own subnet to balance Interconnect traffic

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.

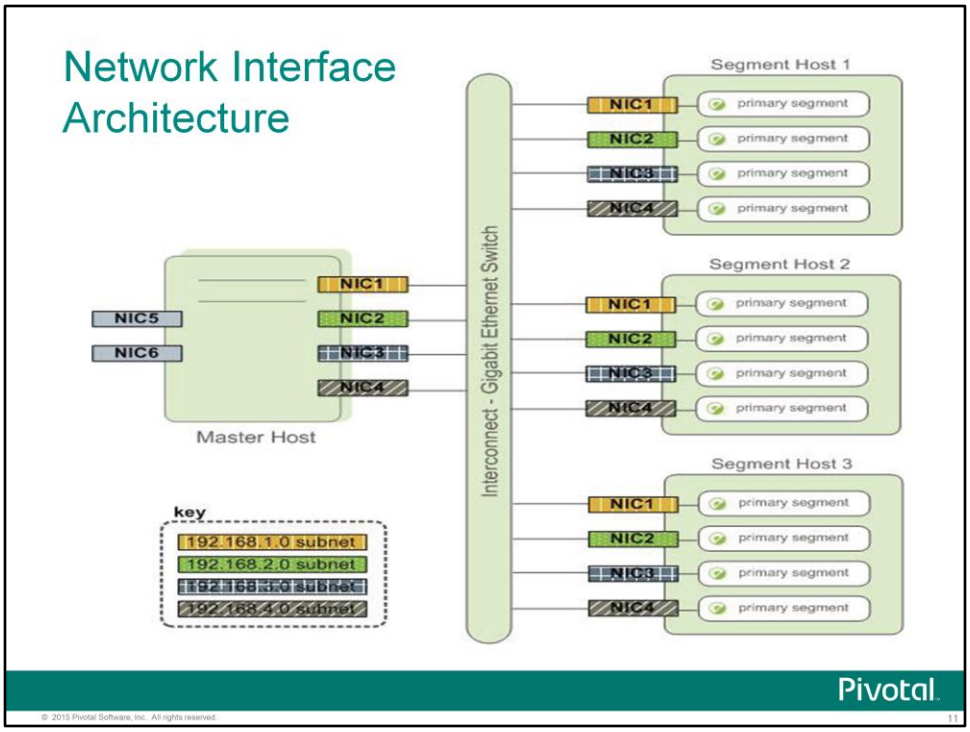


The illustration on the slide shows the segment server configuration in the Pivotal DCA. Each primary segment instance and mirror segment instance pair has the use of two CPU cores. The minimum that can be configured is a primary-mirror pair to a single CPU core. If you are using compression in your tables, increasing the number of cores per pair gives you better CPU cycles for compression and decompression.

The mirrors shown in this illustration will pair with a primary segment on a different segment host or node. For example, primary segment instance 1 pairs with mirror segment instance 24. By keeping the mirrors on a separate node, should the primary segment become unavailable on one node, the mirror segment from another node will activate.

Each server has two CPU processors with 6 CPU cores each for a total of 12 available CPU cores. Each segment server also supports 12 disk drives where the drives are divided into two groups of 6 disk drives with RAID 5 (5+1). Each group has two volumes with the following configuration:

- Volume disk 1 in the first group contains the `/root` partition.
- Volume disk 2 in the first group contains the `/data1` partition.
- Volume disk 1 in the second group contains the `/swap` partition.
- Volume disk 2 in the second group contains the `/data2` partition.



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.

Pivotal DCA – Pivotal Master and Segment Node

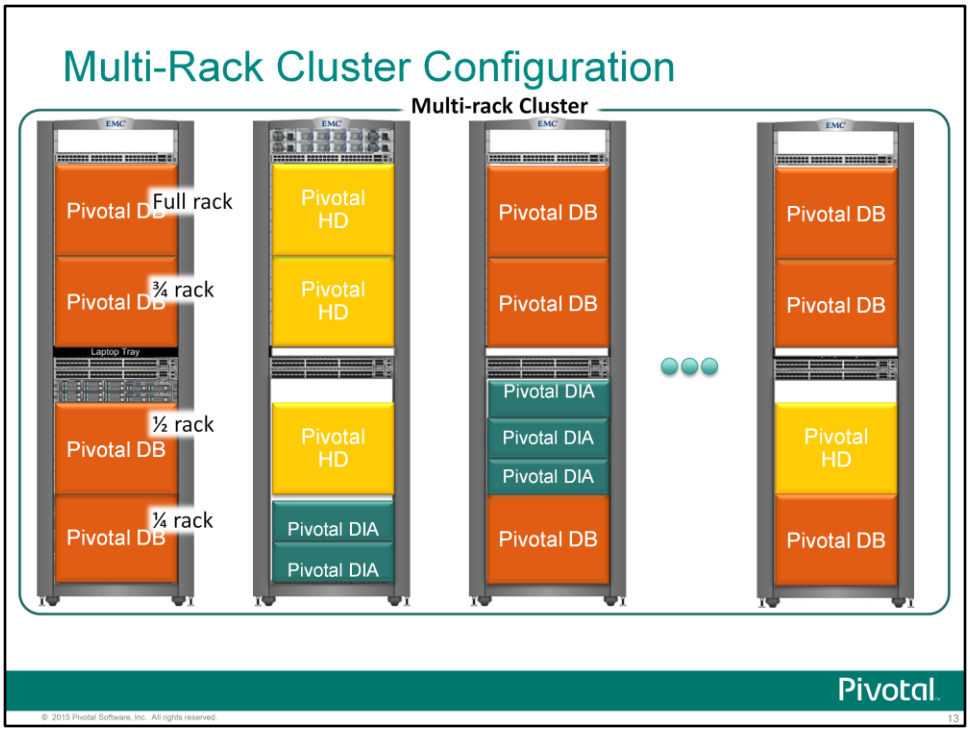
	Pivotal Compute	Pivotal Hi-Memory	Pivotal Standard
System	Intel (16 CPU Cores)		
Memory	64 GByte	256 GByte	64 GByte
Disk drives	• 1 x Dual channel 6GB/s SAS internal RAID card • 6 x 300 GByte SAS (master/standby)		
	24 x 300 GByte SAS 10K (segment servers)	24 x 300 GByte SAS 10K (segment servers)	24 x 900 GByte SAS 10K (segment servers)
Network interfaces	• 4 x 1Gb/s network interfaces (master/standby) • 2 x 1Gb/s network interfaces (segments)		
Administrative interface card	1 x Intel BMC		
Operating system	Redhat Enterprise Linux distribution x86 32/64-bit license		
Network adapter	2 x dual-port 10Gb Converged Network Adapter (CNA)		

Pivotal

12

Now that you have examined the hardware requirements and overall architecture should you decide to build your own hardware solution, we will examine the solutions provided by EMC and Pivotal.

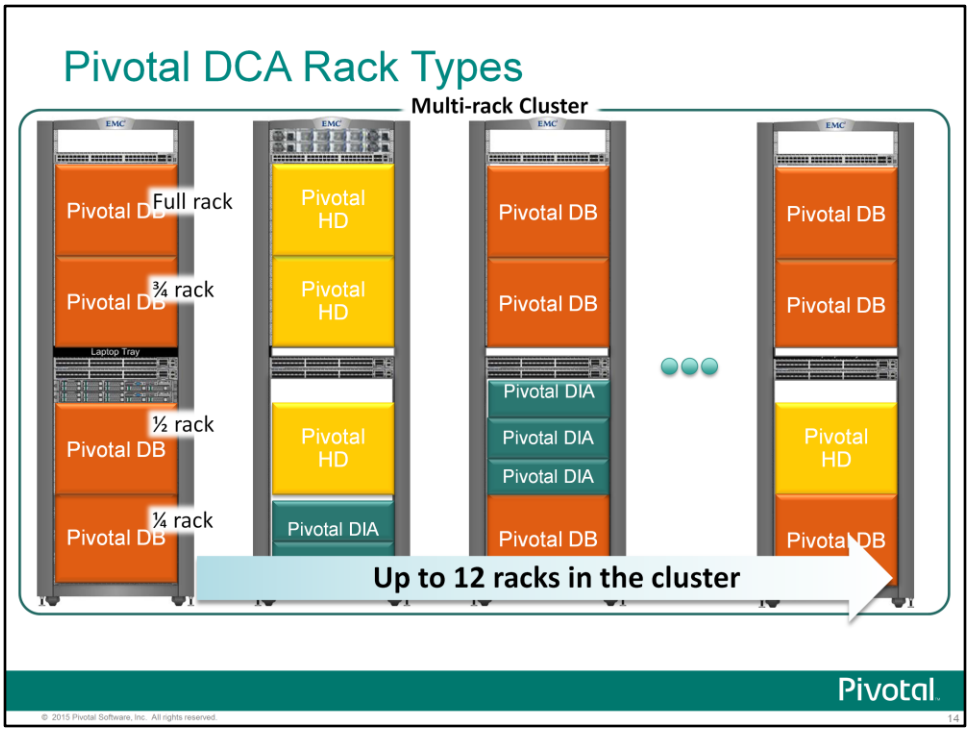
The slide highlights the hardware specifications for the Pivotal DCA. The Pivotal DCA uses Intel servers for all servers within a module. Disk capacity and memory may vary depending on the module type.



The Pivotal DCA is a self-contained data warehouse solution that integrates all the database software, servers and switches necessary to perform enterprise-scale data analytics workloads. The Pivotal DCA is delivered racked and ready for immediate data loading and query execution.

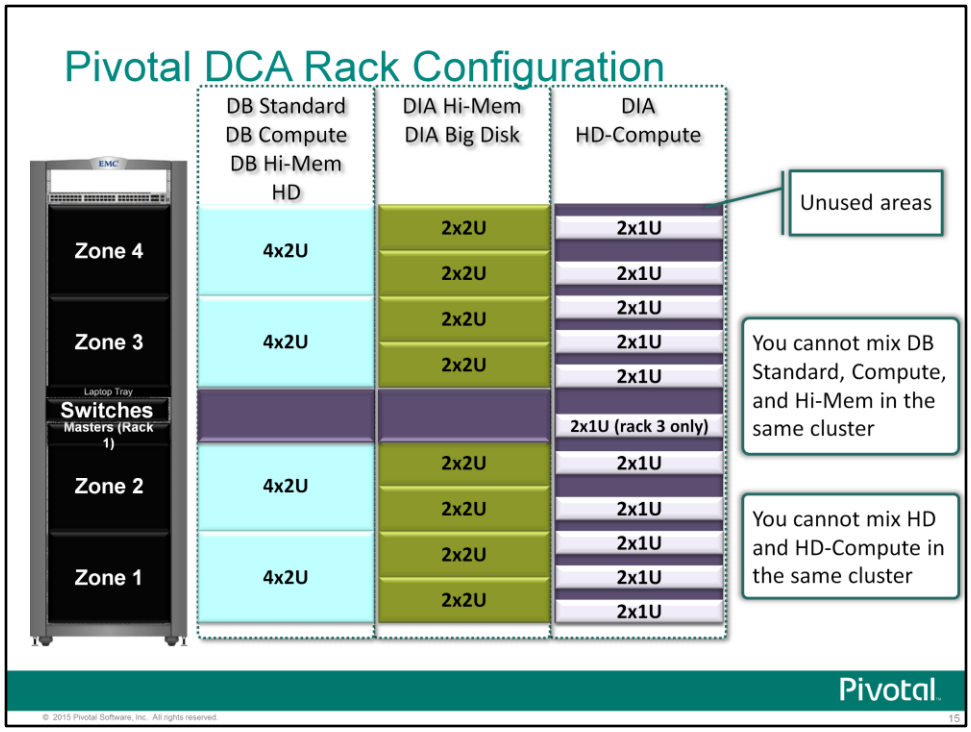
The Pivotal DCA is offered in increments of four servers, starting with a quarter-rack configuration to a multi-rack configuration. The base architecture of DCA is designed with scalability and growth in mind. This enables organizations to easily extend their DW/BI capability in a modular way; linear gains in capacity and performance are achieved by expanding a DCA system to take advantage of additional processing power and memory.

The cluster can be populated with Pivotal DB standard or compute modules, but not both. Within the cluster, you can also support up to 11 DIA modules, and Pivotal HD or Pivotal HD-compute modules.



In a multi-rack configuration, you will find:

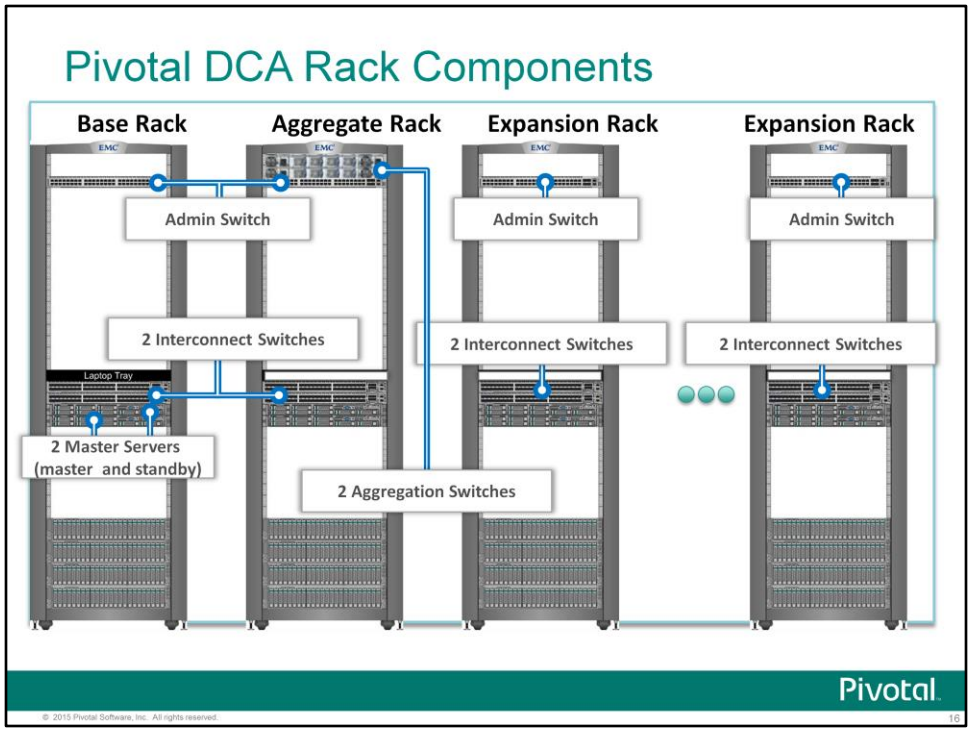
- The base or standard rack which contains the master servers for the entire cluster.
- An aggregate rack which is used to provide inter-rack communication for all of the racks of the cluster. This rack contains two additional racks to accommodate communication among all racks within the cluster.
- One or more expansion racks, depending on the number of modules you have installed, which will expand the cluster up to, and including, twelve racks. This is dependent on the number of ports on the aggregate switch supplied in the cluster.



The Pivotal racks provide up to 40 rack units of usable space and supports a variety of configurations. Each zone, as defined supported by rack power, is eight rack units high.

High memory modules reduce the number of servers that can be placed in any one rack, and in the end, in the total cluster overall. You can mix database modules with DIA modules and HD modules. You cannot combine multiple types of database modules or multiple types of HD modules in the cluster however – you must choose one of those modules.

High memory modules impose limits on the number of modules that can be installed within a rack. Zone 4 cannot be populated if you are using high memory modules.



The Pivotal DCA for Greenplum DB is shipped with:

- Greenplum Database software.
- Two master servers in the base, where one acts as the primary master server and the other acts as the hot standby master server.
- Segment Servers to contain the data and handle the bulk of the query processing. The number of segment servers varies based on the number of Greenplum Database modules installed. At a minimum, each segment server has six primary segments and six mirror segments for a total of 12 segments. A fully populated rack with four modules, each with four segment servers, will display 192 segments.
- High-speed dual interconnect switches to handle requests from the master to the segments, between segments, and to provide high-speed access to the segment servers for data loading.
- An admin switch to provide access to an administration network that is used to manage the DCA system components. The admin switch consists of one 48-port 1 GB Ethernet Layer 3 switch. It is used to connect network-enabled management ports from devices in the cluster. This switch can be cross-connected to a customer switch to provide console access to the cluster from a customer network.
- Aggregate switches in the aggregate rack to support communication between all segment servers in all of the racks of a multi-rack cluster.
- Note: Admin switches are the Arista 7048T switch on the V2 DCAs
- Note: Interconnect and aggregate switches are the Arista 7050S on the V2 DCA

Sizing a Pivotal DCA for Greenplum Database – Performance and Capacity Considerations

	DB Standard Module		DB Compute/High-Memory Module	
	Quarter-Rack	Full Rack	Quarter-Rack	Full Rack
Master Servers	2		2	
Segment Servers	4	16	4	16
Total CPU Core	64	256	64	256
Total Memory	256 GB	1024 GB	256 GB / 1024 GB	1024 GB / 4096 GB
Segment HDDs	96	384	96	384
Usable Capacity Uncompressed	27.5 TB	110 TB	9 TB	36 TB
Usable Capacity (compressed)	110 TB	440 TB	36 TB	144 TB
Scan Rate	10 GB/Sec	40 GB/Sec	10 GB/Sec	40 GB/Sec
Data Load Rate	4 TB/Hour	16 TB/Hour	4 TB/Hour	16 TB/Hour

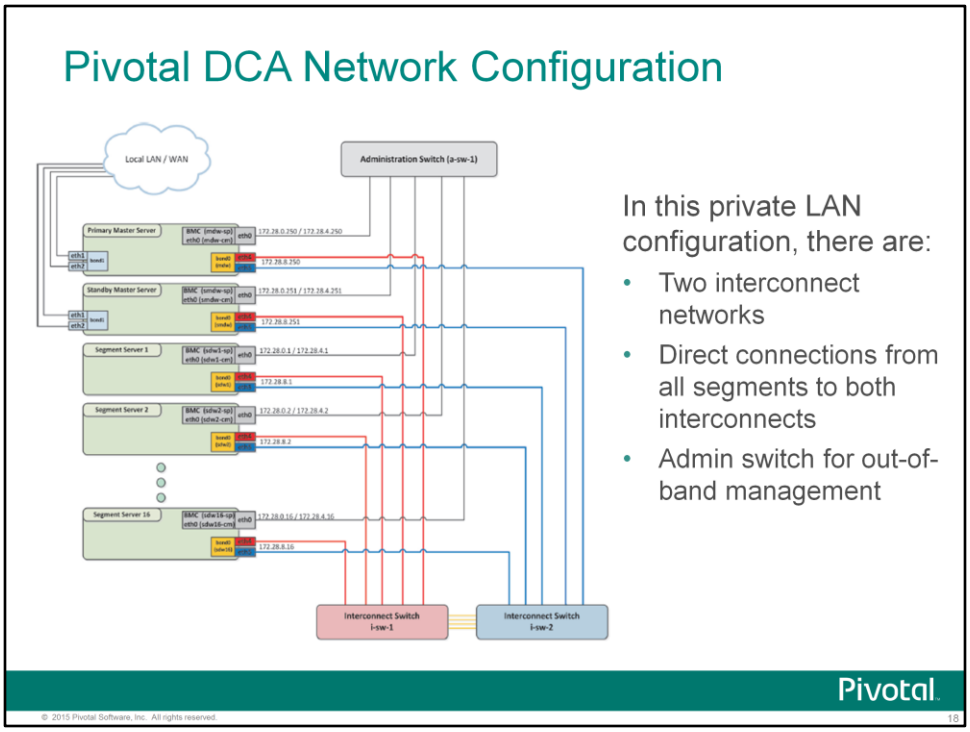
Pivotal

17

The tables on this and the next slide highlight the configurations for the standard DCA and high capacity DCA. Your needs will dictate the configuration that is right for your DW/BI needs.

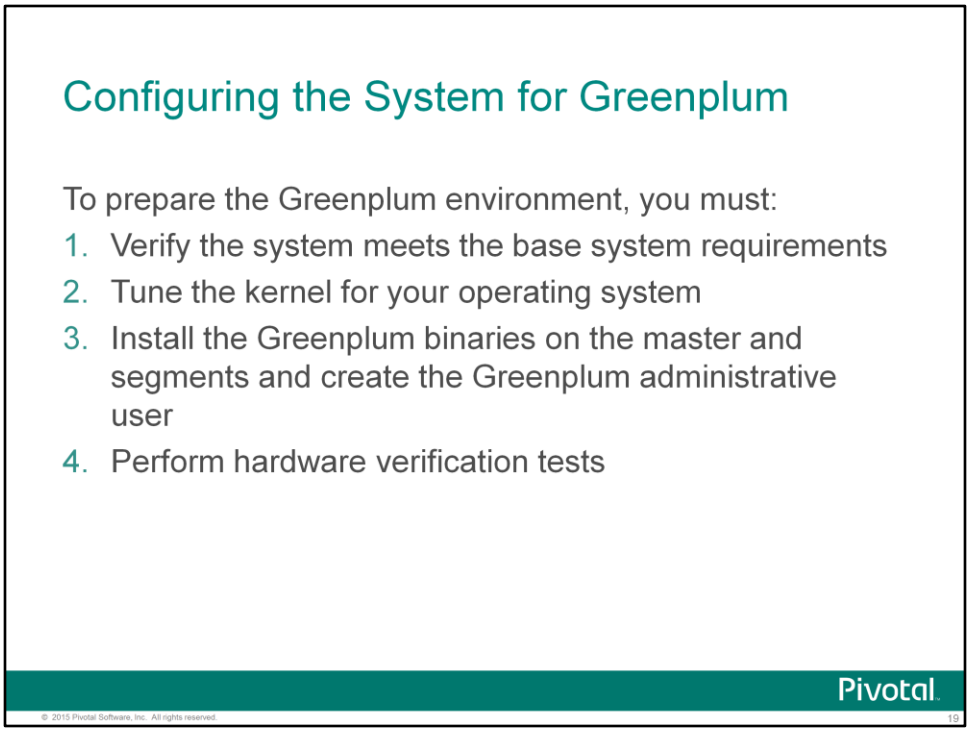
The Pivotal Database Standard modules provide additional support for data-intensive applications by offering large disk drives at 27.5 terabytes for a quarter-rack configuration. The Pivotal Database Compute modules offer a lower-price point where the larger disk size is not required.

Note that the usable capacity on disks is based on RAID-5 configuration.



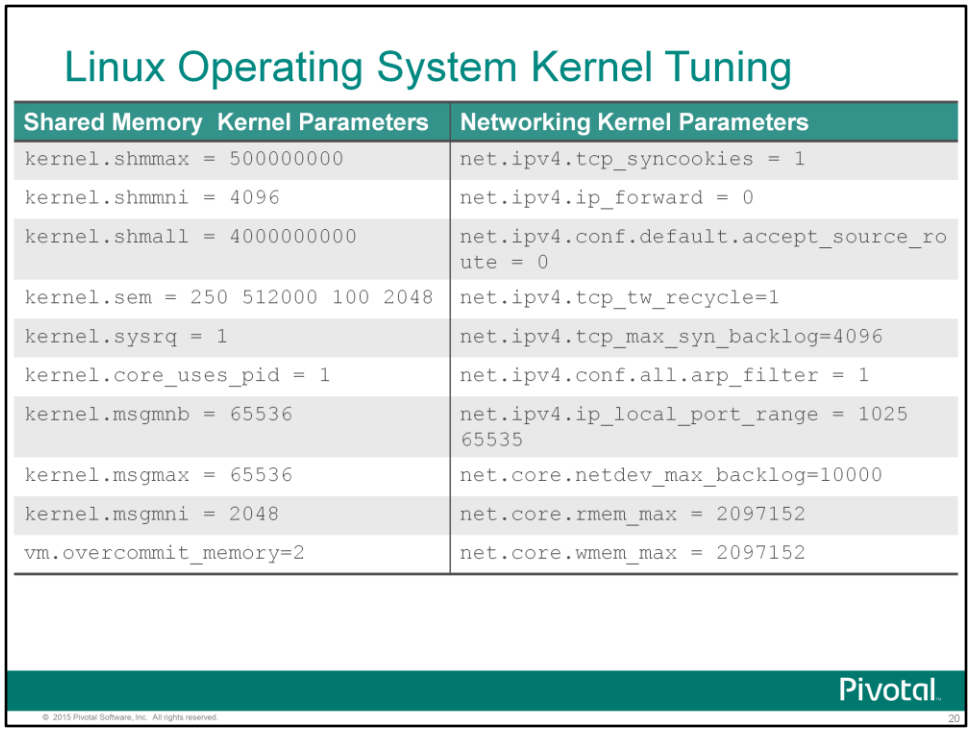
The diagram shows an example of how the network is configured in a fully-populated Pivotal DCA base rack. The Greenplum Database interconnect and administration networks are configured on a private LAN. Outside access to Greenplum Database and to the DCA systems goes through the master host.

To maximize throughput, interconnect activity is load-balanced over two interconnect networks. To ensure redundancy, a primary segment and its corresponding mirror segment utilize different interconnect networks. With this configuration, Greenplum Database can continue its operations in the event of a single interconnect switch failure.



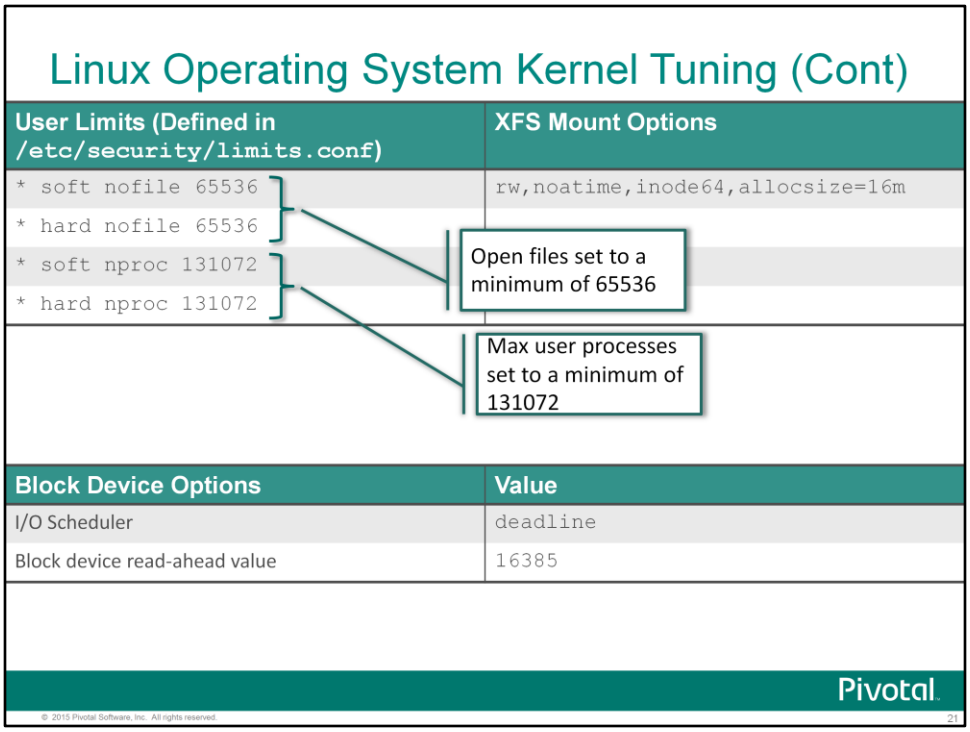
Once you have determined the hardware needs for your Greenplum environment, you will continue configuring the environment for the Greenplum database. Specifically, you will:

1. Verify that the system meets the base requirements defined.
2. Tune the kernel parameters for the operating system you will be using in the Greenplum environment.
3. Run the Greenplum software installer to install the binaries on the master and segments. You will also need to create a Greenplum administrative user to handle the database initialization and configuration.
4. Validate the system configuration by performing hardware stress and verification tests.



The kernel must be tuned before proceeding with the Greenplum installation. If you are installing in a single virtual environment or physical machine, you will only need to configure a subset of the parameters shown. Otherwise, the following must be configured:

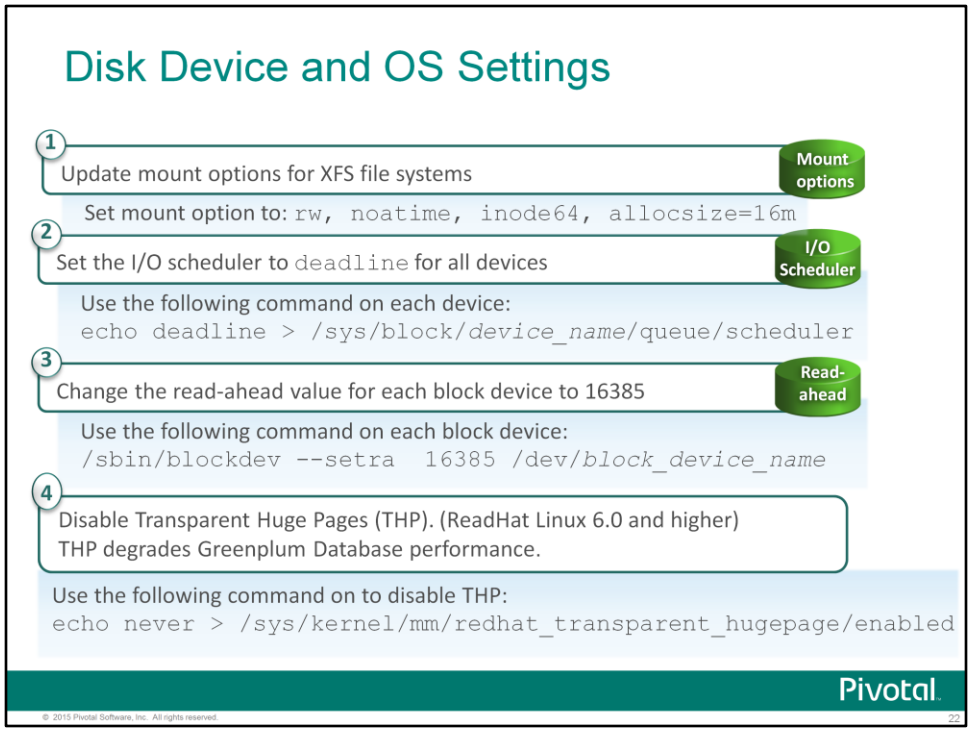
- **Shared Memory** – A segment or master instance (or any PostgreSQL database) will not start unless the shared memory segment for your kernel is properly sized. Most default Linux installations have the shared memory value set too low. Add the following lines to the `/etc/sysctl.conf` file of the master and all segment hosts.
- **Disable OOM killer** – OOM (out of memory) killer exists because the Linux kernel, by default, can commit to supplying more memory than it can actually provide. When the size of the data to be copied exceeds the size of physical memory, OOM killer randomly begins killing processes in order to free memory, often with bad results to any running Greenplum Database queries. In Greenplum Database, a single SQL statement creates several processes on the segment hosts to handle the query processing. Large queries will often trigger the OOM killer (if it is enabled) causing the query to fail. By changing the `vm.overcommit_memory` to `2`, you disable the OOM killer.



- **Network** – The settings in `/etc/sysctl.conf` prevent Interconnect errors on large volume Greenplum systems. They allow for more efficient connection handling and larger volumes of connections required for distributing and executing extremely large query plans.
- **User Limits** – User limits control the resources available to processes started by a user's shell. Use `ulimit`, a system command to see the user limits for the currently logged in user. For Greenplum Database, each OS user that issues SQL statements needs to have their limits for open file descriptors and maximum processes increased.
- **Disk and mount options** – As with other databases, the Greenplum Database is expecting certain behaviors from the block devices and file systems. The xfs file system should be mounted with the options, rw (read and write access), noatime (do not update the file access times on the device), inode64 (use 64-bits for inodes), and allocsize=16m (increase the preallocation size to accommodate large files).
  The I/O scheduler and read-ahead values for block devices must also be updated.

Reboot after making changes to the kernel.

Refer to the Greenplum Database Installation Guide for system requirements for Solaris and MacOS-based systems.



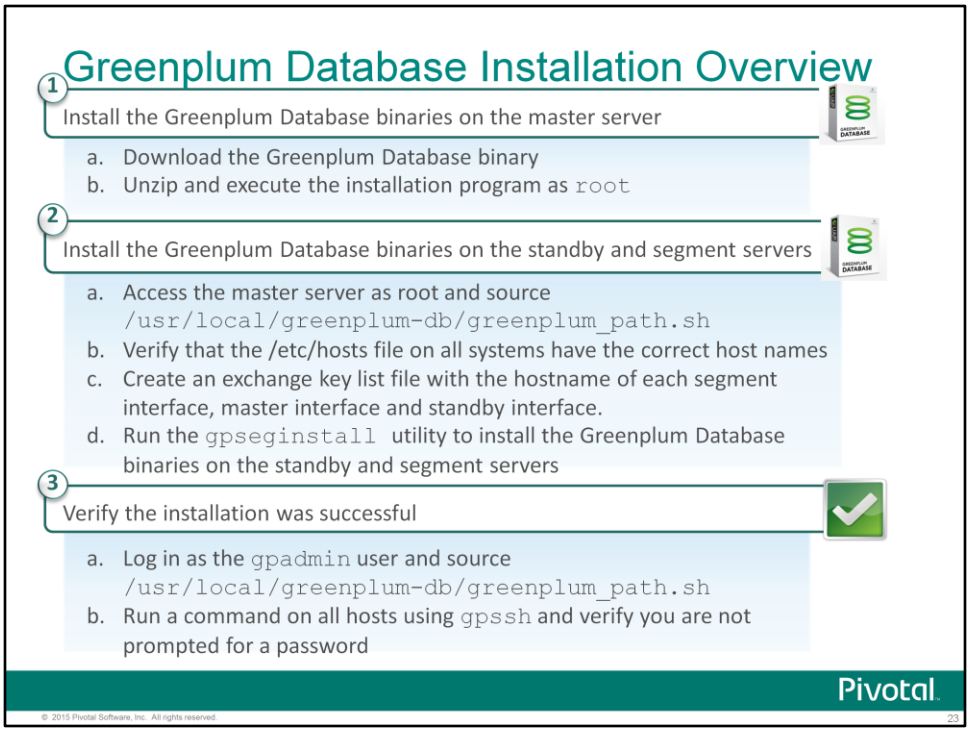
In addition to ensuring kernel parameters are configured with appropriate values, you should also update the file system mount parameters, disk access policies, and disk read-ahead values for Greenplum to improve performance overall for reading and writing to disks.

If you are using the XFS file system for data storage, update the mount options for the file system to include the following: `rw, noatime, inode64, allocsize=16m`. This sets the file system with read and write access, no access time updates, use 64 bits when allocating inodes and data, and increase the preallocation size to 16 MB to ensure large files have some continuous space set aside for writes.

For the I/O scheduler, Greenplum, as do other databases, recommends using deadline for the I/O scheduler. This improves performance and allows all I/O requests to be processed within a specified period of time. This must be done for each block device using the following command: `echo deadline > /sys/block/`*`device_name`*`/queue/scheduler`, where the *`device_name`* could be `sdb`.

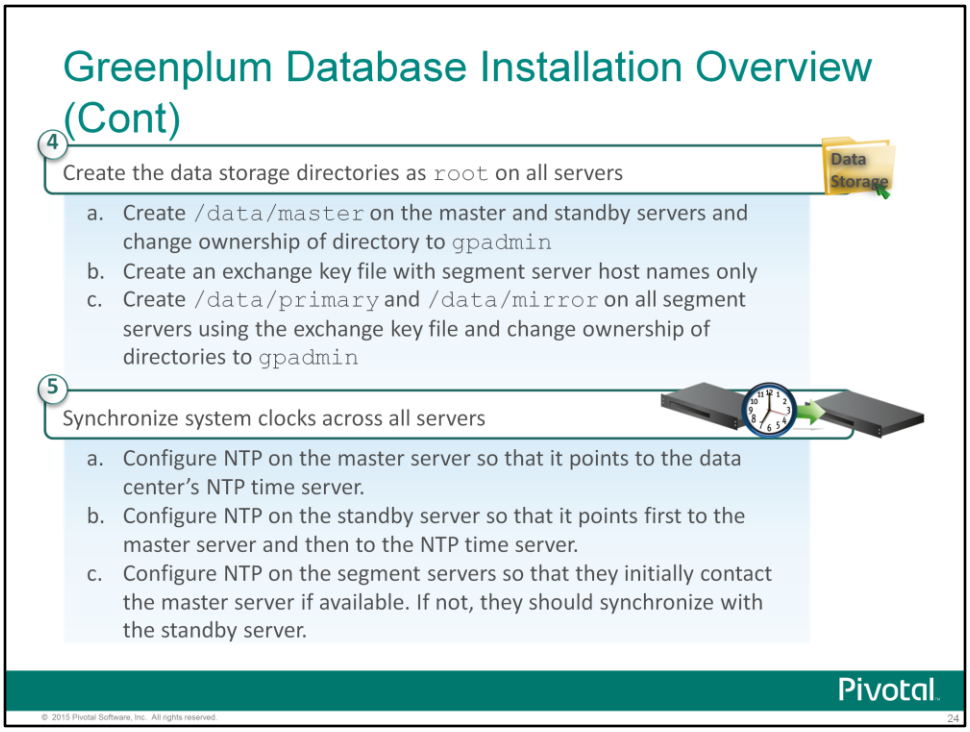
Finally, change the read-ahead value each block device to `16385`, using the following command: `/sbin/blockdev --setra 16385 /dev/block_device_name`, where *`block_device_name`* could be `sdb`.

Disable Transparent Huge Pages (THP). RedHat Enterprise Linux 6.0 or higher enables THP by default. THP degrades Greenplum Database performance.



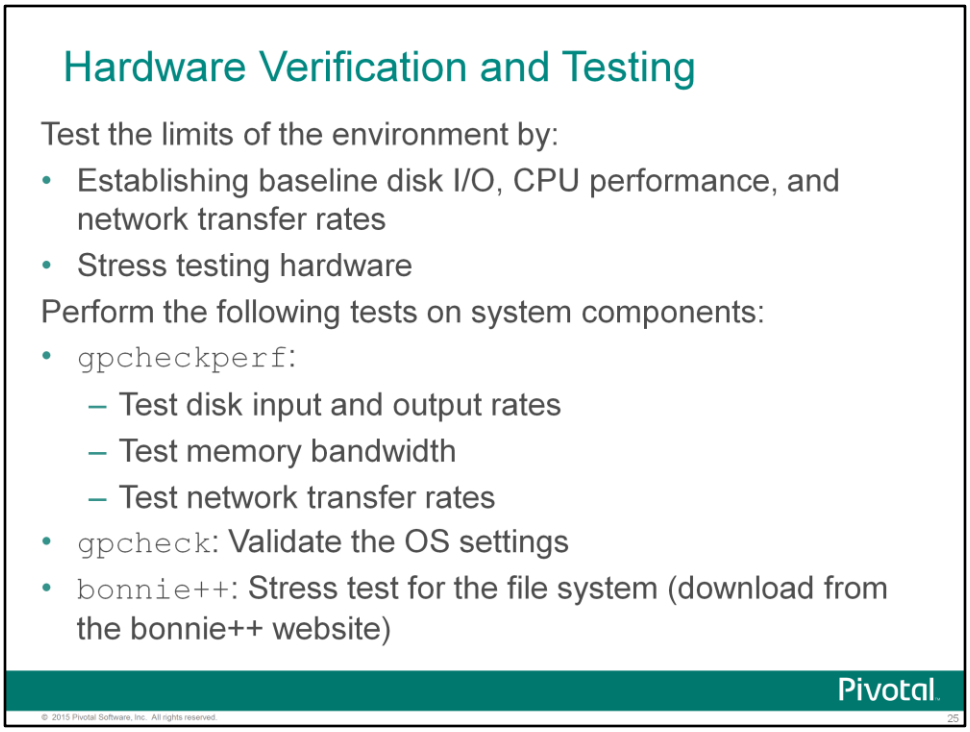
When you begin to install Greenplum software, you always start with the master server. To install the Greenplum Database:

1. Download the Greenplum installer package and install the binaries on the master server. This will install the entire Greenplum DB package including the database server binaries, client programs, documentation, demo programs, management utilities, data loading binaries, bundled programs, and share files.
2. A `greenplum_path.sh` file is created by the installer and has the required settings needed for GPDB. As `root`, source this file after logging into the master server. You will then create an exchange key list that contains the host name of each interface for each segment instance as well as the standby server on a separate line. This file will be used to perform the next step, where you execute `gpseginstall`, to install the binaries on all instances identified in the key exchange file and create the Greenplum system administration user, `gpadmin`, with the password of your choosing. The `gpadmin` user is used for system administrative tasks as well as to run Greenplum. It cannot and should never be run as `root`.
3. Once the installation has completed, log into the master server as the gpadmin user and source the `/usr/local/greenplum-db/greenplum_path.sh` file. You can source this in `/etc/profile` or your Greenplum system administrator's `.bash_profile` file or `.bashrc` file. Next, verify the `gpadmin` user can successfully execute commands on all servers without being prompted for a password using the `gpssh` command. For example, the following command executes the `ls -l` command on all servers:
   `gpssh -f hostfile_exkeys -e ls -l $GPHOME`.



4. As `root`, create the data storage areas on all instances, starting with the master and standby server. These data directories must be owned by the gpadmin user. This can be performed with the following commands:
   ```
   source /usr/local/greenplum-db/greenplum_path.sh
   gpssh -h mdw -h smdw -e 'mkdir /data/master; chown gpadmin /data/master'
   ```
   Next, create the data storage directories on the segment instances. To do this, create an exchange key file that contains the hostname of each segment instance, one per line. Then, execute the gpssh command against that file and create the primary data directory and the mirror data directory on each segment:
   ```
   source /usr/local/greenplum-db/greenplum_path.sh
   gpssh -f hostfile_gpssh_segonly -e 'mkdir /data/primary /data/mirror; chown gpadmin /data/primary /data/mirror'
   ```
5. Synchronize system clocks across all servers by configuring network time protocol (NTP) services on the master, standby, and segment instances. This requires setting the service first on the master server to point to the data center's NTP time server. Then configure NTP on the standby server. The standby server should point to the master server first. If the master service is not available, it should point to the data center's NTP time server. All segment servers should first point to the master server and then to the standby server, should the master server become unavailable.

Once the database has been installed, you are ready to test the hardware and operating system values before initializing the database.

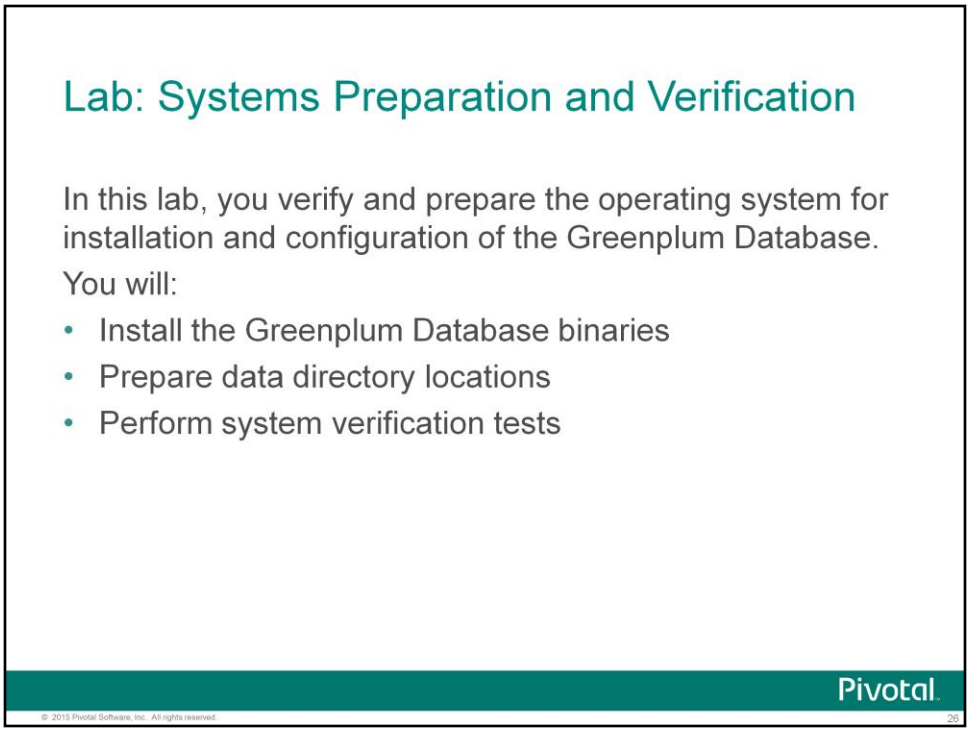


When the Greenplum Database is processing large amounts of data and complex DW queries, it will stress hardware to its limits and utilize all of the resources available. It is therefore important to:

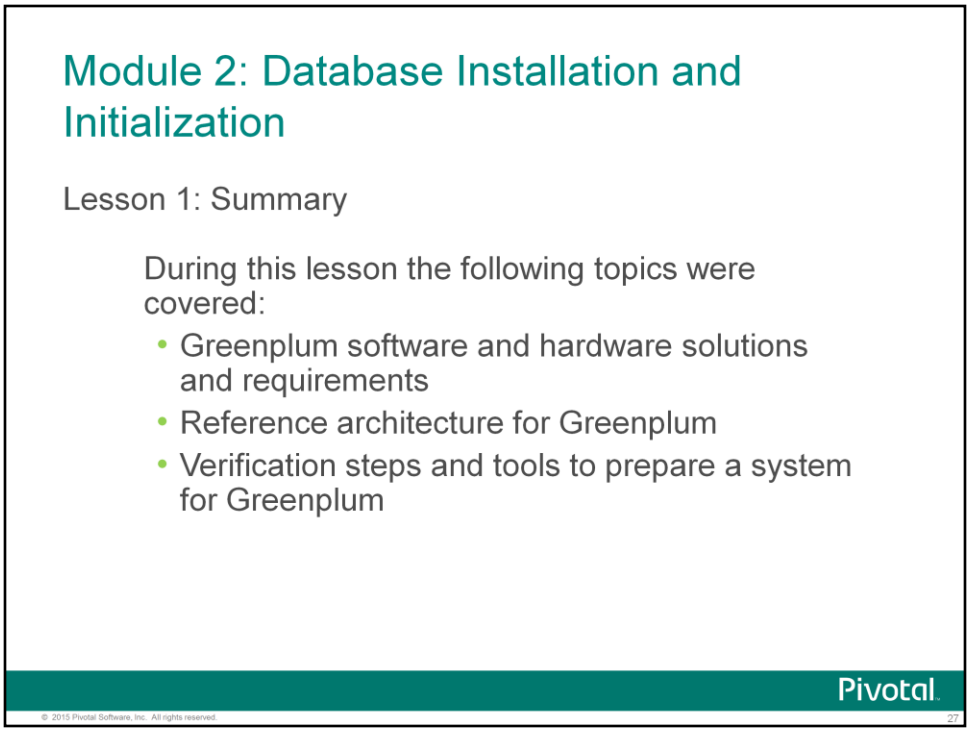
- Perform tests to determine the baseline disk I/O, CPU performance, and network transfer rates. It will help you differentiate hardware performance problems from software tuning performance problems.
- Perform stress tests to burn-in your hardware and flush out any bad disks, processors, memory cards, or network interfaces prior to installing Greenplum DB. The first large table load into Greenplum DB will find these problems, so it is better to stress the hardware and uncover these issues first. Even brand new disks can be corrupted.

To validate the configuration and performance of your system, run the following commands:

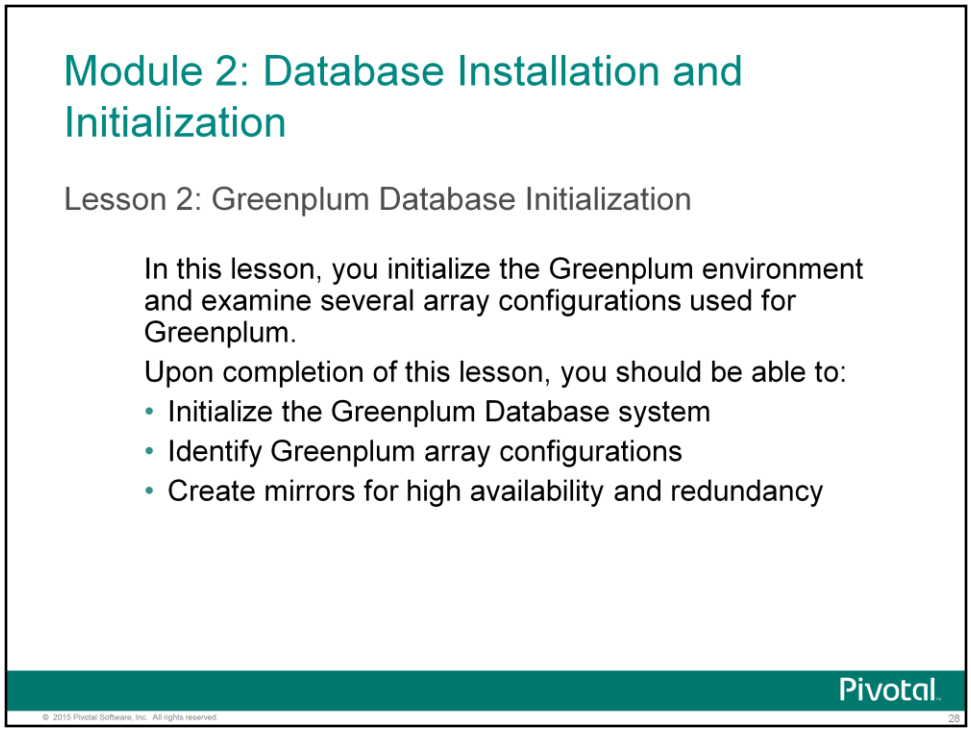
- `gpcheckperf` – The `gpcheckperf` command performs tests on the:
    - Disk I/O rates using the `dd` command
    - Network performance using the `netperf` command
    - Memory bandwidth using the `stream` test
- `gpcheck` – This command validates the operating system settings for all hosts in the array. The output of the command explains the fixes that must be made to the appropriate host in the array. The command accepts a file with the name of each host in the array per line.
- Bonnie++ is a free file system benchmarking tool for Unix-like operating systems, developed by Russell Coker. Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance.



In this lab, you will prepare your system for installation and then install the Greenplum binaries. You will also verify the state of the system by running verification tests.

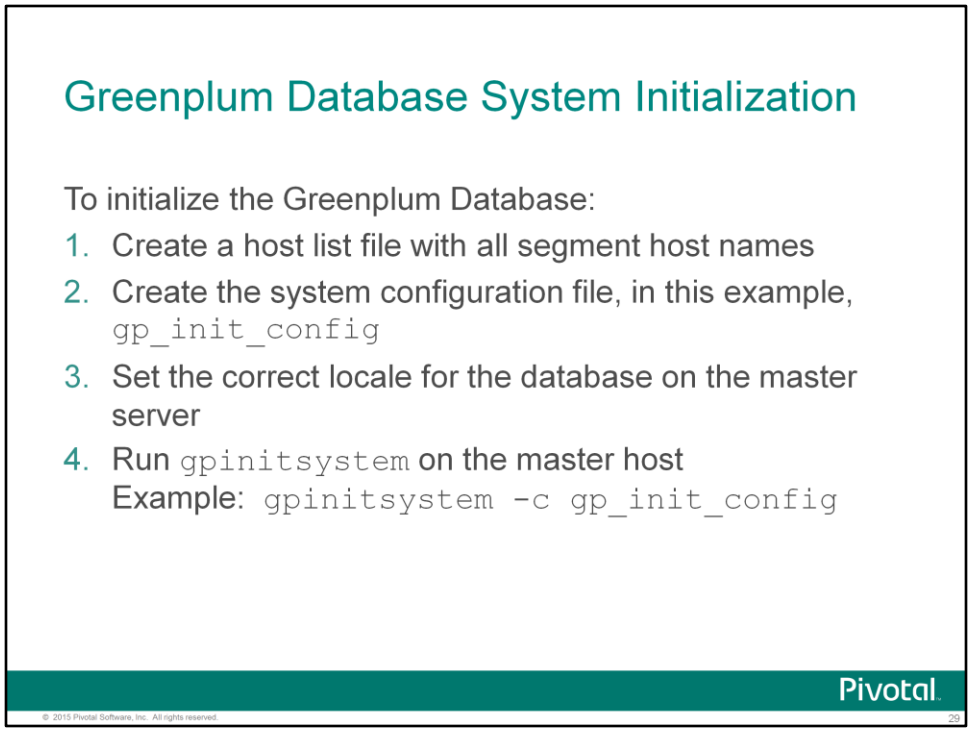


This lesson covered the hardware and software solutions available for the Greenplum Database as well as the requirements for the system. The lesson discussed the reference architecture that can be used to build a production-ready environment for the Greenplum Database. The verification steps and tools that are used to prepare a system for the installation, initialization, and day-to-day operations of the Greenplum Database environment were also discussed.



In this lesson, you will:

- Continue configuring the Greenplum environment by initializing the database instances.
- Examine the array configurations possible for the Greenplum environment.
- Create mirrors to support high availability and redundancy in the Greenplum environment.

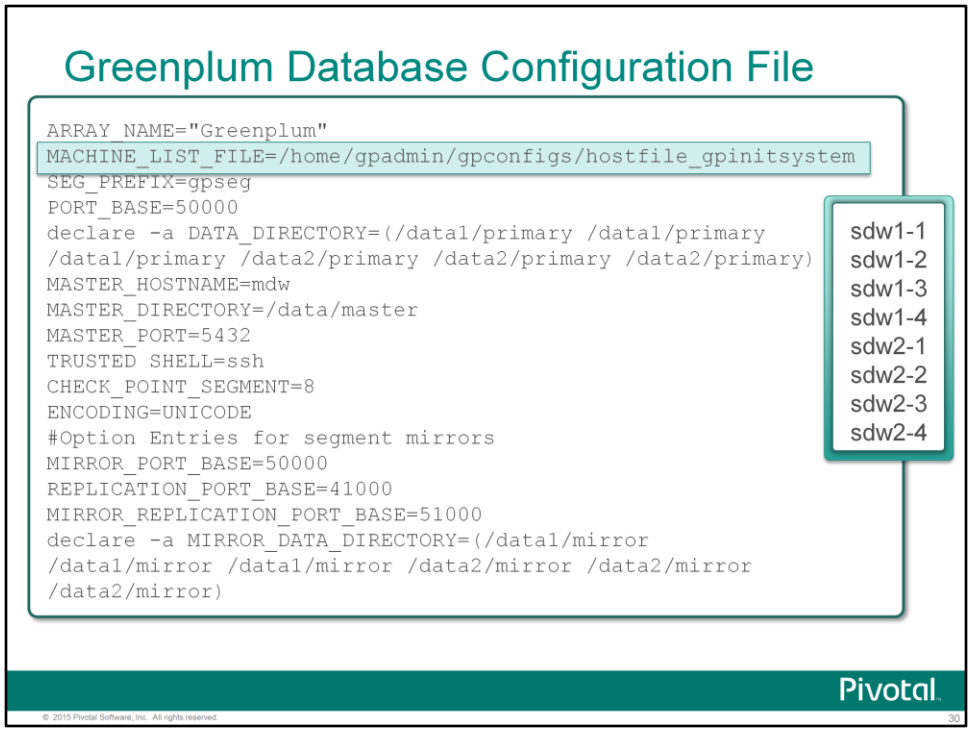


In a Greenplum DB, each database instance, the master and all segments must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS. Greenplum has its own version of `initdb` called `gpinitsystem`, which initializes the database on the master and on each segment instance. This differs slightly from its predecessor, PostgreSQL.

To initialize the Greenplum Database, you must:

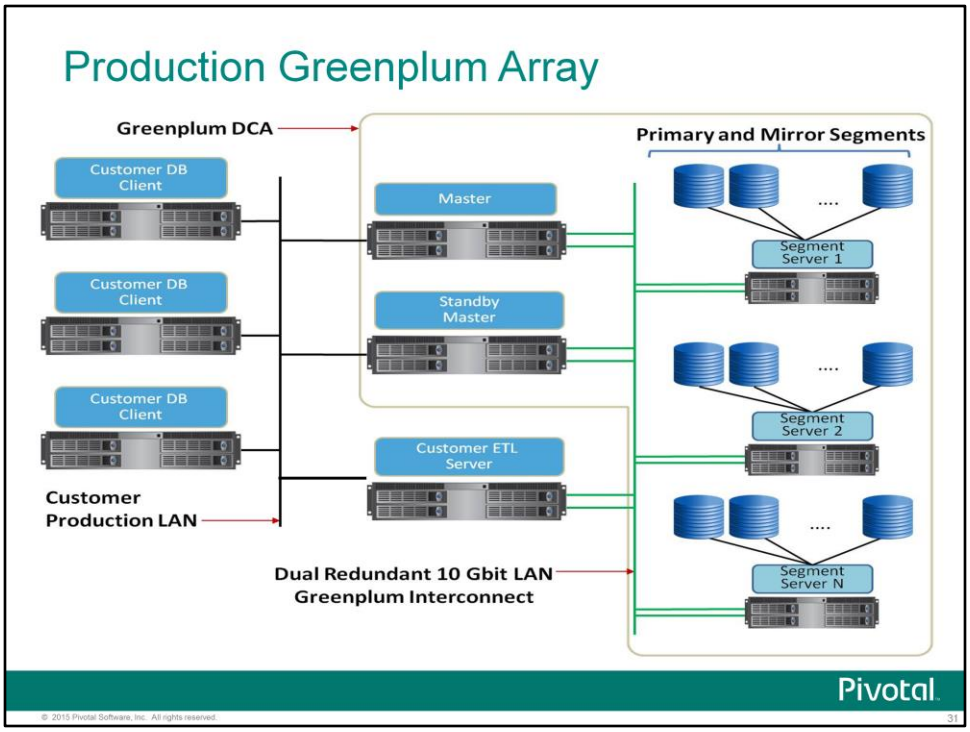
- Create a host list file that contains the name of the segment hosts involved in the array.
- Create a configuration file that describes the system setup, including how many segments per host, data directory locations, if mirroring is enabled, and so on. If you are using multiple network interfaces and have multiple segment hostnames per host, it is important that the host file referenced in the configuration has ALL per-interface hostnames for a segment.
- On the master server, define the locale you wish to use for the database. Some locale options cannot be changed after initialization or upgrade, so it is best to verify the setting before.
- Execute the Greenplum Database initialization utility to create the database instance specified in the system configuration file.

After the Greenplum database system has been initialized, it is then ready for use. You can then create and manage databases as you would in a regular PostgreSQL DBMS.



The Greenplum Database Configuration file contains the required parameters necessary for initializing your Greenplum Database. The file defines the following parameters:

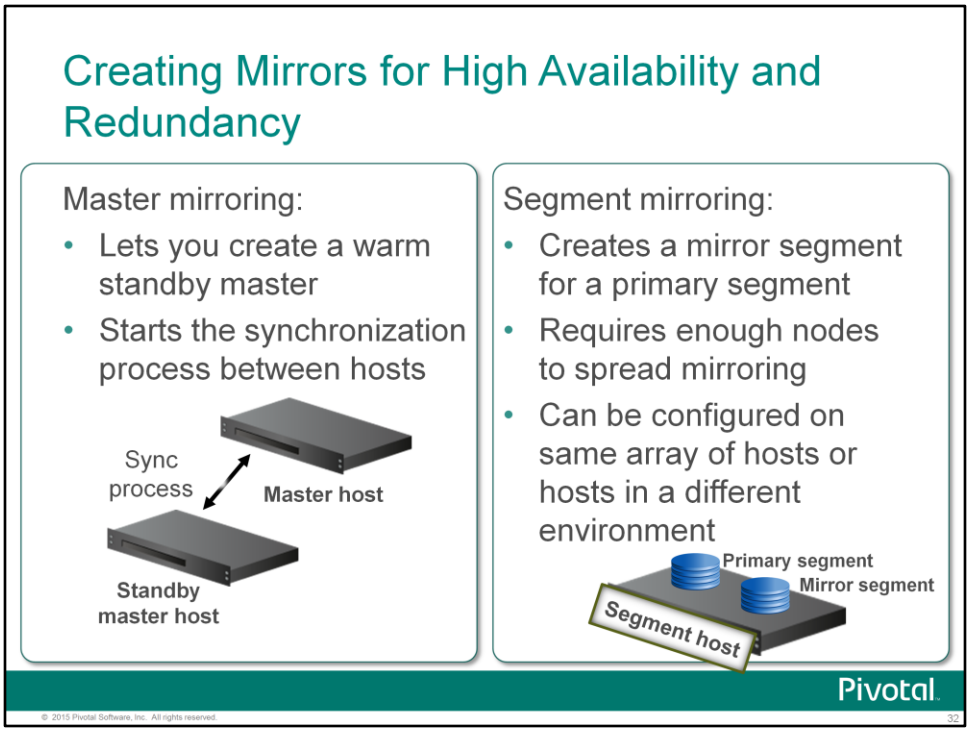
- **ARRAY_NAME** – The unique name of your array
- **MACHINE_LIST** – File name containing the list of segment hosts in the array
- **SEG_PREFIX** – The prefix used for the data directories on the master and segment instances
- **PORT_BASE** – The base port number for the primary segments – this value is incremented by 1 for each primary segment defined
- **DATA_DIRECTORY** – The data directories for the primary segments
- **MASTER_HOSTNAME** – The hostname of the master host
- **MASTER_PORT** – The port number of the master instance
- **TRUSTED_SHELL** – The type of shell gpinitsystem uses to execute commands on remote hosts
- **CHECK_POINT_SEGMENT** – The maximum distance between automatic write ahead logs(WAL) checkpoints
- **ENCODING** – The character encoding to use
- **DATABASE_NAME** – This optional parameter specifies the database to create on initialization
- **MIRROR_PORT_BASE** – The base port number for the mirror segments – this value is incremented by 1 for each mirror segment defined
- **REPLICATION_PORT_BASE** – The base number by which the port numbers for the primary file replication process are calculated - this value is incremented by 1 for each primary segment defined
- **MIRROR_REPLICATION_PORT_BASE** – The base number by which the port numbers for the mirror file replication process are calculated - this value is incremented by 1 for each mirror segment defined
- **MIRROR_DATA_DIRECTORY** – The data directories for the mirror segments



In this configuration, there is one master, one standby master, and two or more segments per host, with one primary segment per CPU core. The network configuration shows that there are multiple network interfaces per segment. Each interface is connected to the interconnect.

An ETL server is connected directly to the interconnect as well to improve performance for data loads.

This represents one type of array configuration that is commonly used in production environments.



You can enable mirroring in the Greenplum database during the database initialization phase at setup time, or to an existing system that was configured without mirroring.

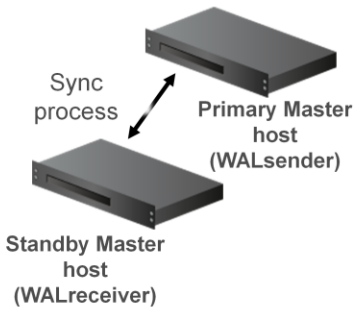
There are two types of mirrors that can be configured:

- **Master mirroring** – When mirroring the master, a warm standby master is created. Once configured, the synchronization or replication process between the master and the standby master is executed. This synchronization process, `gpsyncagent`, is executed from the standby master and ensures that the data between both systems are synchronized. Should the master host become unavailable:
    - The replication process is stopped.
    - The replication logs are used to reconstruct the state of the master at the time of the failure.
    - The standby master can be activated to pick up from the last set of successful transactions completed by the master.
- **Segment mirroring** – A mirror segment is normally configured on a different host than its primary counterpart. It can be configured on systems outside of the array. Changes to the primary segment are copied over to the mirror segment using a file block replication process. Until a failure occurs, there is no live segment instance running on the mirror host, only the replication process. Should the primary segment become unavailable:
    - The file replication process is stopped.
    - The mirror is automatically onlined as a primary segment.

Warm Standby Master

A warm standby master is:

- A replica of the Greenplum master instance (system catalogs)
- Used to remove single point of failure
- Kept up to date by the *WALsender* (Primay Master) and *WALreceiver* (Standby Master) replication processes
- System Catalogs and Transaction Logs are replicated.

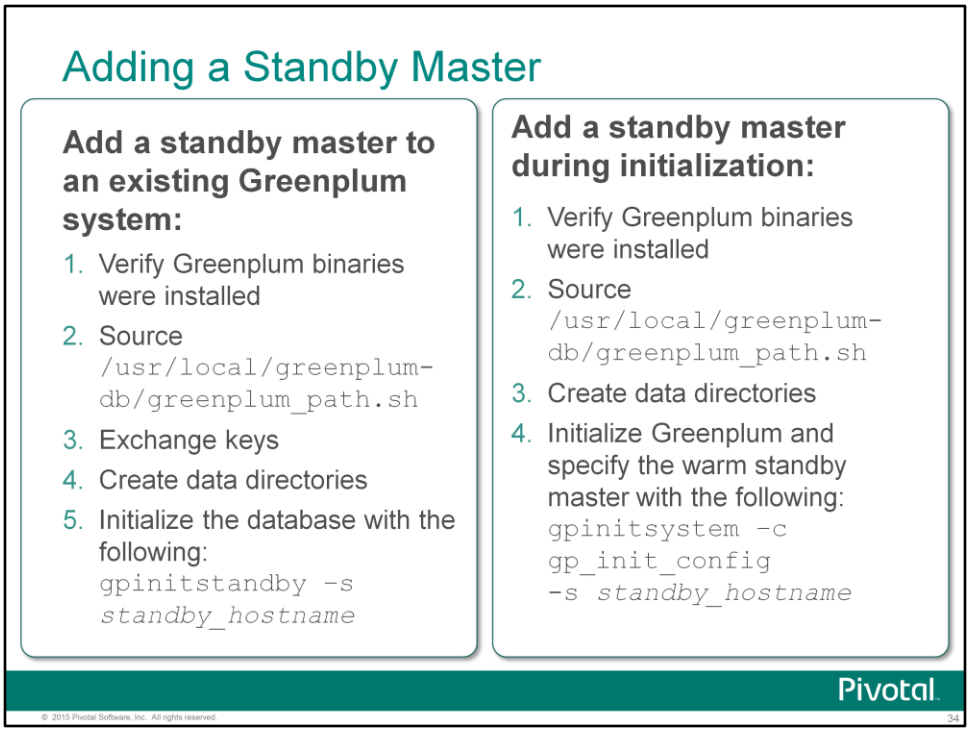


Pivotal

33

A backup or mirror master host serves as a warm standby in the event of the primary master host becoming inoperative. The warm standby master is kept up to date by a transaction log replication process, which runs on the backup master host and keeps the data between the primary and backup master hosts synchronized. The master does not have user data, only system catalog tables. By creating the warm standby master, you ensure that the master is no longer a single point of failure.

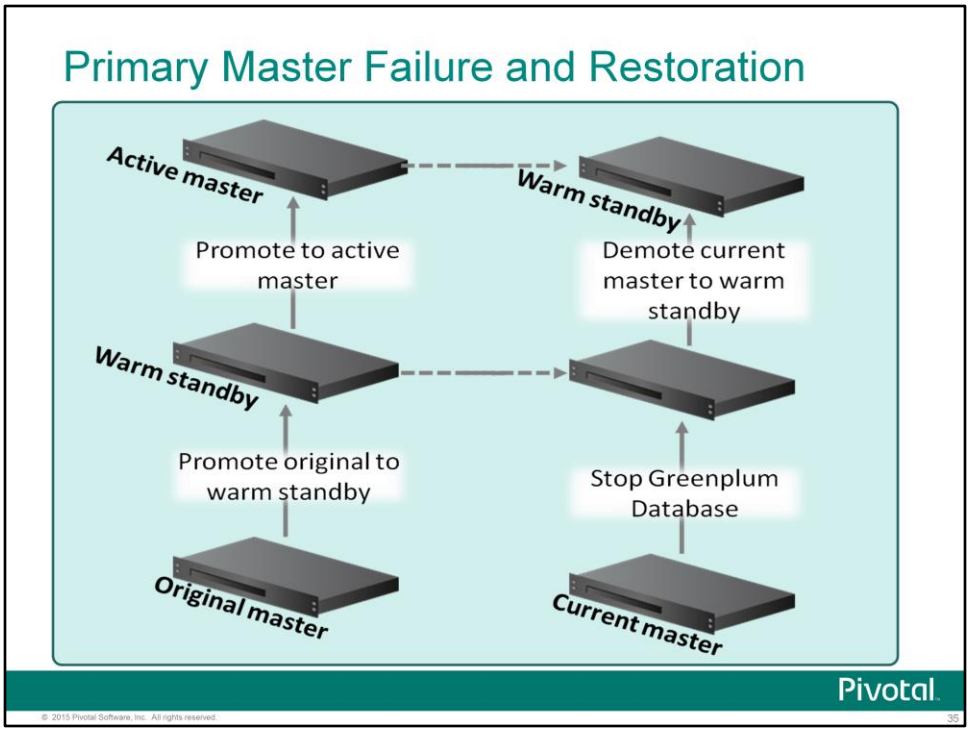
You can enable a standby master on a segment host or on a new host. If you are creating the standby master on a host outside of the existing array, verify that the `pg_hba.conf` file of the current master and segments allow connections from the new standby master host. If you add the standby at initialization time, this process is automatically completed for you.



To add a standby master to an existing Greenplum database system:

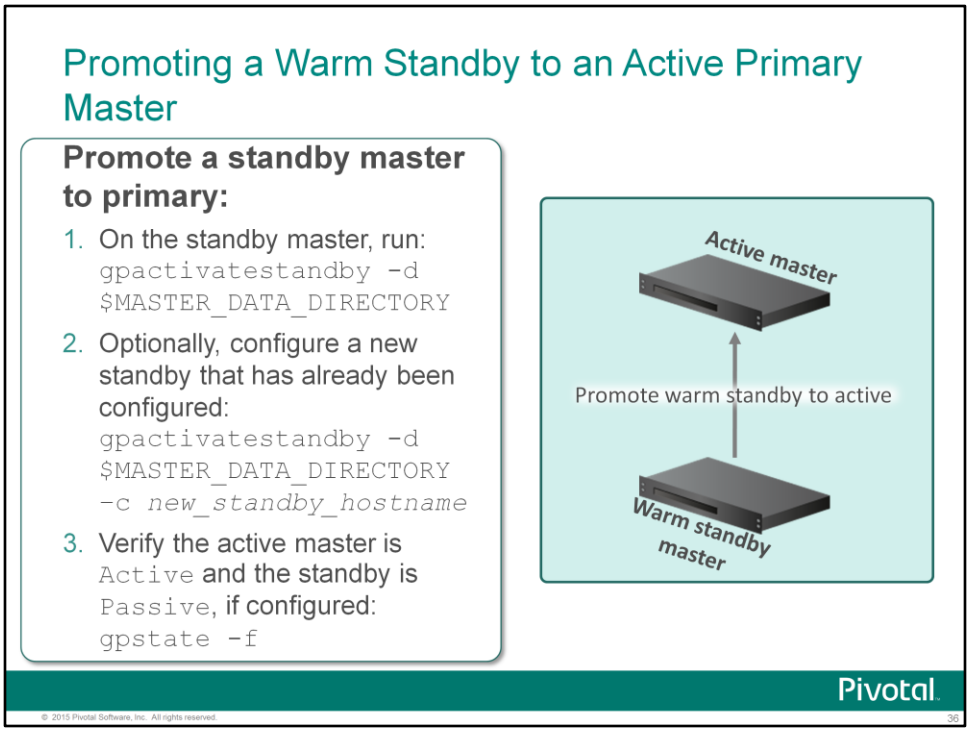
- Verify Greenplum has already been installed and configured on the standby master and that host keys have already been exchanged. This includes ensuring that the superuser account has been created, the binaries for Greenplum installed, the variables configured, keys exchanged, and data directories created on the standby master.
- Run `gpinitstandby` on the active primary master. Include the `-s` option followed by the host name of the standby master.

At initialization time, include the `-s` option followed by the hostname of the standby master when executing the `gpinitsystem` command.



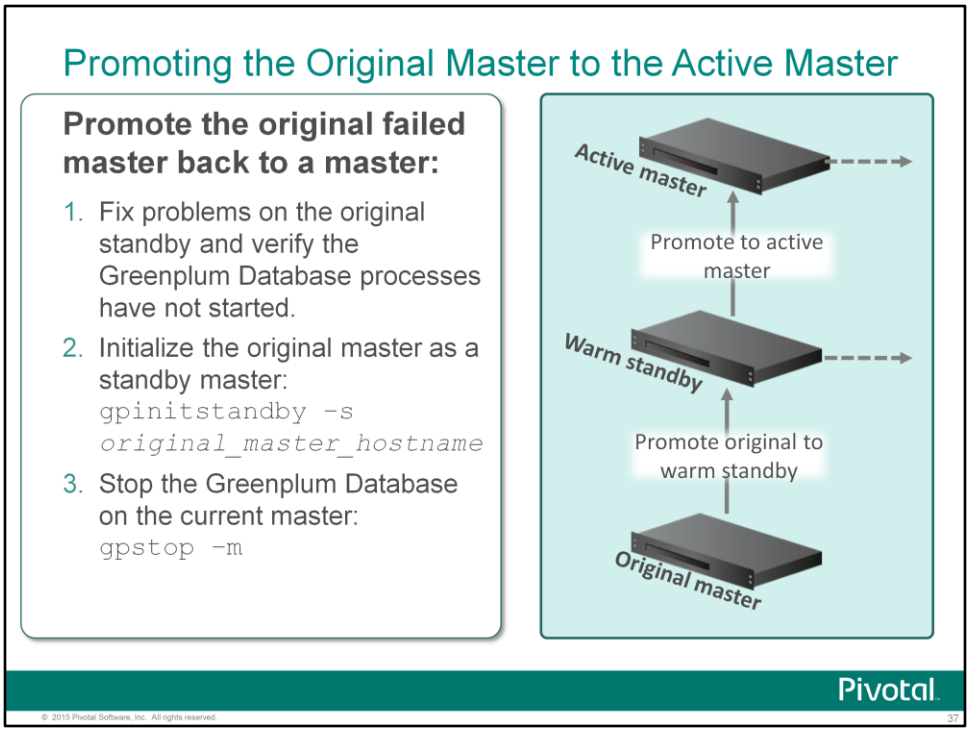
The warm standby master offers a measure of protection for your database by assuming the role of primary should the original primary master fail. The synchronization process, `WALsender and WALreceiver`, allows the standby to be engaged with little to no loss of data by keeping the transaction log data between the primary and standby masters synchronized. In-flight transactions may need to be restarted.

When the primary master fails, the warm standby is manually promoted as the active master. Once the primary master becomes available, it can be promoted to its original role, demoting the warm master from the active mode to its original warm standby role.



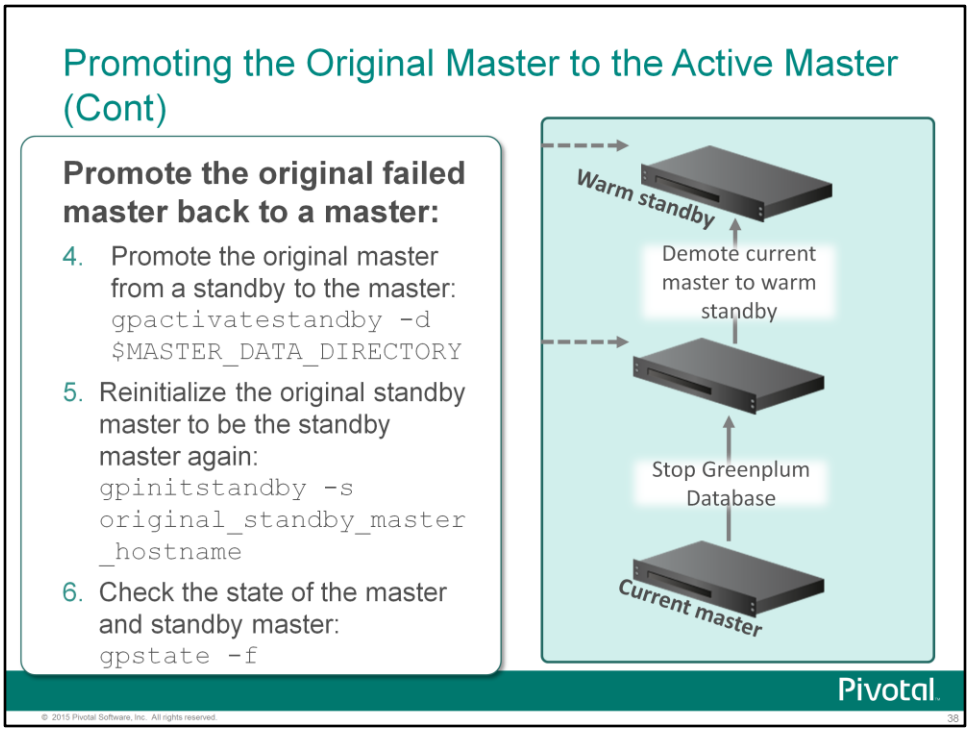
To promote the warm standby you configured to be the master:

1. Run the `gpactivatestandby -d $MASTER_DATA_DIRECTORY` command on the warm standby. This promotes the warm standby to now be the active primary master.
2. Optionally, define a new warm standby with the `gpactivatestandby` command. The `-c` option allows you to specify the hostname to the new warm standby.
3. Once you have promoted the warm standby to be an active primary master, verify the state of the servers with the `gpstate -f` command.

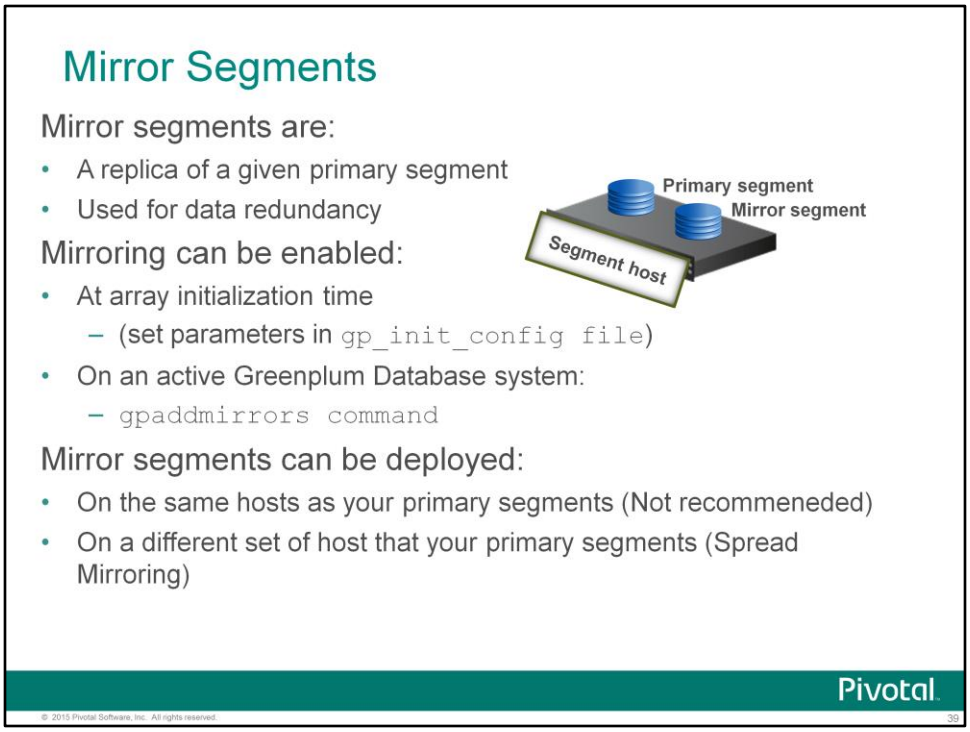


If the original master has been restored, you can promote it back to an active master with the following steps:

1. Ensure that the Greenplum Database is not running on the original master. The in stance should already be running on the server currently acting as the primary master.
2. Initialize the original master so that it is now a warm standby. This is done with the `gpinitstandby` command with the `-s` option that allows you to specify the name of the original master.
3. Stop the Greenplum database on the current master using the `gpstop -m` command. The `-m` option stops only the master.



4. Promote the original master, which is now acting as a warm standby, to be the active primary. Use the `gpactivatestandby` command to promote the original master.
5. Initialize the master that was acting as a primary master to now act as a standby master. Use the `gpinitstandby` command to demote this server.
6. Check the state of the master and current standby servers using the `gpstate -f` command. The original master should now show as Active and the standby status should now be Passive.



Mirrored segments replicate a primary segment counterpart providing redundancy and fault tolerance support for your data. Database queries can fail over to a mirror segment should the primary segment be unavailable.

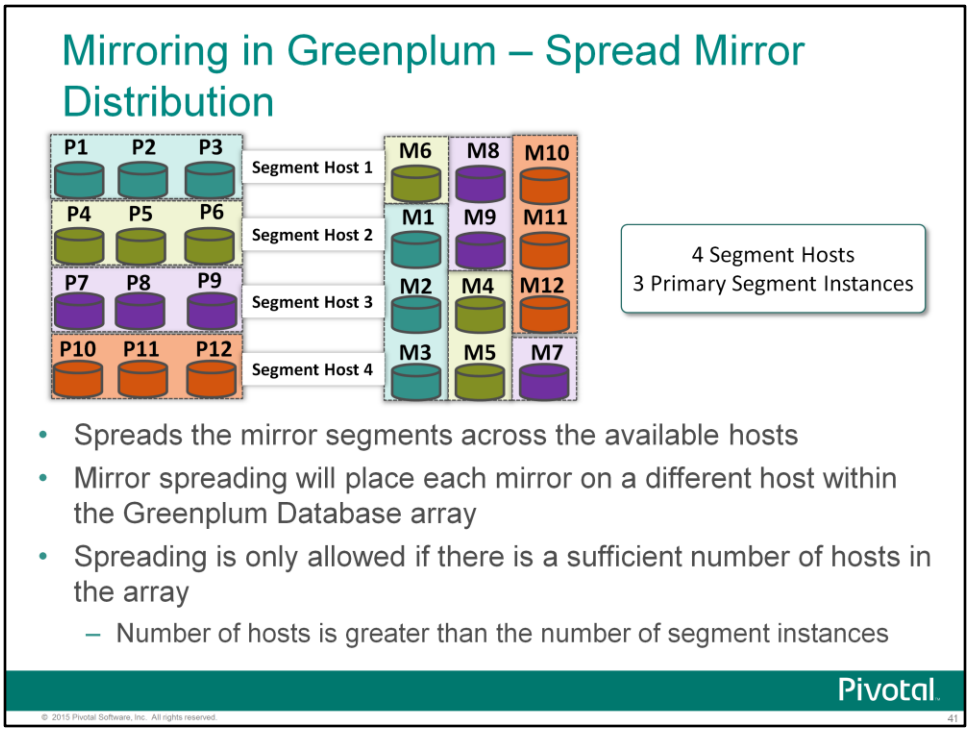
Mirroring can be enabled:

- At array initialization by setting parameters in the `gp_init_config` system configuration file you defined earlier.
- On an active Greenplum Database using the same hosts as the primary segments by:
    1. Creating data storage for the mirror on a different file system than the primary segment is using
    2. Exchanging SSH keys as segment hosts must be able to SSH and SCP to each other without a password
    3. Running the `gpaddmirrors` utility with the `-p` option, followed by the number to add to your primary port segment number to determine the port number for the mirror

Mirror Segments (continued)

- On an active Greenplum Database using different hosts than the primary segments by:
    1. Verifying all segments have Greenplum installed
    2. Allocating data storage for mirror segments on all segment hosts
    3. Exchanging SSH keys
    4. Creating a configuration file with the host names, ports, and data directories where you want your mirrors created. You can use `gpaddmirrors -o` *`filename`* to create a sample file that you can edit with your true parameters.
    5. Running the `gpaddmirrors -i` *`filename`*, where *`filename`* is the name of the file that contains the mirror configuration.

Note: If using a multi-NIC configuration (multiple host names per segment host), DO NOT enable mirrors at initialization time. Do so after you have initialized and edited the system catalog. You can then run `gpaddmirrors`.

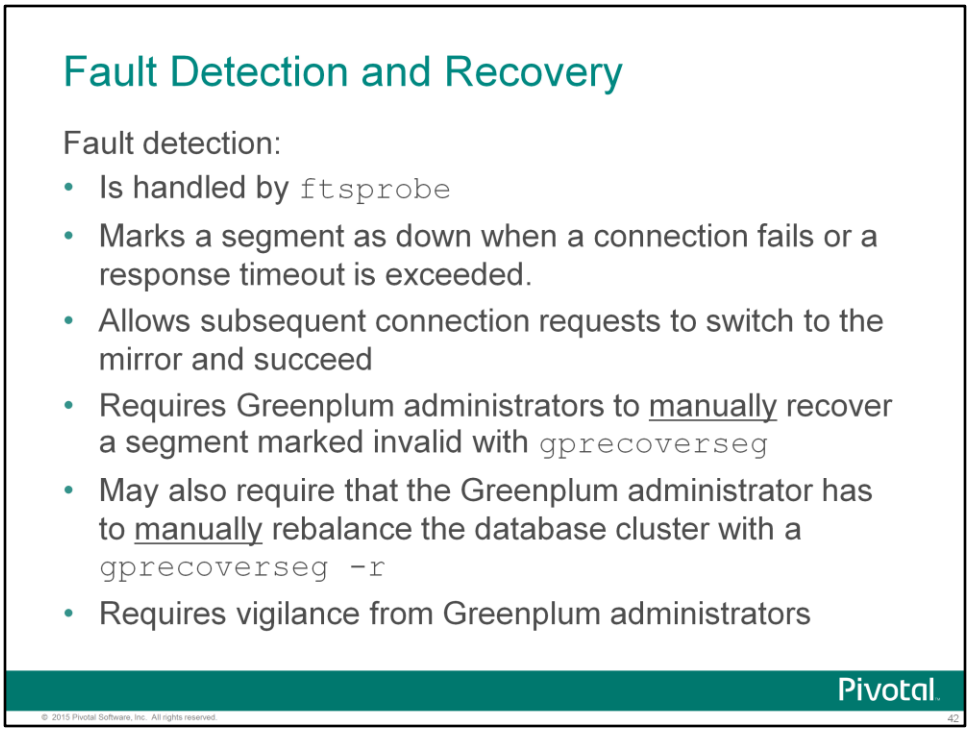


In this example, we focus on the spread mirror distribution for 4 segment hosts with 3 primary segments per host.

In spread mirror distribution, mirror segments are spread across the segment servers so that no single server contains more than one mirror for another segment server. This spreads the mirror segments across the available hosts, balancing the mirrors and reducing the chances that the failure of a single segment will impact the database.

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. To meet this requirement, the number of hosts must be greater than the number of segment instances.

Note that cluster expansions and upgrades need to be carefully planned out to insure proper spreading of new mirrors across all the servers in the cluster.



Fault detection is handled by `ftsprobe` , a Greenplum Database server subprocess. This fault detection process monitors the Greenplum array, scanning all segments and database processes at configurable intervals.

Whenever the fault detection process cannot connect to a segment, it marks that segment instance as down in the Greenplum Database system catalog. Once a segment is down, it will remain out of operation until an administrator initiates the recovery process to bring that segment back online.

When mirroring is enabled in a Greenplum Database system, the system will automatically failover 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.

Fault Detection and Recovery (continued)

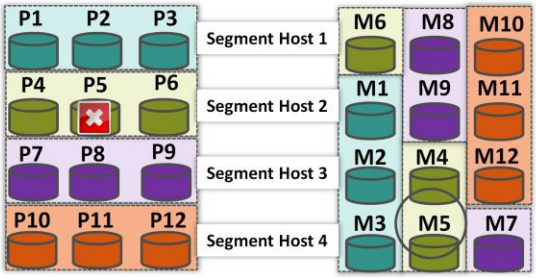
To recover failed segments in the system, the Greenplum administrator the `gprecoverseg` recovery utility. This utility:

- Locates the downed segments
- Checks if they are valid
- Compares the transactional state with the currently active segment to find out what changes were missed when the segment was offline.
- Synchronizes only the changed database files with the active segment
- Brings the segment back online.

This recovery process is performed while the Greenplum Database system is up and running.

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

Mirroring in Greenplum – Failed Primary Segment Example



Primary Segments **Mirror Segments**

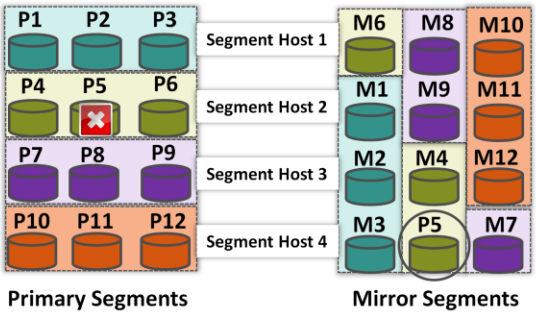
Primary Segment Failure:

1. The `ftsprobe` process on the master detects the segment down and marks it invalid.

Over the next few slides, we will examine how the failure of a segment is detected and handled by Greenplum Database.

Primary segment P5 on segment host 2 is marked down and invalid. The `ftsprobe` process initiates a failover to mirror segment M5 mirror on segment host 4.

Mirroring in Greenplum – Failed Primary Segment Example



Primary Segment Failure:

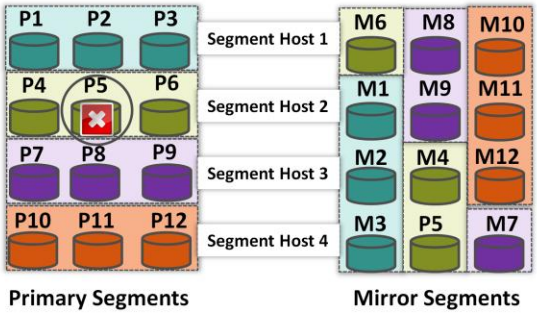
1. The `ftsprobe` process on the master detects the segment down and marks it invalid.
2. The mirror segment is validated to ensure that it was synchronized with its primary segment. Mirror segment becomes the primary.

Pivotal

45

Mirror segment M5, on segment host 4, is validated to be in sync with the corresponding primary that became unavailable. The M5 mirror is now promoted to become the primary segment P5.

Mirroring in Greenplum – Failed Primary Segment Example

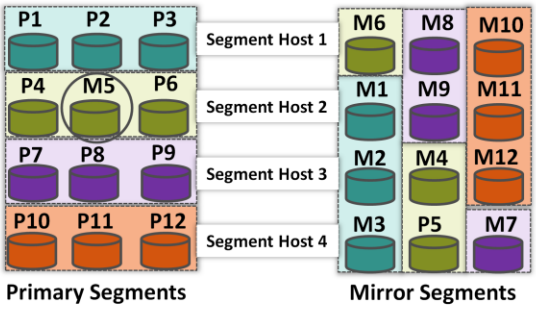


Primary Segment Failure:

3. Once it has been determined why the original primary P5 went down and is repaired, you can then bring that segment back online to become the mirror segment to protect the new primary P5.

Once repaired and made available, the original P5 primary segment can be brought back online. When brought back online it will become the mirror M5, protecting the new P5 primary segment.

Mirroring in Greenplum – Failed Primary Segment Example



Primary Segment Failure:

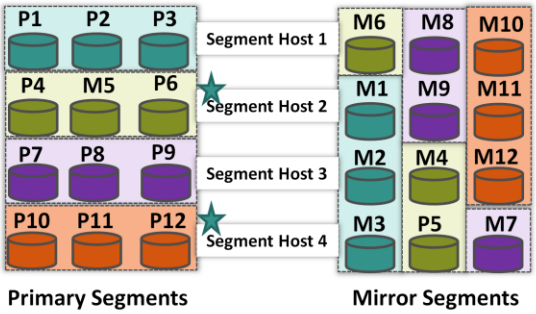
4. Once it has been determined why the original primary P5 went down and is repaired, you can then bring that segment back online to become the mirror segment to protect the new primary P5.
5. Run the gprecoverseg command to bring back up the down segment. The old primary P5 becomes the M5 mirror.

Pivotal

Use the `gprecoverseg` command to restore the original roles of the primary and mirror segments. The database stores information on the roles as originally configured as well as their current status in the `gp_segment_configuration` table. By issuing the `gprecoverseg` command, Greenplum:

- Brings recovered segments back online.
- Restores all segments to their original roles
- Verifies that all segments are valid
- Synchronizes the changed database files with the active segment

Mirroring in Greenplum – Re-balancing Segments

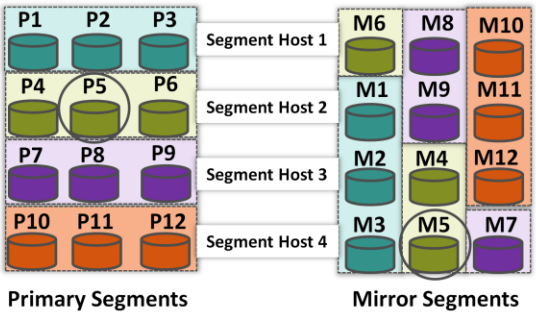


The database cluster is in an un-balanced condition

- Segment Host 2: Two Primaries, 4 Mirrors
- Segment Host 4: 4 Primaries, 2 Mirrors

Before rebalancing the segments, segment host 2 and segment host 4 are in an un-balanced condition. Segment host 2 has 2 primaries and 4 mirrors, and segment host 4 has 4 primaries and 2 mirrors. From a processing of queries perspective, segment host 4 will be processing more and could cause query performance issues. To re-balance the database cluster, run `gprecoverseg -r` command to accomplish this.

Mirroring in Greenplum – Re-balancing Segments



The database cluster is in an un-balanced condition

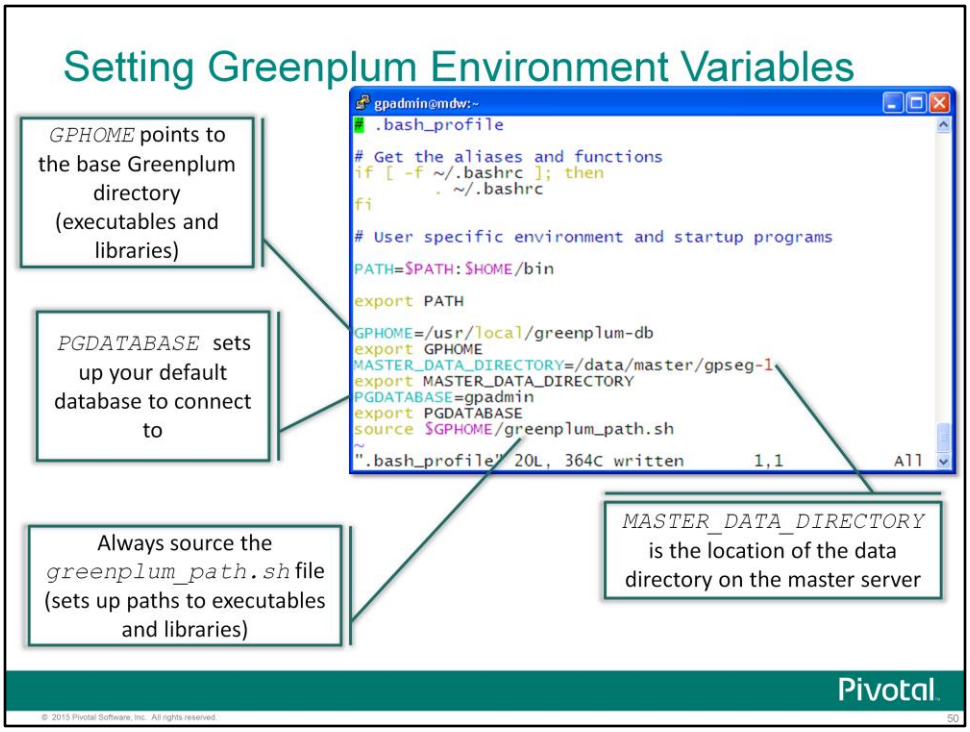
- Segment Host 2: Two Primaries, 4 Mirrors
- Segment Host 4: 4 Primaries, 2 Mirrors

Run the following command to re-balance the cluster

- `gprecoverseg -r`

Pivotal

Once `gprecoverseg` has successfully executed, the original roles are assigned to the segments.



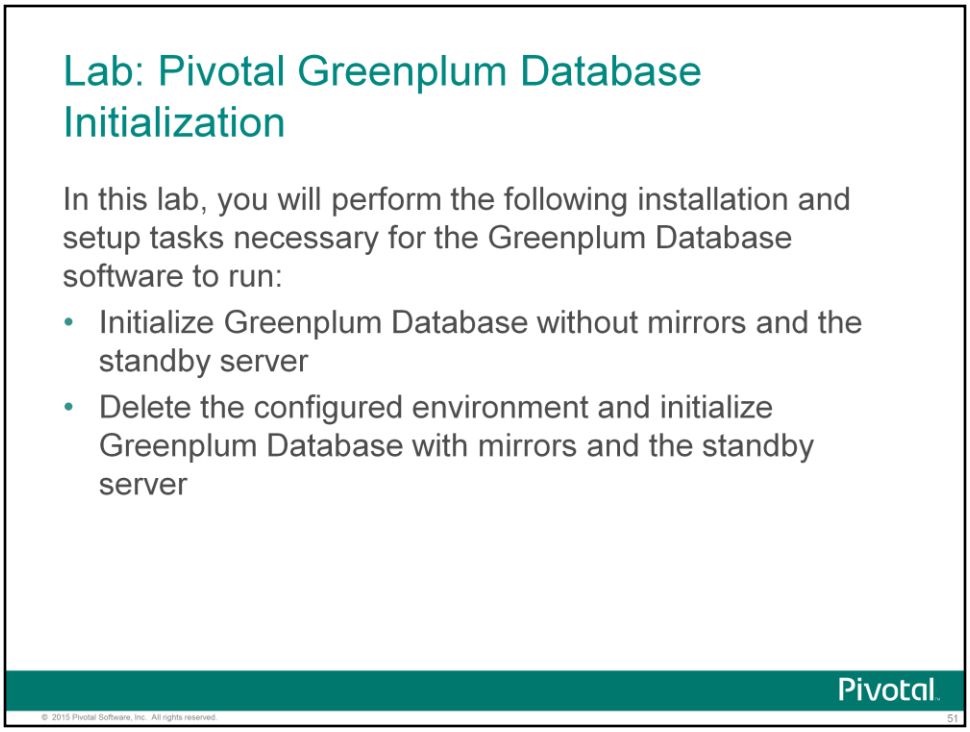
Once you have initialized the Greenplum Database, you must configure the Greenplum environment by setting up or accessing several variables that will be used when you access or administer the database.

To configure the environment, perform the following on the master and standby servers:

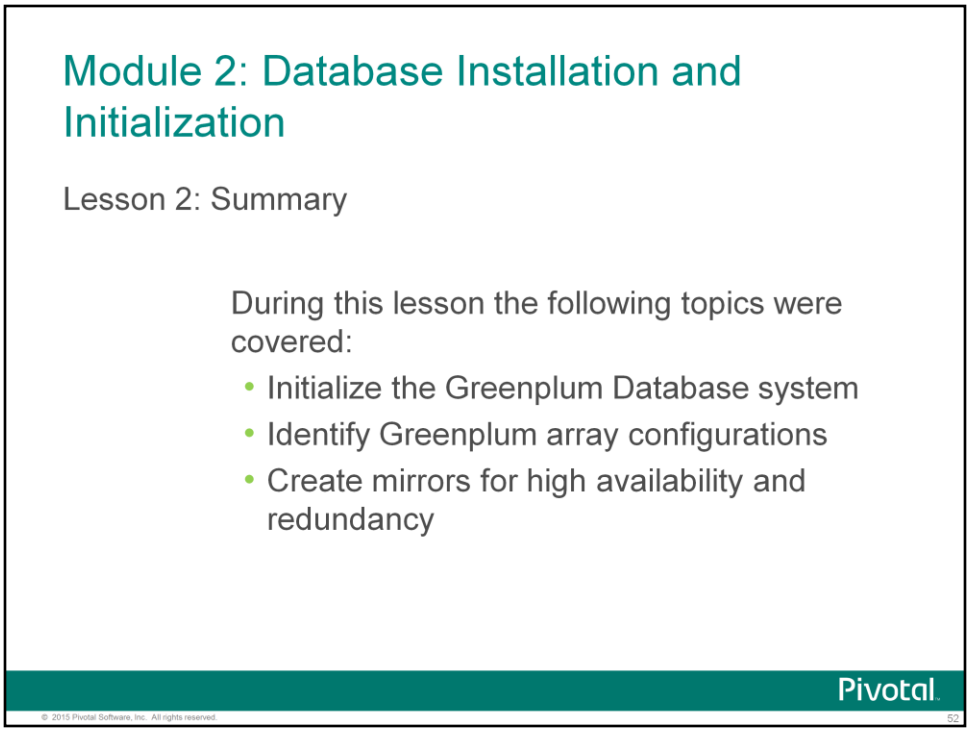
- Set the value of the MASTER_DATA_DIRECTORY variable to the location of the data directory on the master server. This variable is used when starting, stopping, or otherwise accessing the Greenplum Database.
- Source the contents of the /usr/local/greenplum-db/greenplum_path.sh file. This will set the value of other environment variables used by Greenplum.

Add these settings to the `.bash_profile` or `.bashrc` file for the gpadmin user. By adding it to the `.bash_profile` file, the variables will be read once when you log into the system as the `gpadmin` user. If you add it to the `.bashrc` file, it will be read for each separate shell session you initiate as the `gpadmin` user.

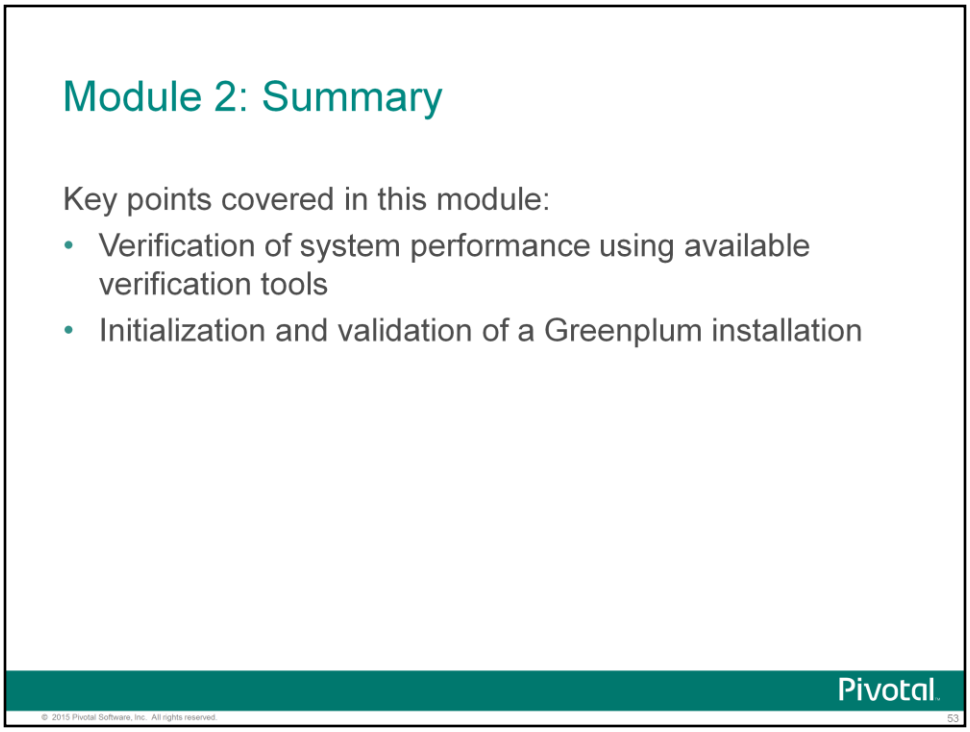
Optionally, you can configure several other variables that will override the behavior of the Greenplum client, `psql`. These variables will be explained in greater detail later in the course.



In this lab, you will continue configuring your Greenplum database environment by initializing the Greenplum Database system. You will also configure mirrors for the primary segments you have configured as part of the lab.



This lesson covered how to initialize the Greenplum Database environment, by configuring all components at once, or by configuring just the master and primary segments for the environment. Different array configurations were shown for production and standalone or testing environments. Finally, a discussion on creating and recovering mirrors for high availability and redundancy were also discussed.



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

- Verify system performance by using available verification tools after installing the Greenplum binaries.
- Initialize and validate the Greenplum installation.

This slide is intentionally left blank.

Pivotal