Open-source array database

Sparse/dense, multi-dimensional arrays

Distributed storage, parallel processing

Excels at parallel linear algebra

ACID, data replication, no-overwrite versioned data, ...
The R Project

Popular and powerful language for data science

Excellent facilities for threaded and simple parallel tasks

Excellent connectivity to diverse data sources and languages

Largely an in-memory computing system

Limited facilities for distributed parallel linear algebra
The SciDB package for R

Defines two main ways to interact with SciDB:

R data frame iterator interface using SciDB query language

*N-dimensional sparse/dense array class for R backed by SciDB arrays*
library("scidb")
scidbconnect()

# An example reference to a SciDB matrix:
> A <- scidb("A")
> dim(A)
[1] 50000 50000

# Subarrays returned as new SciDB array objects:
> A[c(0,49000,171),5:8]
Reference to the 3x4 dimensional SciDB array.attribute array74e47fac6a51.v

# Use [] to materialize data to R:
> A[c(0,49000,171),5:8][]

<table>
<thead>
<tr>
<th>i</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0.72624070</td>
<td>0.2000578</td>
<td>0.5750903</td>
<td>0.1369318</td>
</tr>
<tr>
<td>1</td>
<td>0.02450512</td>
<td>0.7795026</td>
<td>0.7800414</td>
<td>0.7971742</td>
</tr>
<tr>
<td>2</td>
<td>0.35359630</td>
<td>0.7821995</td>
<td>0.7838033</td>
<td>0.9575580</td>
</tr>
</tbody>
</table>
library("scidb")
scidbconnect()

# SciDB array class example reference to a big dense SciDB matrix:
> A <- scidb("A")
> dim(A)
[1] 50000 50000

# Subarrays returned as new SciDB array objects:
> A[c(0,49000,171),5:8]
Reference to the 3x4 dimensional SciDB array/attribute array74e47faca651/v

# Use [] to materialize data to R:
> A[c(0,49000,171),5:8] []
   i   j
0 0.72624070 0.2000578 0.5750903 0.1369318
1 0.02450512 0.7795026 0.7800414 0.7971742
2 0.35359630 0.7821995 0.7838033 0.9575580
# Arithmetic composed with subsetting:

```r
> X <- A %%*% A[,1:5]
> dim(X)
[1] 50000  5
```

# Mixed SciDB and R object arithmetic (everything computed in SciDB):

```r
> Z <- A[c(0,49000,171),5:7]
> ( (Z + t(Z)) %%*% cbind(rnorm(3)) )[, drop=FALSE]

             i
  0 -1.527964  
  1 -4.547520  
  2 -2.590095  
```

# More interesting linear algebra, a truncated SVD:

```r
> S <- svd(A, nu=3, nv=3)

# Result is a 3-D array: S[0,,] = U,  S[1,,] = Sigma (sparse),  S[2,,] = V
> dim(S)
[1]  4 50000 50000
```
# Arithmetic:
> X <- A %*% A[,1:5]
> dim(X)
[1] 50000 5

# Mixed SciDB and R object arithmetic (everything computed in SciDB):
> Z <- A[c(0,49000,171),5:7]
> ((Z + t(Z)) %*% cbind(rnorm(3)))[, drop=FALSE][]

```
j
i  0
  0 -1.527964
  1 -4.547520
  2 -2.590095
```

# More interesting linear algebra, a truncated SVD:
> S <- svd(A, nu=3, nv=3)

# Result is a 3-D array: S[0,,] = U, S[1,,] = Sigma (sparse), S[2,,] = V
> dim(S)
[1] 4 50000 50000
# Arithmetic:

```r
> X <- A %*% A[,1:5]
> dim(X)
[1] 50000 5
```

# Mixed SciDB and R object arithmetic (everything computed in SciDB):

```r
> Z <- A[c(0,49000,171),5:7]
> ( (Z + t(Z)) %*% cbind(rnorm(3)) )[, drop=FALSE]

          i 0
0 -1.527964
1 -4.547520
2 -2.590095
```

# More interesting linear algebra, a large partial SVD:

```r
> S <- svd(A, nu=3, nv=3)

# Result is a 3-D array: S[0,,] = U, S[1,,] = Sigma (sparse), S[2,,] = V
> dim(S)
[1] 4 50000 50000
```
The SciDB array class facilitates exploration and analysis of large data in a familiar language.
It is sometimes possible to use SciDB arrays in R packages with little (or sometimes even no) modification.

Here is a biclustering example:

```r
> library("biclust")
> library("s4vd")
> data(lung)
> A = lung
> x = biclust(A, method=BCssvd, K=1)

# Now with SciDB arrays:
> library("ssvdp4")
> X = as.scidb(A)
> xl = biclust(X, method=BCssvd, K=1)

# Compare the two results (they are numerically nearly identical):
> sqrt( x@info$res[[1]]$u - xl@info$res[[1]]$u )
   [,1]
[1,]  5.202109e-16
```
An example from genomic sequencing

Load sequence data into 2-D sparse SciDB array X (sample x position) and sort samples by genomic position into array Y for display:

```r
cpy>
  iquery("redimension_store(
    cross_join(X,
    project(
      sort(
        apply(
          aggregate(X, min(position) as p_min, read),
          sorted, read
        ), p_min
      ), sorted
    ), read, n
  ), Y
)

> Y <- scidb("Y")
> image(Y, 800, 550, col=rainbow(6), ...)
```
Example genomic sequence coverage (pileup) plot
WE ARE HIRING
join@paradigm4.com

www.paradigm4.com
www.scidb.org