SciDB Install Guide

SciDB is currently available for Ubuntu 9.10, Ubuntu 10.04 and Ubuntu 10.10. You can also install SciDB from sources.

For those using Ubuntu on VirtualBox, if you want to be able to have your Ubuntu window in full screen mode, you must install the GuestAdditions package and restart your VirtualBox.

Install SciDB

Install SciDB from sources. By default SciDB is installed in /opt-0.7.5.

```
tar xvfz scidb-0.7.5.1557.tgz
cd scidb-0.7.5.1557

sudo apt-get update

sudo apt-get install -y build-essential cmake libboost1.40-all-dev \
postgresql-8.4 libpqxx-3.0 libpqxx3-dev libprotobuf6 libprotobuf-dev \
protobuf-compiler doxygen flex bison libxerces-c-dev libxerces-c3.1 \
liblog4cxx10 liblog4cxx10-dev libcppunit-1.12-1 libcppunit-dev \
libbz2-dev postgresql-contrib libconfig++8 libconfig++8-dev \
libconfig8-dev subversion

cmake .
make
sudo make install
```

If you are installing a downloaded pre-build binary package for example, `scidb-RelWithDebInfo-0.7.5.1557-Ubuntu-9.10-amd64.deb,` you can install it using dpkg. We currently provide packages for Ubuntu 9.10, Ubuntu 10.04, and Ubuntu 10.10.

```
sudo dpkg -i scidb-0.7.5-....deb
```

dpkg does not resolve dependencies, you'll need to manually install the dependencies or else use apt-get to resolve any unmet dependencies on the system.

```
sudo dpkg -i scidb-0.7.5-....deb # Fails due to unmet dependencies
apt-get -f install # installs all dependencies
sudo dpkg -i scidb-0.7.5-....deb # Succeeds now
```

Uninstall the package using the package name scidb-0.7.5:

```
sudo dpkg -r scidb-0.7.5
```

Add the following to your path.

```
export PATH=/opt/scidb-0.7.5/bin:/opt/scidb-0.7.5/share/scidb:$PATH
export LD_LIBRARY_PATH=/opt/scidb-0.7.5/lib:$LD_LIBRARY_PATH
```

Configure SciDB

Catalog configuration

Correctly configuring SciDB requires setting up a PostgresSQL instance, and making a database in it to hold the SciDB catalog. Get sudo privileges for your account. sudo privileges are required for creating a postgres account for SciDB and for initializing the catalog database.

Check that postgres is running on the coordinator.

```
sudo /etc/init.d/postgresql-8.4 status
sudo /etc/init.d/postgresql-8.4 start
```

Depending on your installation of postgres, you may have to specify postgresql with a different version number, or without the version number.

If you cannot obtain sudo privileges for the *scidb* account ask your system administrator to run this script as the 'postgres' user:

```
/opt/scidb-0.7.5/bin/scidb-prepare-db.sh
```

This script is used to create a new role or account (say *scidb_user1*) with password (say *scidb_passwd1*) and a database for testing scidb (say *test1*).

SciDB config.ini

Edit the /opt/scidb-0.7.5/etc/config.ini for your environment. The configuration 'test1' below is an example of a single node config. The config file can have multiple sections, one per service instance.

```
[test1]
master_ip=10.248.211.239
db_user=scidb_user1
db_passwd=scidb_passwd1
install_root=/opt/scidb-0.7.5
metadata=/opt/scidb-0.7.5/share/scidb/meta.sql
pluginsdir=/opt/scidb-0.7.5/lib/scidb/plugins
logconf=/opt/scidb-0.7.5/share/scidb/log4cxx.properties
master_data_dir=/mnt/master
master_port=1239
interface=eth1
```

The following table describes the config file contents and how to set them. The install package contains two sample config files. Config.ini is a sample config file for a 2-node cluster. The config.ini.planet is for the Planet cluster used for SciDB testing.

Key	Value
cluster name	Name of the SciDB cluster which must appear as a section heading in the config.ini file, e.g., *[cluster1]*
master_ip	IP address used by the master SciDB process.
db_user	Username to use in the catalog connection string

db_passwd	Password to use in the catalog connection string
install_root	Path name of install root. Must be the same on master and worker nodes, in this wiki we explain how to configure this using NFS.
metadata	Metadata definition file. The recommended NFS configuration makes this visible under the same path name on master and worker processes.
pluginsdir	Plugins folder - location of all plugins. Must be visible under the same path name to all workers.
logconf	Config file for SciDB log. Edit this to set a different filename or log level (default file name is *INFO* and default file name is scidb.log.
master_data_dir	Data directory for each SciDB master node.
master_port	port number on the master for clients.
interface	Ethernet interface used for SciDB used on all nodes - master and workers. Used to bind SciDB to the correct local interface.

In our example shown in the example config.ini file, the db_user field is set to *scidb_user1* and db_passwd is set to *scidb_passwd1*. The postgres database created is the same as the section header, *test1*.

Additionally, the config file also supports the following key-value combinations for specifying worker nodes in the cluster. We describe these key-value combinations in this document, but show their usage in the SciDB Cluster Install Guide.

Key	Value
worker_ip	Comma-separated list of IP addresses, one per SciDB worker process.
worker_data_dir1	Data directory for each SciDB worker node.
worker_data_dir2	Not yet supported. Data directory for second instance on each worker node.

A single config.ini may contain multiple SciDB services, each defined within a separate section. However, each service must use distinct values for cluster name, pluginsdir, master_data_dir, worker_data_dir{1,2}, and master_port.

Launch SciDB

Use the init scripts to launch SciDB. The service is referred to using its name, the section header in the config file.

```
scidb-0.7.5 init test1
scidb-0.7.5 start test1
scidb-0.7.5 status test1
scidb-0.7.5 stop test1
```

SciDB logs are written to the file scidb.log in the master_data_dir on the master node, or on the worker_data_dir on the workers.

iquery client

iquery is the default SciDB client used to issue AQL and AFL commands. iquery connects by default to SciDB on port 1239. If you use a non-default port number, specify it using the "-p" option with iquery.

```
  iquery -aq "list('arrays')"
```

The *iquery* executable is the basic command line tool we use. At the moment, it tries to make a connection on the local node to a standard port, where the scidb engine ought to be listening. Each invocation of iquery connects to the SciDB coordinator node, passes in a query, and prints out the coordinator node's response.

Example SciDB session

Before reviewing the complete AFL and AQL documentation it is useful to review this example SciDB session. Note that it's convenient to run the iquery tool in a new terminal window.

Basic Data Definition Language (DDL) and Data Loads

SciDB uses a language we call 'AQL' for 'Array Query Language'. In AQL, the basic structural motif is the array. To create an array in SciDB, you would use the following commands:

```
$ iquery -q "CREATE ARRAY  test <a: int32, b: int32 > [x=0:2,3,0, y=0:2,3,0]"
Query was executed successfully
```

This creates an array called ***test*** with two attributes named ***a*** and ***b*** of type int32. The new array has rank (number of dimensions) 2 and the two dimensions or indices are named ***x*** and ***y***. These dimensions range from 0:2, have chunks (physical storage size) of 3 in both dimensions, and each chunk has 0 overlap with its neighbors in both dimensions.

At the moment, we only support bulk loading data into SciDB. We have two external formats: one suitable for dense arrays (cases where the vast majority of the elements in the array contain actual values) and one designed for dense arrays (cases where many of the elements are empty, or have some 'default' value.) The idea is that you create and populate file(s) with ASCII data that comply with the rules of one of these representations and then direct SciDB to load data from those file(s).

Dense arrays have the following format:

```
[
[(0,0),(0,1),(0,2)],
[(1,0),(1,1),(1,2)],
[(2,0),(2,1),(2,2)]
]
```

And sparse arrays look like this ...

```
[[
{0,0} (0,0),
{0.1} (0,1),
{0,2} (0,2),
{1,0} (1,0),
{1.1} (1,1),
{1,2} (1,2),
{2,0} (2,0),
{2.1} (2,1),
{2,2} (2,2)
]]
```

In the dense format the dimension values are implicit in the ordering of the attributes. In the sparse format, the dimension indices are explicitly enumerated in the load file. Note also that, in the sparse format, the array is broken up into chunk-sized sections. In this example, as there is only one chunk for the entire array, there is only one block. The following example illustrates how the load file for a very sparse array might look.

```
[[
{5,16} (0.497321)
]]
;
[[
{2,132} (0.944702)
]]
;
[[
{0,244} (0.657221)
,
{53,255} (0.609632)
,
{68,226} (0.448509)
]]
[[
{21,451} (0.767433)
,
{44,427} (0.613046)
]]
```

Assuming that you have placed data in one of these formats into a file named '/tmp/data.txt', to load the data from one of these files into the new array, use the *load* command as shown below.

```
$ iquery -aq "load('test','/tmp/data.txt')"
[[]]
```

Now you have a small array, loaded with data.

Running AQL and AFL Queries

You also use *iquery* to run queries that retrieve and manipulate array data. By default iquery accepts AQL statements. Use the "-a" switch to issue AFL queries to SciDB.

```
$ iquery -aq "subsample(scan('test'), 0, 0, 2, 2)"
```

The result will be printed to atdout. Use --result[-r] option to point new filename for result file. You also can connect to remote SciDB coordinator node by using additional connection information. For complete details, see iquery -h.

End

Congratulations! You're done.