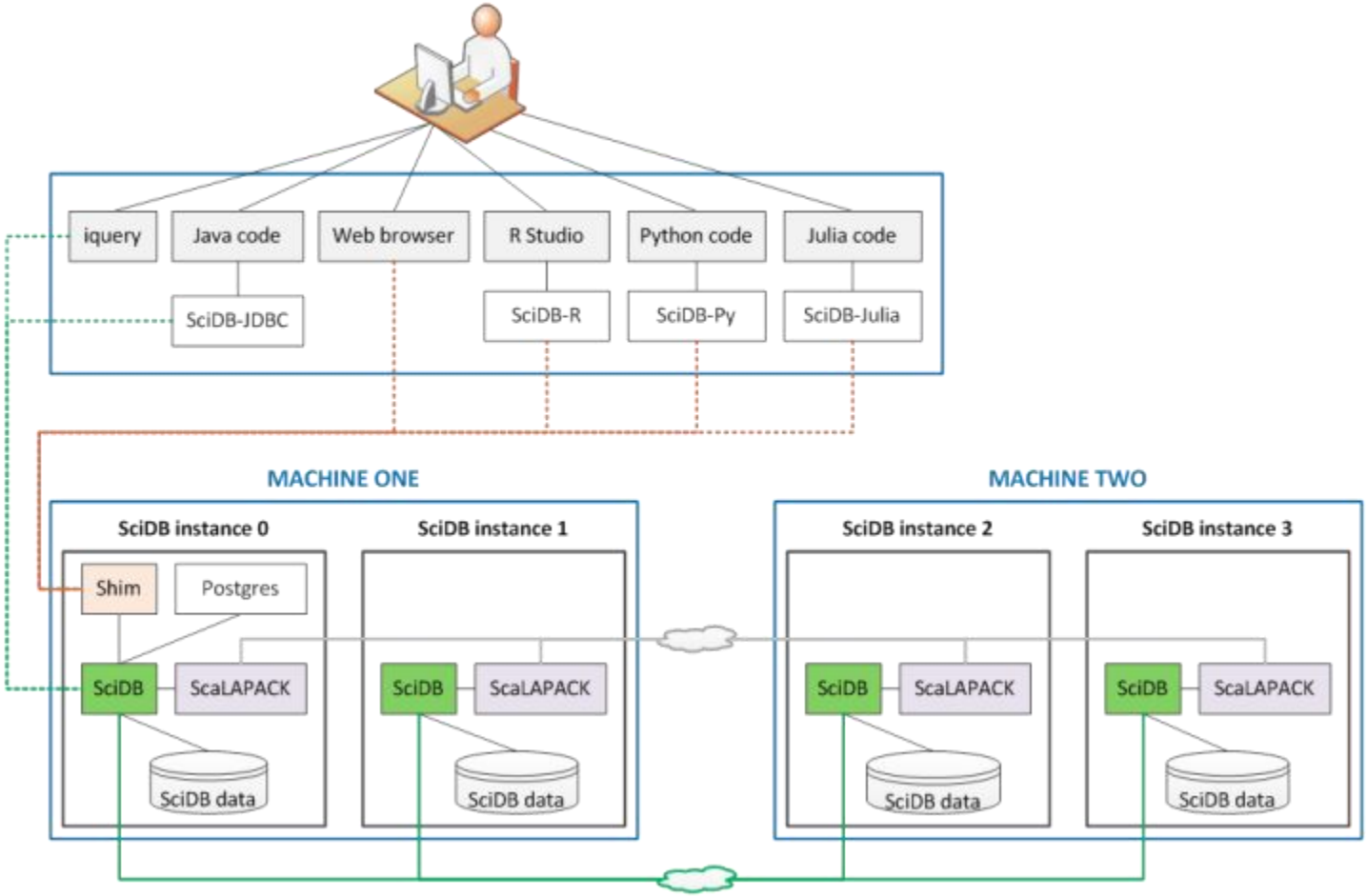
SciDB

AQL & AFL

Referata GeoInformática - INPE SJC
2015-06-09

presented by: Alber Sánchez alber.ipia@inpe.br

Architecture



Source: https://trac.scidb.net/attachment/wiki/public/Architecture/architecture.png

SciDB is an array database for multidimensional data management and analytics (Wikipedia)

Installation

Alternative 1: Paradigm4

wget https://downloads.paradigm4.com/scidb-14.12-repository.deb

dpkg -i scidb-14.12-repository.deb

apt-get update

apt-get install scidb-14.12-installer

Alternative 2: Install docker

```
sudo apt-get update

sudo apt-get install wget

wget -qO- https://get.docker.com/ | sh

sudo docker run hello-world
```

Alternative 2: SciDB on docker

```
git clone https://github.com/albhasan/docker_scidb.git
cd docker_scidb
sed -i 's/localhost,7/localhost,1/g' scidb_docker.ini
./setup.sh
```

```
ssh -p 49901 root@localhost # xxxx.xxxx.xxxx
cd /home/root
./containerSetup.sh
su scidb
cd ~
```

It uses a docker container running Ubuntu 12.04 (LTS)

Alternative 3: Compile

http://paradigm4.com/forum/viewtopic.php?f=16&t=1596

https://github.com/albhasan/docker_scidb

iquery client

http://www.paradigm4.com/HTMLmanual/14.12/scidb_ug/ch02s01.html

Non-interactive mode

iquery -q "my AQL statement"

iquery -f my_input_filename

iquery -aq "my AFL statement"

Examples:

iquery -q "SELECT * FROM list('instances');"

iquery -aq "list('instances');"

iquery output options

Output Option	Description
auto	SciDB array format.
csv	Comma-separated values.
csv+	Comma-separated values with dimension indices.
lcsv+	Comma-separated values with dimension indices and a Boolean flag attribute, **EmptyTag**, showing if a cell is empty.
dcsv (default)	Format used in most doc examples. Visually distinguishes dimensions from attributes. This is the default output format.
tsv	Tab-separated values.
tsv+	Tab-separated values with dimension indices.
ltsv+	Tab-separated values with dimension indices and a Boolean flag attribute, **EmptyTag**, showing if a cell is empty.
dense	Ideal for viewing 2-dimensional arrays. Displays empty cells as parentheses. Not recommended for very sparse arrays.
sparse	Sparse SciDB array format.

iquery -o csv -r output_file.csv -af input_file

Interactive mode

iquery
SELECT * FROM list('instances');
set no fetch;
SELECT * FROM list('instances');
set fetch;
set lang afl;
list('queries');
cancel(1100945697403);
exit;

Interactive mode

iquery

```
CREATE ARRAY A
    <valA:double>
    [row=0:1,2,0, col=0:2,3,0];
INSERT INTO A '[[(1) (0) (0)] [(2) (-3) (1)]]';
SELECT * FROM list('arrays');
SELECT * FROM show(A);
SELECT * FROM scan(A);
DROP ARRAY A;
exit;
```

Ctrl + C for the win!

Create array

SciDB Array

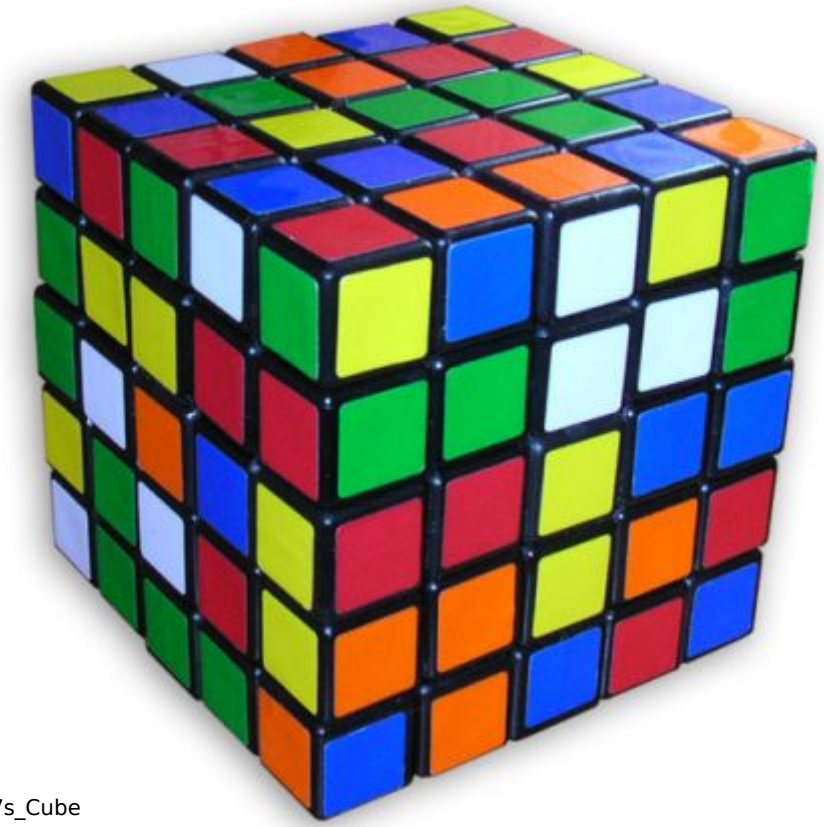
```
CREATE ARRAY Simple_Array
< v1:   double,
v2 :    int64,
v3 :    string >
[I = 0:*, 5, 0,
J = 0:9, 5, 0];
```

Attributes	Dim	Dim size	Chunk	Chunk
v1, v2, v3	I, J	* is unbounded	size	overlap

Source: http://paradigm4.com/HTMLmanual/13.3/scidb_ug/ch01s02.html

Attributes, dimensions and chunks.

SciDB Chunks



Source: http://en.Wikipedia.org/wiki/Professor%27s_Cube

Large arrays are split into chunks
which are distributed among instances

Insert data

AQL - 1D

```
CREATE ARRAY GEODEF <aname:string,
   key:string, value:string>[i=0:*,10,0];


INSERT INTO GEODEF
   '[(TRMM_3B43_SALESKA, invertValuex, 1),
   (TRMM_3B43_SALESKA, invertValuey, -1),
   (TRMM_3B43_SALESKA, lengthx, 0.25)]';


DROP ARRAY GEODEF;
```

AQL - 2D

```
CREATE ARRAY GT_coods
   <lonWGS84deg:double, latWGS84deg:double>
   [col_id=0:2,10,0, row_id=0:1,10,0];

INSERT INTO GT_coods
   '[[(-179.875,49.875) (179.875,49.875)]
   [(89.875,49.875) (-0.125,-49.875)]
   [(0,0) (180,90)]]';

DROP ARRAY GT_coods;
```

Random array

AFL

```
set lang afl;
store(
   build(
        <val:double NULL DEFAULT null>
        [i=0:9,9,0,j=0:9,9,0],
        random() % 2),
   COMWAY
);
remove(COMWAY);
```

Load & save data from files

SciDB array text format

```
[
[(0,100),(1,99),(2,98),(3,97)],
[(4,0),(5,95),(6,94),(7,93)],
[(8,92),(9,91),(),(11,89)],
[(12,88),(13,0),(14,86),(15,85)]
]
```

File m4x4.txt

Load SciDB array text

```
set lang afl;

create array m4x4
   <val1:int32, val2:int32>
   [i=0:3,4,0, j=0:3,4,0];

load(m4x4,'/home/scidb/m4x4.txt');
```

Save to file

```
save(
   m4x4,
   '/home/scidb/m4x4.csv',
   -2,
   'csv'
);

save(m4x4,'/home/scidb/m4x4.csv+',-
   2,'csv+');
```

csv+ includes the dimensions in the output file.

Load csv - Step 1

```
-- sed -n '2,$p' /home/scidb/m4x4.csv+ >
   /home/scidb/newm4x4.csv+

set lang afl;

create array m4x4flat <i:int64, j:int64, val1:int32,
   val2:int32> [dim=0:*,100,0];

load(m4x4flat , '/home/scidb/newm4x4.csv+', -2,
   'csv');
```

Remember to remove the header from the CSV file!

Load csv - Step 2

```
set lang afl;

create array newm4x4 <val1:int32,
   val2:int32> [i=0:3,4,0, j=0:3,4,0];

redimension(m4x4flat, newm4x4);
```

```
create array newm4x4<val1:int32,val2:int32>[i=0:3,?,0,j=0:3,?,0] using
m4x4flat ;
```

Load MODIS data

Install parallel

```
mkdir ~/install_parallel
cd ~/install_parallel
wget http://ftp.gnu.org/gnu/parallel/parallel-
    20140922.tar.bz2
tar -xvjf parallel*
cd parallel*
#less README
./configure
make
sudo make install
cd ~
```

Install boost

```
export LC_ALL="en_US.UTF-8"

sudo apt-get install build-essential g++ python-dev autotools-dev gfortran
    libicu-dev build-essential libbz2-dev libzip-dev wget

mkdir installBoost
cd installBoost

wget -O boost_1_57_0.tar.gz
    http://sourceforge.net/projects/boost/files/boost/1.57.0/boost_1_57_0.tar.gz
    /download

tar xzf boost_1_57_0.tar.gz
cd boost_1_57_0/
n=`cat /proc/cpuinfo | grep "cpu cores" | uniq | awk '{print $NF}'`
./bootstrap.sh --prefix=/usr/local
sudo ./b2  -j $n --prefix=/usr/local install
sudo ldconfig
```

Install modis2scidb

```
export LC_ALL="en_US.UTF-8"
yes | sudo apt-get install apt-utils build-essential cmake libgdal-dev gdal-bin git

mkdir gribeiro
mkdir gribeiro/build-linux
cd gribeiro
git clone https://github.com/gqueiroz/modis2scidb.git
cd build-linux

cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE:STRING="Release" -
     DCMAKE_CXX_FLAGS:STRING="-lpthread -std=c++0x" ../modis2scidb/build/cmake
make
sudo make install
sudo ldconfig
```

Docker container running Ubuntu 12.04 (LTS)

Get MODIS HDFs

wget
http://e4ftl01.cr.usgs.gov/MOLT/MOD09Q1.005/2000.02.18/MOD09Q1.A2000049.h10v10.005.2006268191951.hdf

Create array

```
iquery -q "CREATE ARRAY MOD09Q1
   <red:int16, nir:int16, quality:uint16>
   [col_id=48000:67199,1014,5,
   row_id=38400:52799,1014,5,
   time_id=0:9200,1,0];"
```

Array dimension constraint the tile that can be loaded.

Load MODIS to SciDB

```
export LC_ALL="en_US.UTF-8"
cd ~
wget
    https://raw.githubusercontent.com/albhasan/modis2scidb/master/h
    df2sdbbin.py

mkdir sdbbin

git clone https://github.com/albhasan/modis2scidb.git

python ./modis2scidb/checkFolder.py ~/sdbbin/ ~/modis2scidb/
    MOD09Q1 MOD09Q1 &

python ./modis2scidb/hdf2sdbbin.py
    MOD09Q1.A2000049.h10v10.005.2006268191951.hdf   ./sdbbin
    MOD09Q1
```

Test

```
iquery -q "SELECT * FROM MOD09Q1 WHERE col_id < 48010 AND
    row_id < 48010;"
```

Some common operations

Insert data at specific index

```
set lang afl;
create array foo <val:double> [x=0:*,100,0, y=0:*,100,0];

insert(
    redimension(
        apply(
            build(
                <val:double>
                [i=0:9,10,0],
                random()
            ),
            x, 3, y, 10 + i
        ),
        foo
    ),
    foo
);

scan(foo);
remove(foo);
```

redimension matches the array schema to **foo**. Then *insert* uses the X & Y dimensions to put data in place inside **foo**.

Some matrix operations

Identity matrix

```
set lang afl;

build(<vall:double>[row=0:4,5,0,
   col=0:4,5,0], iif(row=col,1,0));
```

Example 1

- Given A and B:

$$A = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 2 & -3 & 1 \end{bmatrix}$$

- Evaluate:

$$2A + 3B$$

Source: http://www.exampleproblems.com/wiki/index.php/LA2.1.1

Example 1: Create arrays

```
CREATE ARRAY A
   <valA:double>
   [row=0:1,2,0, col=0:2,3,0];

CREATE ARRAY B
   <valB:double>
   [row=0:1,2,0, col=0:2,3,0];
```

Example 1: Insert data

```
set lang aql;

INSERT INTO A '[[(0) (1) (2)] [(2) (3) (4)]]';

INSERT INTO B '[[(1) (0) (0)] [(2) (-3) (1)]]';
```

Example 1: Computation

```
set lang afl;

store(join(A, B), C);
apply(C, res, valA * 2 + valB * 3);


-- short form
apply(join(A, B), res, valA * 2 + valB * 3);
```

Example 2

$A * B^t$

Example 2: Computation

```
set lang afl;
load_library('linear_algebra');

spgemm(A, transpose(B));
```

Example 3

Invert a matrix?????

Game of life

R - Connect to SciDB

```
install.packages(c("scidb", "raster"))
library(scidb)
library(raster)

scidbconnect(
    host="localhost",
    port=49902,
    username = "scidb",
    password = "xxxx.xxxx.xxxx",
    protocol="https"
)
```

R - Preparation

```
# Run an AFL query
runquery <- function(q, ret){
   return(iquery(q, `return` = ret, afl = TRUE, iterative = FALSE,
    n = 400))
}

# Transform an array into a matrix
cw2matrix <- function(cw, size){
   matrix(data = as.vector(unlist(cw['val'], recursive = TRUE)),
    nrow = size, byrow = TRUE)
}
```

Util functions to send AFL queries and export results to matrix

R - Game setup

```
aname  = "myCOMWAY"
steps <- 50
size <- 20
qdel <- paste("remove(", aname, ")", sep = "")

qnew <- paste("store(build(<val:double NULL DEFAULT
    null>[i=0:", size - 1, ",", size, ",0,j=0:", size - 1, ",", size,
    ",0],random()%2), ", aname,")", sep = "")

qnext <- paste("insert(project(apply(apply(apply(join(", aname,
    ", window(", aname,", 1, 1, 1, 1, sum(val))), sum, val_sum -
    val),factor,iif((val = 0 and sum != 3), 0,iif((val = 0 and sum
    = 3), 1,iif(sum < 2, -1,iif(sum > 3,-1,0))))    ),newval,
    double(val + factor)), newval), ", aname,")", sep = "")
```

Set up for the game:
An initial query and another for moving one step forward

R - Run the game

```
runquery(qdel, FALSE)
cw <- list()
for (i in 1:(steps)){
   if(i == 1){
      q <- qnew
   }else{
      q <- qnext
   }
   cw[[i]] <- cw2matrix(runquery(q, TRUE), size)
   plot(raster(cw[[i]]), main = paste("COMWAY t =", i-1, sep = "
    "))
}
```

It does the magic!

References & suggested readings

- iquery client http://www.paradigm4.com/HTMLmanual/14.12/scidb_ug/ch02s01.html
- Fields as a Generic Data Type for Big Spatial Data - Camara, G. *et al*.
- A Database Array Algebra for Spatio-Temporal Data and Beyond - Baumann, Peter

Source: http://bigmouthbarry.com/wp-content/uploads/2015/04/all.jpg