data.table

Higher speed time series queries

Matthew Dowle

LondonR, July 2009

[Repeat from Rmetrics, July 2008]

Presentation Overview

- Objective
- Review : data.frame, with and subset
- data.table syntax [not SQL like]
- data.table[data.table] is a join to keyed columns – O(log n) binary search in C
- Examples
- Demo: svSocket's evalServer()

Objective

1. Reduce programming time
    - Less code
    - Easier to write, read, debug and maintain

2. Reduce compute time

base::data.frame()

DF = data.frame(a=1:5, b=6:10, c=11:15)
[.data.frame = function(x,i,j,...)
i and j must be either integer or character, no expressions

DF[2:3,] # select * from DF where row is 2 or 3
DF[,"b"] # select column b from DF
DF[2:3,"b"] # select column b from DF where row is 2,3

rownames(DF) = letters[1:5]
DF["a", 3] # select column 3 from DF where row named a

base::with()

DF = data.frame(a=1:5, b=6:10, c=11:15)
Instead of
    DF$a+DF$b+DF$c
can write
    with(DF, a+b+c)

So “data.*frame”* as in *evaluation* frame, not just as in *picture* frame (i.e. rectangular)

Works with lists too (base R) :
L = list(a=1:5, b=6:10, c=11:15)
with(L, a+b+c)

base::subset()

DF[DF$a>2 & DF$b<9, ”c”]

or

DF[with(DF,a>2 & b<9), “c”]

or

subset(DF, a>2 & b<9, c)

a>2 is a ***vector scan*** i.e. every item of a is tested

i expression

DT = data.table(a=1:5, b=6:10, c=11:15)

DT[b>2] # select * from DT **where** b>2

i and j can now be *expressions of column names*

i is like a call to subset(DT,i)

j expression

j is like a call to with(DT,j)

DT[,a+b+c] # **select** a+b+c from DT

Combining with and subset :

DT[b>2, a+b+c] # select a+b+c from DT where b>2

NB: Still vector scanning i.e. every row is tested.

DT[b>2, plot(b*c)] # j doesn't have to be data

Time series join

DT = data.table(c1=letters[1:5], c2=6:10, c3=11:15)

setkey(DT, c1) # sorts, and sets “sorted” attribute

DT[J(“c”)]

   # binary search for “c” in the sorted column

   # returns row 3

All but the last column must be equi-joins

The last column may be 'last observation carried forward' or 'roll' join.

Example …

id	date	price
SBRY	20080501	380.50
SBRY	20080502	391.50
SBRY	20080506	389.00
VOD	20080501	159.30
VOD	20080502	163.30
VOD	20080506	160.80

```
1.PRC[J("SBRY")]          # all 3 rows for SBRY
2.PRC[J("SBRY",20080502),price]  # 391.50
3.PRC[J("SBRY",20080505),price]  # NA
4.PRC[J("SBRY",20080505),price,roll=TRUE]  # 391.50
5.PRC[J("SBRY",20080505),price,rolltolast=TRUE] #
391.50
6.PRC[J("SBRY",20080601),price,roll=TRUE] # 389.00
7.PRC[J("SBRY",20080601),price,rolltolast=TRUE] # NA
```

Performance

- PRC - all daily prices since 1986 for all non-US equities
- 183,000,000 rows (id, date, price)
- 2.7 GB

```
> system.time(PRC[id==”VOD”]) # vector scan
user    system  elapsed
66.431  15.395  81.831
> system.time(PRC[J(“VOD”)])  #  Binary
  search with (integer) factors
user    system  elapsed
0.003   0.000   0.002
```

[.data.table

- **i** expression of column names, or integer / boolean, or data.table
- **j** expression of column names, or integer / character column lookup
- **by** quoted series of expressions e.g. ”region,month=trunc(date/100)”
- **nomatch** 0 (inner join) or NA (outer join)
- **roll** TRUE/FALSE
- **rolltolast** TRUE/FALSE
- **mult** “all”/”first”/”last”
- **which** row numbers returned by the join
- **with** FALSE allows column selection

Recursive

Since a data.table query returns a data.table :

DT[where,select,by=...]**[order(...)]**[...][...]...

When DT has a key :

DT[DT] == DT

May be useful.

R DB Stack ?

data.table

+

ff

+

svSocket

+

R

svSocket::evalServer()

- Philippe Grosjean
- Allows R to R communication
- Any R to any R, anywhere e.g. 64Bit linux R server queried by 32bit windows R client
- No need to setup cluster in advance, easy to do
- Doesn't block command line

```
Rserver> startSocketServer(port=8888)
Rclient> con=socketConnection(port=8888)
Rclient> evalServer(con,DT["VOD"])  # query
Rclient> evalSever(con,x,3.4)       # push
```

- Demo ... http://www.youtube.com/watch?v=rvT8XThGA8o
  [maximise and press the HD button, no audio]

Closing thoughts

- Why parallelise inefficient code? Beware of ==. Beware of character().
- 64bit first choice - disk is slow
- Remote namespace e.g. attach(port=8888), then evalServer can be abstracted away. RObjectTables?
- Shared R workspace (data, functions, objects), not just tables
- Remote graphics very easy – cross platform
- An R instance can be a client to many servers and a server to many clients all at the same time.
- Computing on the R client/R server language ... agent based models, genetic algorithms
- Extend data.table to data.array? **:** DA[,J(“a”),J(“r”),dim4<15,sum(dim5)]
- data.table on R-forge:
  - faster sorting, data.frame inheritance and more by Tom Short
  - test.data.table (100+ tests) ported to RUnit by Yohan Chalabi

Contact

- Matthew Dowle
- mdowle@mdowle.plus.com
- LinkedIn.com
- R-forge: data.table