

Pivotal HAWQ

Version 1.1

Installation Guide

Rev: A11

Table of Contents

1	System Requirements	4
1.1	Preparing HDFS	4
1.2	Installing HAWQ	7
1.2.1	Install the HAWQ Binaries	7
1.2.2	Creating the gpadmin User	9
1.2.3	Setting the OS Parameters	10
1.2.4	Editing the Configuration Files	13
1.2.5	Ensuring that HDFS works	15
1.3	HAWQ on Secure HDFS	16
1.3.1	Requirements	16
1.3.2	Preparation	16
1.3.3	Configuration	17
1.3.4	Troubleshooting	18
1.4	Running a Basic Query	19
1.5	Installing Cryptographic Functions for PostgreSQL	19
1.5.1	Upgrading from 1.1.3 to 1.1.4	20
1.5.2	Automated Installation of pgcrypto	20
1.5.3	Uninstalling pgcrypto	20
1.5.4	Installing PL/R	21
1.6	HAWQ Configuration Parameter Reference	21

Copyright © 2014 GoPivotal, Inc. All rights reserved.

GoPivotal, Inc. believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." GOPIVOTAL, INC. ("Pivotal") MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any Pivotal software described in this publication requires an applicable software license.

All trademarks used herein are the property of Pivotal or their respective owners.

Use of Open Source

This product may be distributed with open source code, licensed to you in accordance with the applicable open source license. If you would like a copy of any such source code, Pivotal will provide a copy of the source code that is required to be made available in accordance with the applicable open source license. Pivotal may charge reasonable shipping and handling charges for such distribution.

This document describes how you can install HAWQ manually. HAWQ can be installed along with Pivotal HD Enterprise using the Command Line Interface (CLI); however, if you choose to not install HAWQ using the CLI, then you need to follow the instructions in this chapter.



Note:

This document does not describe how to install the Pivotal Extension Framework (PXF), which enables SQL querying on data in the Hadoop components such as HBase, Hive, and any other distributed data file types. To install Pivotal Extension Framework (PXF), see the *Pivotal Extension Framework Installation and User Guide*.

To install HAWQ manually, perform the following tasks:

- Review the System Requirements
- Prepare HDFS
- Install HAWQ
- Run a Basic Query
- Review the HAWQ Configuration Parameter Reference section



Note:

These tasks should be performed for *a//hosts* in your HAWQ array (master, standby master and segments).

1 System Requirements

Check that you meet the following system requirements before you install HAWQ:

Operating System:

- RedHat 6.4 and 6.2, 64 bit
- CentOS 6.4 and 6.2, 64 bit

Minimum CPU: Pentium Pro compatible (P3/Athlon and above)

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: disks should have at least 30% free space (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, and GNU zip

1.1 Preparing HDFS

You need to complete the following steps to configure HDFS:

1. To make sure HDFS block files can be read by other users, configure OS file system umask to 022.

Add the following line into `${HADOOP_HOME}/etc/hadoop/hadoop-env.sh`

```
umask 022
```

Add the following parameter to `hdfs-site.xml`

```
<property>
  <name>dfs.datanode.data.dir.perm</name>
  <value>755</value>
  <description>Permissions for the directories on on the local filesystem w
    the DFS data node store its blocks. The permissions can either be octal o:
    symbolic.</description>
</property>
```

2. Edit the `/hdfs-install-directory/etc/hadoop/hdfs-site.xml` file. To do this, change the `dfs.block.local-path-access.user` to the user who starts HDFS if the short circuit feature is enabled in `libhdfs3`. See the HAWQ Parameter Reference section for more information.
3. Set the `dfs.namenode.name.dir` and `dfs.datanode.data.dir` to your preferred path as shown in the example:

```
<configuration>
  <property>
    <name>dfs.support.append</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.client.read.shortcircuit</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.block.local-path-access.user</name>
    <value>gpadmin</value>
    <description>
      specify the user allowed to do short circuit read
    </description>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/gpadmin/hadoop-2.0.0-alpha/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/gpadmin/hadoop-2.0.0-alpha/dfs/data</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
</configuration>
```

```
<name>dfs.datanode.max.transfer.threads</name>
<value>40960</value>
</property>
<property>
</property>
</property>
<property>
<name>dfs.client.socket-timeout</name>
<value>300000000</value>
</property>
<property>
<name>dfs.datanode.handler.count</name>
<value>60</value>
</property>
<property>
<name>ipc.client.connection.maxidletime</name>
<value>3600000</value>

</property>
<property>
<name>ipc.server.handler.queue.size</name>
<value>3300</value>

</property>

</property>
<property>
<name>ipc.client.connection</name>
<value>3</value>
</property>
<property>
<name>dfs.datanode.max.transfer.threads</name>
<value>40960</value>
</property>
<property>
</property>
</property>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.namenode.accesstime.precision</name>
<value>-1</value>
</property>
<property>
```

4. To configure the JVM, edit the */hdfs-install-directory/etc/hadoop/hadoop-env.sh*

This configures the memory usage of the primary and secondary namenodes and datanode. For example, on servers with 48GB memory, if HDFS and HAWQ are on two separate clusters, Pivotal recommends that the namenodes use 40GB (-Xmx40960m), while each datanode uses 6GB and with a stack size of 256KB (-Xmx6144m -Xss256k).

5. To verify that HDFS has started, run the following command sequence.

1. List the directory:

```
hadoop fs -ls /
```

2. Create a test directory:

```
hadoop fs -mkdir /test
```

3. Put a test file (*/path/file*) into the HDFS root directory:

```
hadoop fs -put /path/file /
```

4. Perform a get on */file* from HDFS to the current local file system directory:

```
hadoop fs -get /file ./
```

1.2 Installing HAWQ

This section contains the following procedures to help you install HAWQ:

- Install the HAWQ Binaries
- Creating the gpadmin User
- Setting the OS Parameters
- Editing the Configuration Files
- Ensuring that HDFS works

1.2.1 Install the HAWQ Binaries

There are two ways HAWQ is released, as an RPM and a binary tarball.

To Install the RPM Release

1. Log in to the master host as *root*:

```
$ su - root
```

2. Launch the installer using rpm. For example:

```
# rpm -ivh hawq-dev-dev.x86_64.rpm
```

The installer installs HAWQ to the default install path (*/usr/local/hawq-dev*), and creates the soft link */usr/local/hawq* for */usr/local/hawq-dev*.

3. Source the path file from your master host's HAWQ installation directory:

```
# source /usr/local/hawq/greenplum_path.sh
```

4. Create a file called *hostfile* that includes host names in your HAWQ system using segment hosts. Make sure there are no blank lines or extra spaces. For example, if you have a standby master and three segments per host, your file will look something like this:

```
smdw
sdw1
sdw2
sdw3
```

5. Perform the ssh key exchange by running the following command. This allows you to log in to all hosts as root user without a password prompt. Use the *hostfile* file you used for installation.

```
gpssh-exkeys -f hostfile
```

6. Run the following command to reference the *hostfile* file you just created and copy the HAWQ rpm file (*hawq-dev-dev.x86_64.rpm*) to all hosts:

```
gpscp -f hostfile
hawq-dev-dev.x86_64.rpm =:~/
```

7. Run the following command to install HAWQ to all hosts:

```
# gpssh -f hostfile -e "rpm -ivh hawq-dev-dev.x86_64.rpm"
```

To Install from a Binary Tarball

1. Log in to the master host as root.

```
# su - root
```


2. Copy the HAWQ tarball to the binary directory you want to install HAWQ, go to the binary directory and uncompress the tarball. For example:

```
# cp /path/to/hawq-dev-dev.tar.gz /usr/local
# cd /usr/local
# tar xf hawq-dev-dev.tar.gz
```

A HAWQ directory is generated.

3. Open the file `/usr/local/greenplum_path.sh` and edit the `GPHOME` parameter to set it to `/usr/local/hawq`.

```
GPHOME=/usr/local/hawq
```

4. Source the path file from your master host's HAWQ installation directory:

```
# source /usr/local/hawq/greenplum_path.sh
```

Create a file called `hostfile` that includes host names used in your HAWQ system in segment hosts format. Make sure there are no blank lines or extra spaces. For example, if you have a standby mnaster and three segments per host, your file will look something like this:

```
smdw
sdw1
sdw2
sdw3
```

5. Perform the `ssh` key exchange by running the following command. This allows you to log in to `all_hosts` as root `user` without a password prompt. Use the `all_hosts` file you used for installation:

```
# gpssh-exkeys -f all_hosts
```

6. Run the following commands to reference the hostfile file you just created and copy the HAWQ binary directory (`/usr/local/hawq-dev`) to all hosts:

```
# gpscp -r -f hostfile hawq-dev =:/usr/local/
# gpssh -f hostfile -e "ln -s /usr/local/hawq-dev /usr/local/hawq"
```

1.2.2 Creating the gpadmin User

1. Create the *gpadmin* user account on each host:

```
# gpssh -f all_hosts -e '/usr/sbin/useradd gpadmin'
# gpssh -f all_hosts -e 'echo -e "changeme\nchangeme" | passwd gpadmin'
```

2. Log in to the master host as *gpadmin*.

```
$ su - gpadmin
```

3. Source the path file from the HAWQ installation directory:

```
$ source /usr/local/hawq/greenplum_path.sh
```

4. Run the following command to do the *ssh* key exchange to enable you to log in to all hosts without a password prompt as *gpadmin* user. Use the *all_hosts* file you used for installation:

```
$ gpssh-exkeys -f all_hosts
```

Use the *gpssh* utility to add the above command line to the profile file. For example:

```
$ gpssh -f all_hosts -e "echo source /usr/local/hawq/greenplum_path.sh >> .bashrc"
```

5. Use the *gpssh* utility to confirm that the Greenplum software was installed on all hosts. Use the *all_hosts* file you used for installation. For example:

```
$ gpssh -f all_hosts -e "ls -l $GPHOME"
```

**Note:**

You may want to change the default configuration parameters in */usr/local/hawq/etc/hdfs-client.xml* for *libhdfs3*. See the topic, HAWQ Configuration Parameter Reference.

6. Log in to the master host as *root*.

```
$ su - root
```

1.2.3 Setting the OS Parameters

This topic describes the OS parameter options that you need to set up for the following:

- Linux
- RHEL
- Security Configuration
- XFS

Linux

**Note:**

Pivotal recommends that you do not set the *vm.overcommit_memory* parameter if you run HAWQ on small memory virtual machines. If you set this parameter you may encounter out of memory issues.

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

```
xfs_mount_options = rw,noatime,inode64,allocsize=16m
sysctl.kernel.shmmax = 500000000
sysctl.kernel.shmmni = 4096
sysctl.kernel.shmall = 4000000000
sysctl.kernel.sem = 250 512000 100 2048
sysctl.kernel.sysrq = 1
sysctl.kernel.core_uses_pid = 1
sysctl.kernel.msgmnb = 65536
sysctl.kernel.msgmax = 65536
sysctl.kernel.msgmni = 2048
sysctl.net.ipv4.tcp_syncookies = 0
sysctl.net.ipv4.ip_forward = 0
sysctl.net.ipv4.conf.default.accept_source_route = 0
sysctl.net.ipv4.tcp_tw_recycle = 1
sysctl.net.ipv4.tcp_max_syn_backlog = 200000
sysctl.net.ipv4.conf.all.arp_filter = 1
sysctl.net.ipv4.ip_local_port_range = 1025 65535
sysctl.net.core.netdev_max_backlog = 200000
sysctl.vm.overcommit_memory = 2
sysctl.fs.nr_open = 3000000
sysctl.kernel.threads-max = 798720
sysctl.kernel.pid_max = 798720
#increase network
sysctl.net.core.rmem_max = 2097152
sysctl.net.core.wmem_max = 2097152
```

RHEL

RHEL version 6.x platforms, the above parameters do not include the *sysctl.* prefix, as follows:

```
xfs_mount_options = rw,noatime,inode64,allocsize=16m
kernel.shmmax = 500000000
kernel.shmmni = 4096
kernel.shmall = 4000000000
kernel.sem = 250 512000 100 2048
kernel.sysrq = 1
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.msgmni = 2048
net.ipv4.tcp_syncookies = 0
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 = 200000
net.ipv4.conf.all.arp_filter = 1
net.ipv4.ip_local_port_range = 1025 65535
net.core.netdev_max_backlog = 200000
vm.overcommit_memory = 2
fs.nr_open = 3000000
kernel.threads-max = 798720
kernel.pid_max = 798720
# increase network
net.core.rmem_max=2097152
net.core.wmem_max=2097152
```

Security Configuration

Set the following parameters (in the exact sequence displayed in the example) in the `/etc/security/limits.conf` file after updating the `/etc/sysctl.conf`:

```
soft nfile 2900000
```

```
hard nfile 2900000
```

```
soft nproc 131072
```

```
hard nproc 131072
```

XFS

XFS is the preferred file system on Linux platforms for data storage. Pivotal recommends the following xfs mount options:

```
rw,noatime,inode64,allocsize=16m
```

See the Linux manual page (`man`) for the `mount` command for more information about using that command (`man mount` opens the man page).

The Linux disk I/O scheduler for disk access supports different policies, such as CFQ, AS, and deadline.

Pivotal recommends the following scheduler option:

deadline

To specify a scheduler, run the following:

```
# echo schedulename > /sys/block/devname/queue/scheduler
```

For example:

```
# echo deadline > /sys/block/sbd/queue/scheduler
```

Each disk device file should have a read-ahead (blockdev) value of 16384. To verify the read-ahead value of a disk device:

```
# /sbin/blockdev --getra devname
```

For example:

```
# /sbin/blockdev --getra /dev/sdb
```

To set blockdev (read-ahead) on a device:

```
# /sbin/blockdev --setra bytes devname
```

For example:

```
# /sbin/blockdev --setra 16385 /dev/sdb
```

Refer to the Linux manual page (man) for the blockdev command for more information about using that command (man blockdev opens the man page).

1.2.4 Editing the Configuration Files

Edit the /etc/hosts file and make sure that it includes the host names and all interface address names for every machine participating in your HAWQ system.

1. Run the following command to copy the /etc/sysctl.conf file and /etc/security/limits.conf file to the same location of all hosts:

```
# gpscp -f all_hosts /etc/sysctl.conf =:/etc
# gpscp -f all_hosts /etc/security/limits.conf =:/etc/security
```

**Note:**

You may need to configure other parameters(for example, scheduler configuration)on all hosts.

2. 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 /data/master
```

3. Change ownership of this directory to the gpadmin user. For example:

```
# chown -R gpadmin /data/master
```

4. Using gpssh, create the master data directory location on your standby master as well. For example:

```
# gpssh -h smdw -e 'mkdir /data/master'
# gpssh -h smdw -e 'chown -R gpadmin /data/master'
```

5. Create a file called seg_hosts. This file should have only one machine configured host name for each segment host. For example, if you have three segment hosts:

```
sdw1
sdw2
sdw3
```

6. Using gpssh, create the data directory locations on all segment hosts at once using the seg_hosts file you just created. For example:

```
# gpssh -f seg_hosts -e 'mkdir /data/primary'
# gpssh -f seg_hosts -e 'chown gpadmin /data/primary'
```

7. To use JBOD, create temporary directory locations for the master, standby, and all the segments. The following example uses two disks with the workfile names /data1/tmp and /data2/tmp.

```
# dirs="/data1/tmp /data2/tmp"
# mkdir $dirs# chown -R gpadmin $dirs.
# gpssh -h smdw -e "mkdir $dirs"
# gpssh -h smdw -e "chown -R gpadmin $dirs"
# gpssh -f seg_hosts -e "mkdir $dirs"
# gpssh -f seg_hosts -e "chown -R gpadmin $dirs"
```

8. Log in to the master host as gpadmin. Make a copy of the gpinitssystem_config file to use as a starting point. For example:

```
$ su - gpadmin$ cp
$GPHOME/docs/cli_help/gpconfigs/gpinitssystem_config
/home/gpadmin/gpconfigs/gpinitssystem_config
```

9. Open the file you just copied in a text editor. Set all of the required parameters according to your environment. A HAWQ system must contain a master instance and at least two segment instances (even if setting up a single node system). The DATA_DIRECTORY parameter is what determines how many segments per host will be created. Here is an example of the required parameters in the gpinitssystem_config file:

```
ARRAY_NAME="EMC GP-SQL"
SEG_PREFIX=gpseg
PORT_BASE=40000
declare -a TEMP_DIRECTORY=(/data1/tmp /data2/tmp)
declare -a DATA_DIRECTORY=(/data/primary /data/primary)
MASTER_HOSTNAME=mdw
MASTER_DIRECTORY=/data/master
MASTER_PORT=5432
TRUSTED_SHELL=ssh
CHECK_POINT_SEGMENT=8
ENCODING=UNICODE
DFS_NAME=hdfs
DFS_URL=mdw:9000/gpsql
```

1.2.5 Ensuring that HDFS works

Make sure that your hdfs is working and change the following parameters in the gpinitssystem_config:

```
DFS_NAME=hdfs
DFS_URL=namenode-host-name:8020/hawq
```

Save and close the file.

Run the following command referencing the path and file name of your initialization configuration file (gpinitssystem_config) and host file (seg_hosts). For example:

```
$ cd ~
$ gpinitssystem -c gpconfigs/gpinitssystem_config -h seg_hosts
```

For a fully redundant system (with a standby master and a spread mirror configuration) include the -s and -S options. For example:

```
$ gpinitssystem -c gpconfigs/gpinitssystem_config -h seg_hosts -s standby_master_hostname
```

The utility verifies your setup information and ensures that 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 prompts you to confirm your configuration. For example:

```
=> Continue with Greenplum creation? Yy/Nn
Press y to start the initialization.
```

Set the MASTER_DATA_DIRECTORY environment variable. For example, add the following line to the profile of the master host:

```
export MASTER_DATA_DIRECTORY=/data/master/gpseg-1
```

1.3 HAWQ on Secure HDFS

1.3.1 Requirements


- A secure HDFS installation
- HDFS on wire encryption (dfs.encrypt.data.transfer) **MUST** be set to false.
- A new un-initialized HAWQ instance or a stopped already initialized HAWQ instance that was previously running on non-secured HDFS

1.3.2 Preparation

1. If HAWQ is already initialized and running stop HAWQ using "service hawq stop" or "<HAWQ installation directory>/bin/gpstop".
2. Secure the HDFS cluster using the instructions provided in the *Pivotal HD Stack and Tool Reference Guide* or using available security tools.
3. Insure HDFS is running properly in secured mode.
4. Insure that the property dfs.encrypt.data.transfer is set to false in the hdfs-site.xml for your cluster.


1.3.3 Configuration

1. Generate a "postgres" principal and keytab file as shown below:

 The form of principal for the HAWQ master is `postgres@REALM`, where `postgres` is the default service name of HAWQ and `REALM` is the default realm in the cluster's Kerberos configuration. In the examples below we use `EXAMPLE.COM` for the `REALM` part; this should be replaced by your cluster's actual `REALM`.

```
kadmin: addprinc -randkey postgres@EXAMPLE.COM
kadmin: ktadd -k /etc/security/keytab/hawq.service.keytab postgres@EXAMPLE.COM
```

2. Move this keytab file to the appropriate keytab directory on the HAWQ master node (for example, `/etc/security/phd/keytab/`).
3. Set the ownership of the keytab file to `gpadmin:gpadmin` and the permissions to 400.
4. Refer to your `gpinitssystem_config` file (typically in `/etc/gphd/hawq/conf`) to determine your configured HAWQ HDFS data directory (typically `/hawq_data`). This will be the last part of the `DFS_URL` value. For example if `DFS_URL` is set to `centos61-2:8020/hawq_data` then your HAWQ HDFS data directory is `/hawq_data`.
5. Create (if required) the HAWQ HDFS data directory in HDFS, and assign ownership as `postgres:gpadmin` and permissions 755.

- 
- If HAWQ has already been initialized and the directory exists just modify the owner and permissions as shown.
 - You need to have HDFS super-user permissions to create or modify a directory in HDFS root. If necessary create an "hdfs" principal to accomplish this task.

6. Create in HDFS (if not present) the directory `/user/gpadmin` with ownership `gpadmin:gpadmin` and permissions 777.

7. Modify the `hdfs-client.xml` file (typically in `/usr/lib/gphd/hawq/etc`) on the master node and ALL segment server nodes by adding the following:

```
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
</property>

<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>HDFS_NAMENODE_PRINCIPAL</value>
</property>
```



- `hdfs-client.xml` is in <HAWQ installation directory>/etc, typically `/usr/lib/gphd/hawq/etc`.
- These property blocks should be in the file but commented out, if so uncomment and edit the values.
- `HDFS_NAMENODE_PRINCIPAL` should be value from your cluster's `hdfs-site.xml` file.
- Make sure the namenode principal value is correct.

8. Edit your `gpinitssystem_config` file (typically in `/etc/gphd/hawq/conf`) and add (or uncomment if they are present and commented out):

```
KERBEROS_KEYFILE=/path/to/keytab/file
ENABLE_SECURE_FILESYSTEM=on
```



- Make sure there is no space between the `key=value`; for example:
`ENABLE_SECURE_FILESYSTEM = on` will cause errors because there are spaces.
- Make sure the value of `KERBEROS_KEYFILE` is the full path of where you placed the `hawq.service.keytab` file on the master.

9. After you have completed all these steps you can start or initialize HAWQ:
 1. If HAWQ was already initialized on non-secured HDFS before this process, start it using "service hawq start" or "<HAWQ installation directory>/bin/gpstart".
 2. If HAWQ has not been initialized you may now initialize HAWQ.
10. Verify HAWQ is operating properly, if not see next section.

1.3.4 Troubleshooting

If initialization or start-up fails you can look into the `gpinitssystem` log output and the namenode logs to see if you can pinpoint the cause. Possible causes:

- Incorrect values in your `hdfs-client.xml`
- `hdfs-client.xml` not updated on master and all segment servers
- Unable to login with Kerberos; possible bad keytab or principal for "postgres"
 - Validate on master by doing: `kinit -k <keytab dir path>/hawq.service.keytab postgres@EXAMPLE.COM`
- Wrong HAWQ HDFS data directory or directory permissions: Check your `gpinitssystem_config` file and the `DFS_URL` value and the directory permissions.
- Unable to create the HAWQ HDFS data directory errors: insure that you have created the proper directory as specified in `gpinitssystem_config` and that the ownership and permissions are correct.

1.4 Running a Basic Query

You can run the create database query to test that HAWQ is running:

```
changl1-mbp:gpsh changl1$ psql -d postgres
psql (8.2.15)
Type "help" for help.
postgres=# create database tpch;
CREATE DATABASE
postgres=# \c tpch
You are now connected to database "tpch" as user "changl1".
tpch=# create table t (i int);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'i' as the Greenplum
Database data distribution key for this table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s)
chosen are the optimal data distribution key to minimize skew.
CREATE TABLE
tpch=# \timing
Timing is on.
tpch=# insert into t select generate_series(1,100);
INSERT 0 100
Time: 311.390 ms
tpch=# select count(*) from t;
count
-----
100
(1 row)
Time: 7.266 ms
```

1.5 Installing Cryptographic Functions for PostgreSQL

The `pgcrypto` package contains the cryptographic functions for PostgreSQL. It must be installed separately from the main installation.

**Note:**

Your HAWQ installation must be running, or the pgcrypto package will fail to install.

All the cryptographic functions run inside database server. That means that all the data and passwords move between pgcrypto and the client application in clear-text. This means you must:

- Connect locally or use SSL connections.
- Trust both system and database administrator.

The pgcrypto package can be installed either through automated or manual installation.

1.5.1 Upgrading from 1.1.3 to 1.1.4

The pgcrypto package is not distributed with 1.1.4 version of HAWQ. You will have to use the pgcrypto package from 1.1.3 distribution and reinstall pgcrypto after the upgrade is done. Follow the installation instructions below.

1.5.2 Automated Installation of pgcrypto

These installation instructions assume you have obtained a precompiled build of pgcrypto from Pivotal.

1. Extract files from the pgcrypto package. For example:

```
mkdir pgcrypto
mv pgcrypto.tgz pgcrypto
tar -xzf pgcrypto/pgcrypto.tgz
```

2. Run pgcrypto_install.sh script with the hostfile as argument. The hosts file must contain hostname of each segment, one per line.

```
cd pgcrypto
./pgcrypto_install.sh -f ~/hostfile
```

1.5.3 Uninstalling pgcrypto

To uninstall the pgcrypto objects, use uninstall_pgcrypto.sql.

**Note:**

This script does not remove dependent user created objects.

1.5.4 Installing PL/R

The following instructions assume that you have downloaded a precompiled build of PL/R from Pivotal.

**Note:**

Check the name of your plr package. If it is plr-1.1.4.0-5152.x86_64.tgz, download the latest version plr-1.1.4.0-5664.x86_64.tgz for HAWQ 1.1.4.0 from Pivotal. The new package contains the file plr.sql with the necessary PL/R helper functions.

1. Create a directory named plr and extract the archive in it.

```
mkdir plr
mv plr*.tgz plr
tar -xzf plr*.tgz -C plr
```

2. Run the plr_install.sh script with the hostfile as argument. The hosts file must contain hostname of each segment, one per line..

```
cd plr;
./plr_install.sh -f ~/hostfile
```

3. After the installation finishes successfully, restart the HAWQ instance.

```
source $GPHOME/greenplum_path.sh
gpstop -ar
```

4. Create the plr language. Connect to a database using PSQL and run the following SQL command.

```
psql -d template1 -c "CREATE LANGUAGE plr"
```

You are now ready to create new PLR functions. A library of convenient PLR functions may be found in \$GPHOME/share/postgresql/contrib/plr.sql. These functions may be installed by executing plr.sql, as follows:

```
psql -d template1 -f $GPHOME/share/postgresql/contrib/plr.sql
```

1.6 HAWQ Configuration Parameter Reference

Describes the configuration in the path \$HAWQ_install_path/etc/hdfs-client.xml.

Parameter	Description	Default value	Comments
ipc.client.connection.maxidletime	The idle timeout interval of a rpc channel, rpc channel will exit if timeout occurs.	10000 (ms)	
ipc.client.connect.max.retries	The max retry times when a rpc channel failed to connect to the server for any reason except timeout.	1	
ipc.client.connect.max.retries	The max retry times when a rpc channel failed to connect to the server for any reason except timeout.	1	
ipc.client.connect.max.retries.on.timeouts	The max retry times when a rpc channel failed to connect to the server since timeout.	1	
ipc.client.connect.timeout	The timeout interval for a rpc channel to connect to the server.	10000 (ms)	
ipc.client.write.timeout	The timeout interval for a rpc channel to write data into socket	10000 (ms)	
ipc.client.tcpnodela	To set rpc channel to tcpnodelay mode	true	
dfs.ConfigKey.type	Default checksum type, valid value is: CRC32, CRC32C, NULL	CRC32C	Must be set to the same value as the HDFS side.
dfs.bytes-per-ConfigKey	The size of chunk to calculate checksum.	512	
dfs.datanode.rpc.timeout	The timeout interval to the client to wait for the datanode to finish a rpc call.	3	
dfs.namenode.rpc.timeout	The timeout interval to the client to wait for the namenode to finish a rpc call.	3600000	
dfs.client-write-packet-size	The packet size for the output stream.	65536	
dfs.client.block.write.retries	The packet size for the output stream.	3	

dfs.client.close.file.timeout	The timeout interval to the output stream to wait for close file operation completion.	3600000 (ms)	
dfs.client.block.write.timeout	Time of timeout interval to the output stream to write data into socket.	3600000 (ms)	
dfs.prefetchsize	The number of blocks which metadatas will be prefetched.	10	
dfs.client_local_block_read_buffer	The buffer size if read block from local file system instead of network.	1048576	
dfs.client.socket.write.timeout	The timeout interval for socket write operation.	3600000 (ms)	
dfs.client.socket.read.timeout	The timeout interval for socket write operation.	3600000 (ms)	
dfs.client.socket.connect.timeout	The timeout interval to setup a tcp socket connection, include resolve host name.	3600000 (ms)	
dfs.client.read.verifychecksum	Input stream will verify checksum.	true	
dfs.client.enable.read.from.local	Enable input stream to read block directly from local file system. Need to configure on server.	true	
dfs.log.file.prefix	The libhdfs3 log file prefix, see glog document.	libhdfs3	
dfs.log.stderr	Output libhdfs3 log to stderr.	true	
dfs.replication	The default number of replica.	3	
dfs.blocksize	The default block size, can be overwritten when create output stream.	67108864	