Massively parallel analytics for large datasets in R with *nza* package

Cezary Dendek, Przemysław Biecek, Paweł Chudzian, Justin Lindsey

useR! 2010
July 21, 2010
Main assumptions

- **Data stored in Netezza Performance Server database**
  - Provides data parallelism in cluster environment
  - Provides what R lacks: performance & out-of-memory storage

- **Processing should be close to the (large) data**
  - Limit a data transfer between the cluster nodes

- **Symmetrical processing**
  - Single Instruction Multiple Data
  - Lack of deadlocks due to smart representation
Tradeoff between Flexibility and Performance

- R language (high flexibility, low performance)
- In-database processing
- C++ language (low flexibility, high performance)
Approaches to data processing in nza

• **Direct data processing in R**
  > Execution of R functions passed to **data operators**
  > Abstraction over SQL
  > Parallel execution over the data chunks
  > High flexibility, medium performance

• **Construction of statistical models using in-database processing**
  > Efficient, parallel algorithms for model construction
  > High performance, medium flexibility

• **High performance&flexibility**
  > Fast in-database calculation of data aggregates
  > Flexibility of aggregate manipulation in R
Direct parallel data processing in R

- R code is executed in database, close to data and in parallel.

- **Data operator** is responsible for
  - Propagation of given task and data stream to R
  - Selecting optimal model of execution
  - Propagation of the results to DB

- Data operators

<table>
<thead>
<tr>
<th></th>
<th>nzApply</th>
<th>nzTApply</th>
<th>nzGroupedApply</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data abstraction</strong></td>
<td>row</td>
<td>group</td>
<td>row in group</td>
</tr>
<tr>
<td><strong>State machine?</strong></td>
<td>SL</td>
<td>SL</td>
<td>SF, merged</td>
</tr>
<tr>
<td><strong>Type of transformation</strong></td>
<td>1 row -&gt; 1 row</td>
<td>1 group -&gt; 1 row</td>
<td>(1+) row -&gt; 1 row reduction</td>
</tr>
</tbody>
</table>
Direct parallel data processing in R

```r
library(nza)
nzConnectDSN("NSQL")
nzadults <- nz.data.frame("database..table")

FUN1 <- function(x) {sqrt(x[[1]])}
FUN2 <- function(x) {mean(x)}

nzApply(nzadults[,1:2], FUN1, output.table="ttable1")
nzTApply(nzadults[,1:3], nzadults[,4], FUN2, output.table="ttable2")
```
In-database processing

- Efficient, specialized parallel algorithms (SQL+native processing)
  - Decision trees (classification and regression)
  - K–means
  - Naive Bayes
  - One- and two-way ANOVA
  - Simple statistics and support for hypothesis testing

- Output compatible with native R objects

- Model creation and application
In-database processing
Decision trees [regression]

• Tree building
  > `tree.model = nzDecTree(form, data, outtable = uniqueTableName(), minsplit = 1000, maxdepth = 62, id = "id", qmeasure = "wAcc")`

• Prediction
  > `predict.nzDecTree(tree.model, newdata, id = "id")`

• Generic
  > `print(tree.model), plot(tree.model)`
In-database processing

Decision trees [regression]

- **Tree building**
  
  > `tree.model` = nzRegTree (form, data, outtable = uniqueTableName(), minimprove = 0.1, maxdepth = 62, minsplit = 2, id = "id")

- **Prediction**
  
  > `predict.nzRegTree (tree.model, newdata, id = "id")`

- **Generic**
  
  > `print (tree.model), plot (tree.model)`
In-database processing

K – means

• Model building
  
  > \texttt{model} = \texttt{nzKMeans} (data, k = 2, maxiter = 10, 
distance = "euclidean", outtable = uniqueTableName(), id = "id", 
getLabels = F, randseed = 1234)

• Prediction
  
  > \texttt{predict.nzKMeans (model, newdata, id = "id")}

• Generic
  
  > \texttt{print (model)}
In-database processing
Naive Bayes

- **Model building**
  
  ```
  > model = nzNaiveBayes(form, data, 
  > outtable = uniqueTableName(), id = "id")
  ```

- **Prediction**

  ```
  > predict.nzNaiveBayes(model, newdata, id = "id")
  ```

- **Generic**

  ```
  > print(model)
  ```
In-database processing

ANOVA

- **Testing hypothesis about means**
  
  ```
  > model.av = nzAnova (form, data, 
  outtable = uniqueTableName())
  ```

- **Generic**
  
  ```
  > print (model.av)
  ```
In-database processing
Example R session

```r
library(mza)
nzconnect("admin", "password", "TT4-R040", "nza")
nzadult = nz.data.frame("adult")
adultTree = nzRecTree(income ~, data=nzadult, maxdepth=4)
plot(adultTree)

> adultTree
node, split, n, deviance, yval, (yprob)
   * denotes terminal node

1) root 32561 0 small ( 0.24081 0.75919 )
   2) marital_status=Married-civ-spouse 14976 0 small ( 0.44685 0.55315 )
      4) education_num < 12 10807 0 small ( 0.33102 0.66898 )
      5) capital_gain < 5013 9979 0 small ( 0.29672 0.70328 ) *
      9) capital_gain > 5013 528 0 large ( 0.97917 0.02083 ) *
   3) marital_status = Married-civ-spouse 17585 0 small ( 0.66534 0.33466 )
      6) capital_gain < 6849 17274 0 small ( 0.04915 0.95085 ) *
      7) capital_gain > 6849 311 0 large ( 0.96463 0.03537 ) *

> head(predict(adultTree, nzadult))
   ID CLASS
1 12 large
2 64 large
3 88 large
4 112 large
5 124 large
6 136 large

> getContTable(~EDUCATION+OCCUPATION, nzadult, T) Same
          Adm-clerical Armed-Forces Craft-repair Exec-managerial Farming-fishing Handlers-cleaners
    12th       38       0       170       24       44       44       71
    9th        67       0       175       54       37       123
    10th       38       1       58       15       16       38
    7th-8th    0        0       23        4       16       16
Asoco-acdm   6        0       43        1       36       40
   Masters    11       0       116       19       70       46
   l1c-4th    193      0       115      145       14       24
```
Data aggregates (high performance & flexibility)

- Decomposition of model creation into:
  - In-database calculation of Sufficient Statistics for model construction
  - Dot product matrix ($X^TX$)
  - Multivariate contingency table
  - Model fitting directly in R environment

- Data aggregates
  - Computed „close“ to the data in (embarrassingly) parallel way
  - Information extraction (reduction of the data being transferred)
  - Does not reduce flexibility of model creation
  - Computed and transmitted once, used multiple times
Dot product \((X^TX)\) matrix

- Sufficient statistics for linear models

- In-database calculation of model matrix with
  - support for categorical variables (dummy variables)
  - support for continuous variables (centering, scaling)
  - support for bootstrap samples and weighted case

- Model size depends only on the number of columns (and levels)
Dot product ($X^T X$) matrix

- nza functions using the dot product matrix:
  - `nzLm(formula, nzdf)`
  - `nzRidge(formula, nzdf, lambda=10)`
  - `nzPCA(formula, nzdf)`
  - `nzANOVA(formula, nzdf)`
  - `nzPCR(formula, nzdf)`
  - `nzCanonical(formula, nzdf)`

- R package nzMatrix can operate on matrices as large as e.g. 100k x 100k (limited by total RAM of the NPS)
Multivariate Contingency Table

- 2 or more categorical variables
  (a multimatrix, a hypercube)

- The “dot product” matrix for categorical data

- Analysis of the relations and correspondence between given variables (correlation, CA, MCA)
Multivariate Contingency Table

- **Aggregate creation:**
  - Actual
    model = nzTable(form, nzdf, makeMatrix=T)
  - Read from DB
    model = getContTab(form, nzdf, makeMatrix=F)

- **Example**
  > getContTab(~EDUCATION+OCCUPATION, nzadult, T)
Multivariate Contingency Table

- \textit{nza} functions that make use of contingency table:
  - \texttt{nzca(form, nzdf, ...)}
  - \texttt{nzchisq.test(form, nzdf, ...)}
  - \texttt{nzMantelHenszel.test(form, nzdf, ...)}
  - \texttt{nzGoodman(form, nzdf, ...)}

- Compute once and reuse
Summary: *nza*

- Provides various **tools** and **strategies** for parallel, out-of-memory data processing
  - Direct use of R code with specialized **data operators**
  - R wrappers for specialized in-DB functions for model creation
  - In-DB intensive calculation of data aggregates (sufficient statistics) and R functions for model creation (from suff. statistics)
- **Uses Netezza Performance Server as a backend**
Thank you

Questions?

cdendek@netezza.com

pbiecek@netezza.com