How to Build, Install, and Test SciDB 14.3

Paradigm4, Inc.
2014-4-8

1. INTRODUCTION

This document provides guidelines how SciDB developers can get SciDB source code and build/install/test. It is fairly extensive:

- It covers both external users (who get source code from the forum) and internal users (Paradigm4 employees).
- It covers both the Community Edition (CE) and the Enterprise Edition (EE) of SciDB code.
- It covers both the *local development* case of developing on a single machine (host), and the *cluster development* case of developing on a cluster of multiple hosts.
- It covers both Ubuntu and CentOS.

Some assumptions:

- You have a cluster of 1 or more hosts. The document assumes the names of the hosts are <*hostIP0*>, <*hostIP1*>, etc., but other names are fine.
- All the hosts have a *scidb* user, which belongs to the *scidb* group.
- You have passwords of both the *root* user and the *scidb* user on all the hosts.
- You will use <*hostIP0*> as both the build host and the *coordinator host*, i.e. where the SciDB coordinator instance will be started on. (See Section 7 for a note about using separate build and coordinator hosts.)
- Unless otherwise stated, you will issue all the commands below while logging on to <*hostIP0*> as the *scidb* user.
- Without lose of generality, on <*hostIP0*> assume the CE version and EE version source code will be placed into <dev_dir>/scidbtrunk and <dev_dir>/p4trunk, respectively.
- The <dev_dir> directory is owned by the *scidb* user.

2. BOOTSTRAPPING

2.1. Install svn, expect, and ssh packages

with (CentOS):

```
sudo yum install subversion expect openssh-server openssh
```

or (Ubuntu):

```
sudo apt-get install subversion expect openssh-server openssh-client
```

2.2. Configure and start the SSH server

with (CentOS):

```
sudo chkconfig --add sshd
sudo chkconfig sshd on
sudo service sshd start
```

or (Ubuntu):

```
sudo service ssh restart
```

2.3. Open the SciDB port and SSHD port

Make sure the following lines exist in /etc/sysconfig/iptables:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 1239 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

Then run:

```
service iptables restart
```

Note: subversion is only needed for internal users.

Note: SSH is needed on all the hosts.

3. GET SOURCE CODE

For external developers:

- Download the source code tarball from ?http://www.scidb.org/forum/viewtopic.php?f=16&t=364
- cd <dev_dir>; tar -xvfz <the_tarball>
- Rename the source-code directory name to *scidbtrunk*. (Optional; but the steps below assumes that.)

For internal developers:

```
cd <dev_dir>
svn co http://svn.scidb.net/scidb/trunk scidbtrunk
```

ADDITIONAL FOR ENTERPRISE EDITION:

```
cd <dev_dir>
svn co http://svn.scidb.net/p4/trunk p4trunk
```

4. PREPARE TOOLCHAIN

4.1. Configure passwordless SSH to all the cluster nodes

We require that the scidb user have passwordless ssh to all nodes, logging in on each node as the scidb user, for starting and stopping SciDB.
We require that the scidb user have passwordless ssh to all nodes, logging in on each node as the root user, for initializing SciDB on each node.

```
ssh-keygen # DO NOT enter a passphrase; accept the default.
cd <dev_dir>/scidbtrunk
deployment/deploy.sh access root  "" "" <hostIP0> <hostIP1>
deployment/deploy.sh access scidb  "" "" <hostIP0> <hostIP1>
```

Note: in the above *deploy access* commands, if you have a single host, omit <*hostIP1*>; if you have more than two hosts, list all of them. The rule applies to all subsequent commands in which <*hostIP1*> is mentioned.

YOU SHOULD CHECK that as user 'scidb' on <hostIP0>, you are able to ssh to all hosts WITHOUT BEING ASKED FOR A PASSWORD:

```
ssh <hostIP0> date
ssh <hostIP1> date
```

Note: If passwordless SSH does not work, you should fix it before proceeding, to prevent unexpected errors in future steps. See the troubleshooting section in the user manual.

4.2. Remove older versions of SciDB, if any

Remove scidb.list.* and p4.list.* from /etc/yum.repos.d (CentOS) or /etc/apt/sources.list.d (Ubuntu).

Then (CentOS):

```
yum list | grep scidb | awk '{print $1}' | grep '[a-zA-Z]' | xargs sudo rpm -e
```

or (Ubuntu):

```
# Ubuntu
dpkg --list | grep scidb | awk '{print $2}' | xargs sudo dpkg --purge
```

4.3. Toolchain

Install the various third party libraries, scripts, tools, debuggers, compilers, and so on, that are needed to build SciDB:

```
cd <dev_dir>/scidbtrunk
deployment/deploy.sh prepare_toolchain <hostIP0>
```

ADDITIONAL FOR ENTERPRISE EDITION:

```
cd <dev_dir>/p4trunk
deployment/deploy.sh prepare_toolchain <hostIP0>
```

4.4. Chroot (only in cluster development)

```
cd <dev_dir>/scidbtrunk
deployment/deploy.sh prepare_chroot scidb <hostIP0>
```

4.5 On CentOS, fix /etc/hosts

The default `/etc/hosts` file on CentOS 6.5 (and earlier?) contains bad IPv6 aliases for localhost that must be removed before SciDB will start successfully. A good `/etc/hosts` file for workstation "frobnitz" might look like this:

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost6 localhost6.localdomain6
10.0.20.42   frobnitz frobnitz.localdomain frobnitz.local.paradigm4.com
```

5. PREPARE RUNTIME

5.1. Install Postgres

SciDB uses a PostgreSQL database to store metadata describing its arrays etc. So, we would configure a PostgreSQL instance as follows:

```
cd <dev_dir>/scidbtrunk
deployment/deploy.sh prepare_postgresql postgres postgres 10.0.20.0/24 <hostIP0>
```

This will install the software and configure the PostgreSQL instance under username *postgres* with password *postgres* to accept client network connections from network 10.0.20.0/24. Change the network mask based on your environment.

5.2 Enable the *postgres* user to access scidb source code

```
sudo usermod -G scidb -a postgres
chmod g+rx <dev_dir>
```

Failure to complete this step may result in an error during *run.py install*, where the postgres user cannot read / execute the scidb.py script.

YOU SHOULD CHECK:

```
sudo -u postgres ls <dev_dir>  # the 'postgres' user should have access to dev_dir.
```

If the above step generates a 'permission denied' error, you should check the permission of not only <dev_dir>, but its ancestor directories. A common misconfiguration is to have <dev_dir> inside the home directory of the 'scidb' user, while /home/scidb has 700 access mode.

6. BUILD AND INSTALL

6.1. Environment Variables

Add the following lines to your local shell configuration script (e.g `.bashrc`), then source it.

Local development

```
export SCIDB_VER=14.3
export SCIDB_INSTALL_PATH=<dev_dir>/scidbtrunk/stage/install
export PATH=$SCIDB_INSTALL_PATH/bin:$PATH
```

Note: For local development, you have the flexibility of pointing SCIDB_INSTALL_PATH at an arbitrary location (the default is <dev_dir>/scidbtrunk/stage/install), but it must be different from /opt/scidb/$SCIDB_VER. The reason is as follows. A subsequent local-development step, `./run.py install`, will wipe out all content in SCIDB_INSTALL_PATH. But /opt/scidb/$SCIDB_VER cannot be wiped out, because it contains useful 3rd-party packages (that was installed there using a previous prepare_toolchain step).

Note: Setting the PATH environmental variable is optional, but it is helpful to find utilities such as *iquery*.

Cluster development

```
export SCIDB_VER=14.3
export SCIDB_INSTALL_PATH=/opt/scidb/$SCIDB_VER
export PATH=$SCIDB_INSTALL_PATH/bin:$PATH
export SCIDB_BUILD_TYPE=RelWithDebInfo
```

Note: For cluster development you must provide SCIDB_INSTALL_PATH and it must point to /opt/scidb/$SCIDB_VER.

Note: SCIDB_BUILD_TYPE is used in *run.py setup* and *run.py plugin_setup*. The default value is Debug. For performance measurement, use RelWithDebInfo. See *run.py setup -h*.

6.2. Build

```
cd <dev_dir>/scidbtrunk
./run.py -h   # to learn its usage.
./run.py setup    # to configure build directories and cmake infrastructure
./run.py make -j4 # to build the sources.
```

Note: the above setup and make commands are shared both when you plan to install on the local host and when you plan to install to a cluster. Once again, if the intention is to install to a cluster, the environmental variable SCIDB_INSTALL_PATH must point to /opt/scidb/$SCIDB_VER.

Note: In every sub-command, you can use '-h' to get more info. E.g. you can say *./run.py setup -h*

Note: In the 'make' sub-command, use '-j4' to speedup build with 4 threads; use 'doc' to build doxygen comments (ie `./run.py make -j2 doc`

> and the doxygen HTML pages will be placed in trunk/stage/build/doc/api/html); use 'user_doc' to build user manual (stage/build/doc/user/pdf).

Note: run.py supports various additional commands and parameters including build and install directories, build type, SciDB configuration, and package generation. The chances are it supports what you need. For instance, 'run.py cleanup' will revert to the state before 'run.py setup' was called.

ADDITIONAL FOR ENTERPRISE EDITION:

```
./run.py plugin_setup -n p4 --path <dev_dir>/p4trunk
```

```
./run.py plugin_make -j4 -n p4  # to build the P4 sources
```

Hint: for faster builds, using an SSD or ramdisk may be helpful. Here is an example of using ramdisk to do *run.py setup*:

```
cd <dev_dir>/scidbtrunk
mkdir BUILD
sudo echo "tmpfs <dev_dir>/scidbtrunk/BUILD tmpfs size=8G,nr_inodes=80k" >> /etc/fstab
sudo mount -a
SCIDB_BUILD_PATH=<dev_dir>/scidbtrunk/BUILD ./run.py setup
```

6.3. Local development

```
./run.py install
```

This creates a SciDB installation with a default configuration. If the environmental variable SCIDB_INSTALL_PATH is provided, SciDB is installed there. Otherwise, SciDB is installed to ./stage/install.

ADDITIONAL FOR ENTERPRISE EDITION:

```
./run.py plugin_install -n p4   # to install P4 libraries into the SciDB installation.
```

6.4. Cluster development

```
./run.py make_packages /tmp/packages
deployment/deploy.sh scidb_install /tmp/packages <hostIP0> <hostIP1>
deployment/deploy.sh scidb_prepare scidb "" mydb mydb mydb  # continue to next line
    /home/scidb/mydb-DB 2 default 1 default <hostIP0> <hostIP1>
```

Note: full path is needed in the make_packages sub-command.

Note: you may omit the scidb_prepare sub-command, if the catalog was already generated beforehand.

In the scidb_prepare sub-command above, the parameters are (left to right):

- **scidb**: the 'scidb' user
- **""**: the optional password for the *scidb* user
- three occurrences of **mydb**: Postgres user, Postgres password, and database name
- **/home/scidb/mydb-DB**: the base-path, i.e. where database files will be stored
- **2**: two SciDB instances per host
- **default**: use the default option of no-watchdog=false
- **1**: redundancy=1
- **default**: use default value for the chunk-segment-size parameter

ADDITIONAL FOR ENTERPRISE EDITION:

```
./run.py plugin_make_packages -n p4 /tmp/p4packages
deployment/deploy.sh scidb_install /tmp/p4packages <hostIP0> <hostIP1>
```

Note: currently *plugin_make_packages* is insensitive to SCIDB_INSTALL_PATH; it always generates

packages installable into /opt/scidb/<CURRENT_SCIDB_VERSION>

If you ever need to remove the packages:

```
cd <dev_dir>/scidbtrunk
deployment/deploy.sh scidb_remove /tmp/packages <hostIP0> <hostIP1>
deployment/deploy.sh scidb_remove /tmp/p4packages <hostIP0> <hostIP1>  # for EE only
```

Note: the *scidb_remove* sub-command only removes the installed packages that can be found in the directory specified at command line.

7. START OR STOP SCIDB

7.1. Official way (works both for local development and cluster development)

```
scidb.py startall mydb  # start the 'mydb' database
scidb.py stopall mydb   # stop the 'mydb' database
```

7.2. Alternative way (in local development)

```
cd <dev_dir>/scidbtrunk
./run.py start   # start
./run.py stop    # stop
```

Note: the benefit of the above alternative is that fewer bytes need to be typed (database name is not needed).

7.3. Alternative way (in cluster development)

```
cd <dev_dir>/scidbtrunk
deployment/deploy.sh scidb_start scidb mydb <hostIP0>  # start
deployment/deploy.sh scidb_stop scidb mydb <hostIP0>   # stop
```

Note: the benefit of the above alternative is that the command may be issued from a host different from the coordinator host. The majority of the document did not mention this use case but you could do it. Essentially what this option requires is that you run the *deploy access* steps both on the build host and on the coordinator host.

8. HARNESS TEST (ONLY FOR INTERNAL USERS)

Note: external users may also run these tests (at own risk), but some tests will fail because they expect to find some data files in a certain /public directory.

8.1. How to run harness tests

After SciDB is started:

```
cd <dev_dir>/scidbtrunk
./run.py tests
```

Note: the CE test results are in *stage/build/tests/harness/testcases*. E.g. see Report.xml there for an easy-to-browse report of the test results.

Note: by default only the checkin suite is run. Use '-h' to see how to specify more tests or less tests

ADDITIONAL FOR ENTERPRISE EDITION:

```
./run.py plugin_tests -n p4  # to run P4 checkin tests
```

Note: The EE test results are in *stage/build/external_plugins/p4/test/testcases*.

8.2. How to run harness test on someone else's already installed SciDB cluster

Suppose you got access to a SciDB cluster from someone else, you do not need to build or install SciDB, but you want to run harness test. Below are the steps to follow.

- Log on to the coordinator host.
- Get source code (see Step 3).
- Set environmental variable:
    - SCIDB_INSTALL_PATH: should be like /opt/scidb/14.3; needed for all steps below.
    - SCIDB_NAME, SCIDB_PORT, SCIDB_HOST, SCIDB_DB_USER, SCIDB_DB_PASSWD: should match the values specified in config.ini; needed only for the *tests* and *plugin_tests* sub-commands.
- Execute the following commands (note that build or install are not specified):

```
cd <dev_dir>/scidbtrunk
rm -rf ./stage
./run.py setup
./run.py plugin_setup -n p4 -p <dev_dir>/p4trunk
./run.py tests
./run.py plugin_tests -n p4
```