The nzseq and nzxsc packages for processing genetic data in the Netezza Performance Server

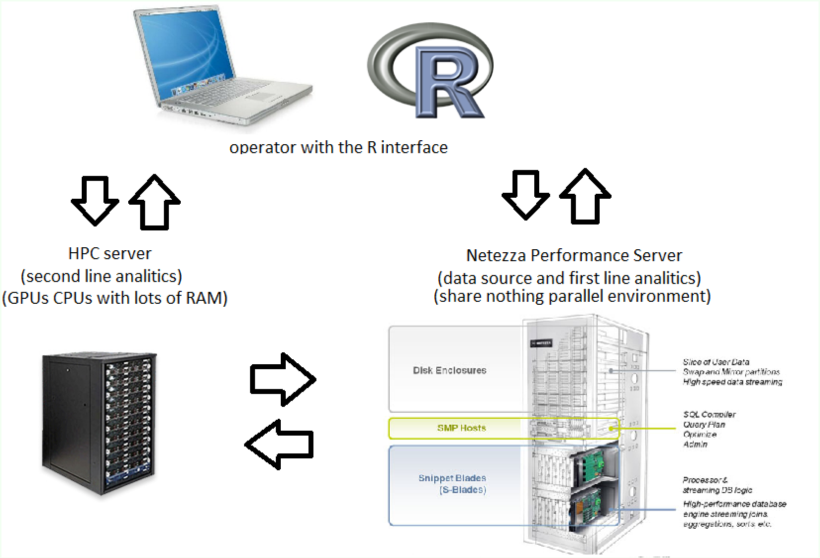
Przemyslaw.Biecek@gmail.com

University of Warsaw / Netezza, an IBM company

Outline

1. Introduction to the Netezza Performance Server,
2. The problem / The hardware / The software - overview,
3. The nzseq package - remote operations on large genetic sequences,
4. The nzxsc package - remote high performance computing.

The architecture


operator with the R interface
HPC server
(second line analitics)
(GPUs CPUs with lots of RAM)
Netezza Performance Server
(data source and first line analitics)
(share nothing parallel environment)
Disk Enclosures
SMP Hosts
Snippet Blades
(S-Blades)
Slice of User Data
Swap and Mirror partitions
High speed data streaming
SQL Compiler
Query Plan
Optimize
Admin
Processor &
streaming DB logic
High-performance database
engine streaming joins
aggregations, sorts, etc.

Overview - Movie

R
ATM
Mutant p53
p53
Siah1
TRF2
Δ133p53
bio NZ LABS
F
M
<30
>30
<40
>40
$\hat{\beta}_{lasso} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|_{L1}$
$f(\theta|x) = \frac{f(x|\theta)f(\theta)}{\int f(x|\theta)f(\theta)d\theta}$

The nzxsc package - NPS and HPC

```
# create pointers for tables in databases
xdf1 <- x.data.frame("user","password","host1","database1","table1")
xdf2 <- x.data.frame("user","password","host2","database2","table2")

# create MPI grids on a HPC cluster
solver1 <- getSolver("user","password","host3",nodes=6)
solver2 <- getSolver("user","password","host3",nodes=6)
solver3 <- getSolver("user","password","host3",nodes=6)

# submit requests for computations
res1 <- solve(xdf1, solver1, mode = "bygroup", broadcast=FALSE, group="COL1",
              fun=dim)
res2 <- solve(xdf2, solver2, mode = "byrow", broadcast=FALSE,
              fun=predict.lm, object=lmmodel)
res3 <- solve(xdf1, solver3, mode = "dataset", broadcast=TRUE,
              fun=boot, R=100, statistic=function(x) var(x[,2]))
...

# check are the computations finished
isFinished(res1)
materialize(res1)
```

The nzxsc package - a list of functions

Functions in the nzxsc package:

- set a pointer to a table in NPS database
  x.data.frame(), [,], $,
- create a set of MPI nodes with R on the HPC cluster
  getSolver(),
- start computations (asynchronic), by row, by group, by dataset with or without broadcasting, you can use any R code and/or call external software like samtools or BWA
  solve().

The nzxsc package - How does it work?

- On the R client - user operates only on pointers to data sources or pointers to solvers, results from computations are returned to the client,
- On the HPC cluster - a manager creates a MPI cluster of R processes, starts the communication with database, runs the submitted R code on each MPI node, supplies each process with a row, group of rows or whole dataset from a NPS table,
- On the NPS - data are divided among processing units, on each processing unit the database engine creates a data.frame like object and sends it to the R process which may be run locally or on other machine, results are stored in the database or returned to the R client.

The nzseq package - upload/download data

```
> library(nzseq)
> nzConnect("user", "password", "server", "database")
> head(nz.data.frame("humanGenome"))
          CHUNK    CHUNK_ID      SEQ_ID
1 ATGGTCCCTAGAAC          1           1
2 ATGGTCACTAGCCC          2           1
3 GCGGTCCACCGAAC          3           1
4 TAGGTCGGTAGATT          1           2
5 ATTGGCCCTAGAAT          2           2
6 ACCGAAACTAGAAT          1           3

> # upload sequence 'gattaca' to a table 'humanGenome'
> putSequence(22, table="humanGenome", seq="gattaca", chunksize=10000)

> # download sequence id=2 from a table 'humanGenome'
> (tmp <- getSequence(2, table="humanGenome"))
Sequence 2(humanGenome): TAGGTCGGTAGATTATTGGCCCTAGAAT

> deleteSequence(2, table="humanGenome")

> (tmp <- getSubSequence(1, table="humanGenome", pstart=1, pstop=5))
Sequence 1(humanGenome): ATGGT
```

The nzseq package - send R code/download results

```
> userFun <- function(seq) {
+   require(seqinr)
+   c2s(translate(s2c(seq)))
+ }
> (tmp <- seqApply("humanGenome", userFun))
        userFun  SEQ_ID
1  MVPRTWSLARGPPN       1
2       *VGRLLALE       2
3            TETR       3

> # [pwm] description of a motif
> # A [ 13  0 52  0 25 ]
> # C [ 13  5  0  0  7 ]
> # G [ 18 48  1  0 15 ]
> # T [  9  0  0 53  6 ]
> mat = matrix(c(13,0,52,0,25,13,5,0,0,7,18,48,1,0,15,9,0,0,53,6),4,5,by=T)
> # find positions in which likelihood of binding is greater than 1
> findMotifsBindingSites("humanGenome", mat, 1)
  SEQ_ID  POS   SCORE
1      1   34  2.8475
2      1   19  2.8475
3      2    4       2
```

The nzseq package - a list of functions

Functions in the nzseq package:

- for data downloading/uploading
  setSequence(), getSequence(), getSubSequence(),
- for processing R code remotely in database
  seqApply(),
- for calling remotely database functions
  findMotifsBindingSites(), getTranslation(),
  getLocalAlignment(), getGlobalAlignment()...

Summary

- The package nzxsc is designed for handling efficient parallel processing of massive datasets on computational clusters easily.
- The package nzseq is designed for handling operations on large number of large sequences. Sequences are stored in the parallel database and user can download a few of them or may upload an R code for remote computations.
- Both packages create a layer of abstraction between R user and the remote hardware (data bases and/or computational nodes).