Directions in Statistical Computing 2014

Renjin's JIT

Thinking about R as a Query Language

Alexander Bertram
BeDataDriven

Quick Intro: Renjin

- R-language Interpreter written in Java, uses GNU R core packages (base, stats, etc) as-is
- Goals: Completeness first, performance next
- C/Fortran: Supported with translator and emulation layer
- Can run roughly ~50% of CRAN packages (see packages.renjin.org)
- Actively user group, diverse

R as a "Query Language"

~~How can R be as fast as Fortran or C++ ?~~

How can R be more like SQL?

- Analyst describes the *what*
- Query planner determines the *how*
  - Implicit parallelism
  - Target diverse architechture (in-memory, single node, clusters)

Is R dynamic?

Argument: Not where/when performance matters

“But R is too dynamic!”

```
airlines <- read.bigtable(“airlines”)
print(nrow(airlines))  # ~240m

fit.exp <- function(x, max.iter = 10 ) {
  rate <- 1 / mean(x)
  repeat {
    loglik <- sum(-dexp(r = rate, x = lambda, log = T)
    if( goodEnough(loglik) ) break
    rate <- nex
 }
}
```

Complicated Argument Matching

sum() is group generic, dispatches based on argument

Is the break() function redefined?

```
airlines <- read.bigtable("airlines")
delay <- airlines$delay[airlines$delay > 30]

dexp <- function(x, rate=1, log = FALSE) {
   mean <- 1/rate
   d <- exp(-x / mean) / mean
   if(log) return(log(d))
   d

    }

fit.exp <- function(x, max.iter = 10 ) {
  rate <- 1 / mean(x)
  repeat {
    loglik <- sum(-dexp(r = rate, x, log = T)
    if( logLik > epsilon ) break
    rate <- update(rate)
 }
}

rate <- fit.exp
```

Real world example:
Distance Correlation
[see energy package]

```r
function (x, y, index = 1)
{
 x <- dist(x)
 y <- dist(y)
  x <- as.matrix(x)
  y <- as.matrix(y)
  n <- nrow(x)
  m <- nrow(y)
  dims <- c(n, ncol(x), ncol(y))
  Akl <- function(x) {
    d <- as.matrix(x)^index
    m <- rowMeans(d)
    M <- mean(d)
    a <- sweep(d, 1, m)
    b <- sweep(a, 2, m)
    return(b + M)
  }
  A <- Akl(x)
  B <- Akl(y)
  dCov <- sqrt(mean(A * B))
  dVarX <- sqrt(mean(A * A))
  dVarY <- sqrt(mean(B * B))
  V <- sqrt(dVarX * dVarY)
  if (V > 0)
    dCor <- dCov/V
  else dCor <- 0
    return(list(dCov = dCov, dCor = dCor, dVarX = dVarX, dVarY = dVarY))
}
```

Optimizations: Views

```
x <- dist(x)
y <- dist(y)
x <- as.matrix(x)
y <- as.matrix(y)
# GNU R: x^2 + y^2 memory alloc'd
# Renjin: ~ 0
```

DistanceMatrix

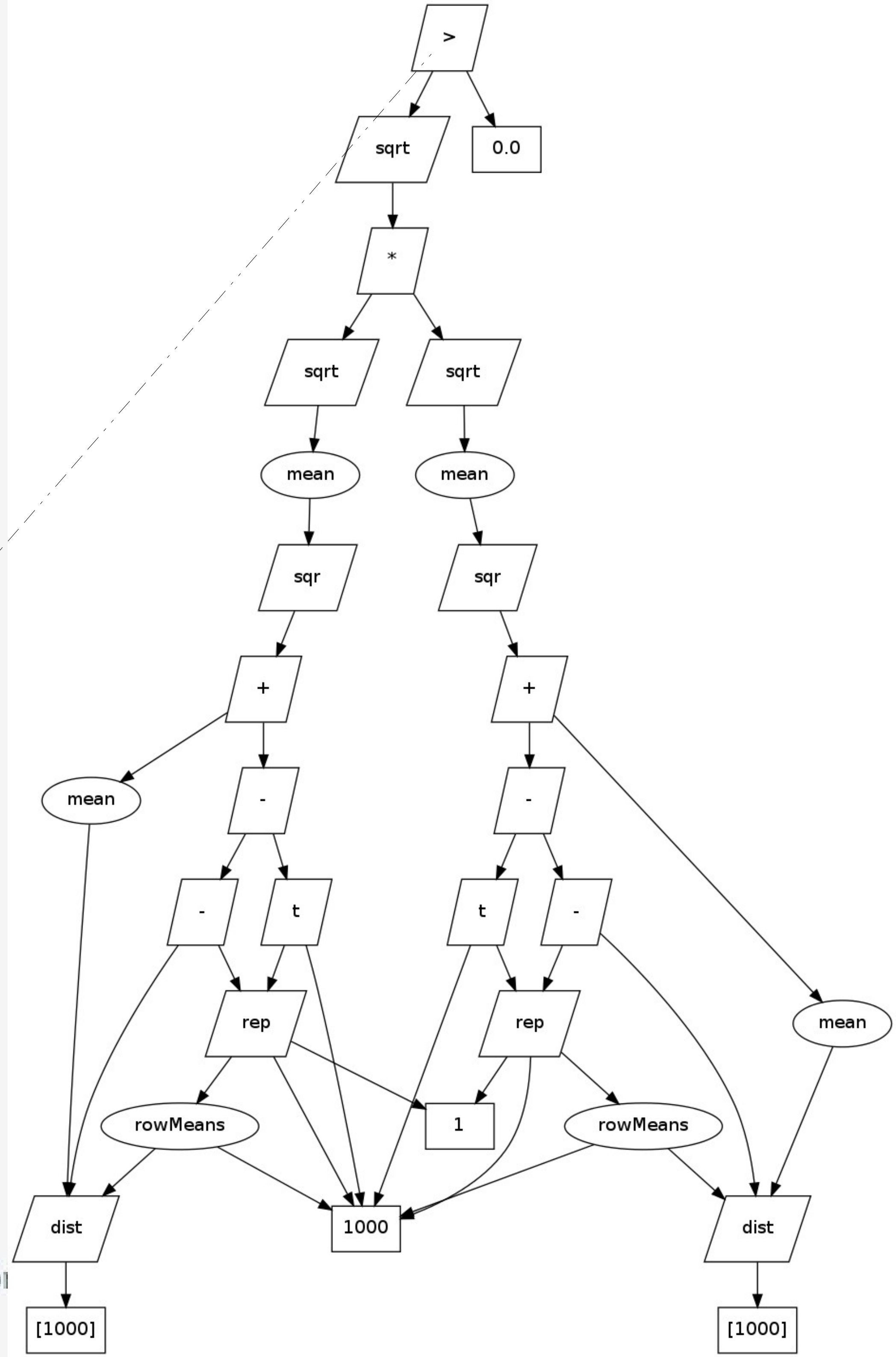
```
public class DistanceMatrix extends DoubleVector {
  private Vector vector;

  public double getElementAsDouble(int index) {
    int size = vector.length();
    int row = index % size;
    int col = index / size;
    if(row == col) {
      return 0;
    } else {
      double x = vector.getElementAsDouble(row);
      double y = vector.getElementAsDouble(col);
      return Math.abs(x - y);
    }
  }
  public int length() {  return vector.length() * vector.length();   }
}
```

Deferred Evalution

- Defer computation of **pure** functions when inputs exceed some threshold:

```
x <- (1:100) + 4     # x is computed

y <- (1:e^6) + 4     # no work done
                     # x is a view

z <- y – mean(z)

z <- dnorm(z)

print(z)  # triggers evaluation
```

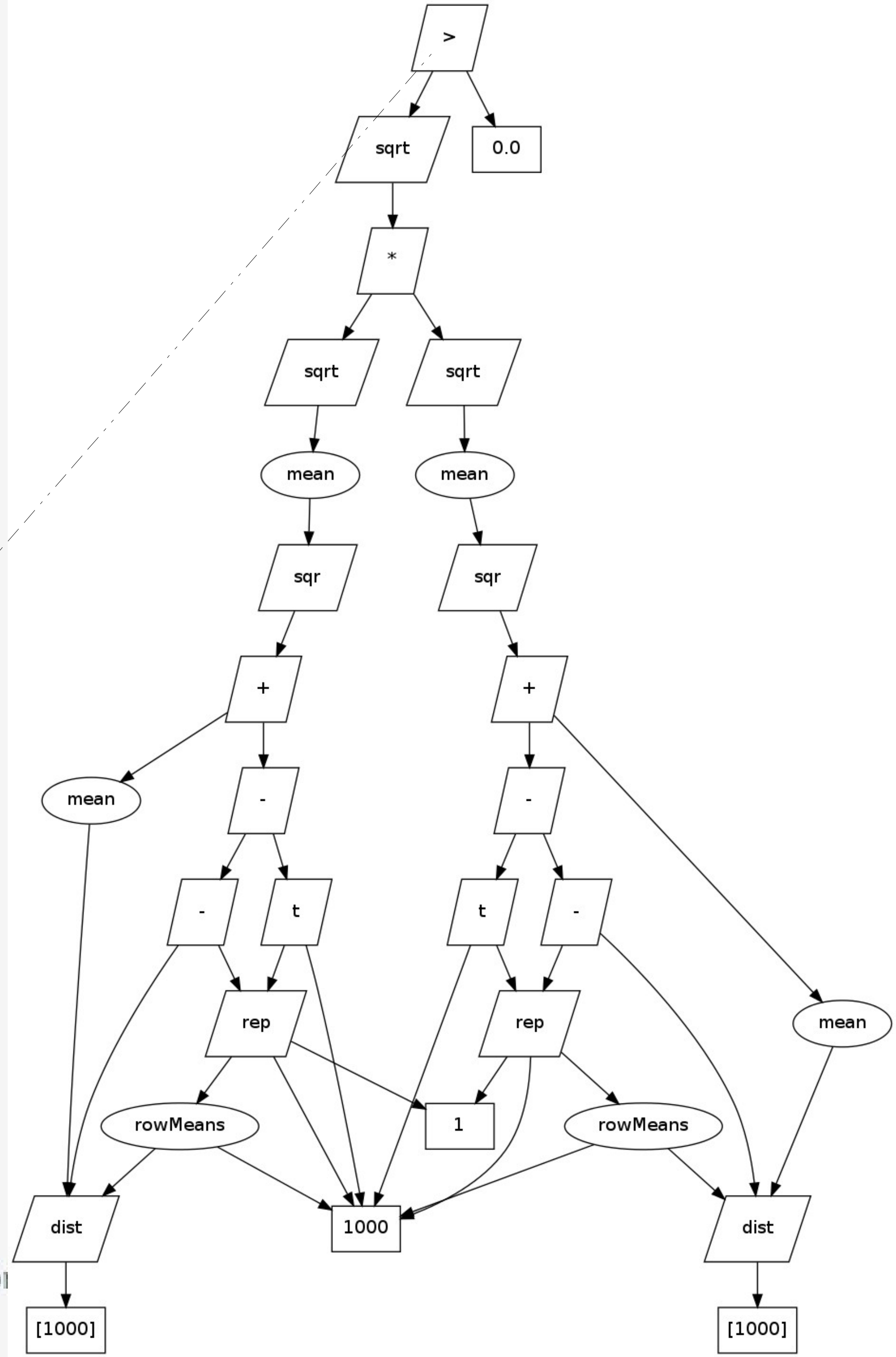
```r
function (x, y, index = 1)
{
 x <- dist(x)
 y <- dist(y)
  x <- as.matrix(x)
  y <- as.matrix(y)
  n <- nrow(x)
  m <- nrow(y)
  dims <- c(n, ncol(x), ncol(y))
  Akl <- function(x) {
    d <- as.matrix(x)^index
    m <- rowMeans(d)
    M <- mean(d)
    a <- sweep(d, 1, m)
    b <- sweep(a, 2, m)
    return(b + M)
  }
  A <- Akl(x)
  B <- Akl(y)
  dCov <- sqrt(mean(A * B))
  dVarX <- sqrt(mean(A * A))
  dVarY <- sqrt(mean(B * B))
  V <- sqrt(dVarX * dVarY)
  if (V > 0)
    dCor <- dCov/V
  else dCor <- 0
    return(list(dCov = dCov, dCor = dCo
}
```

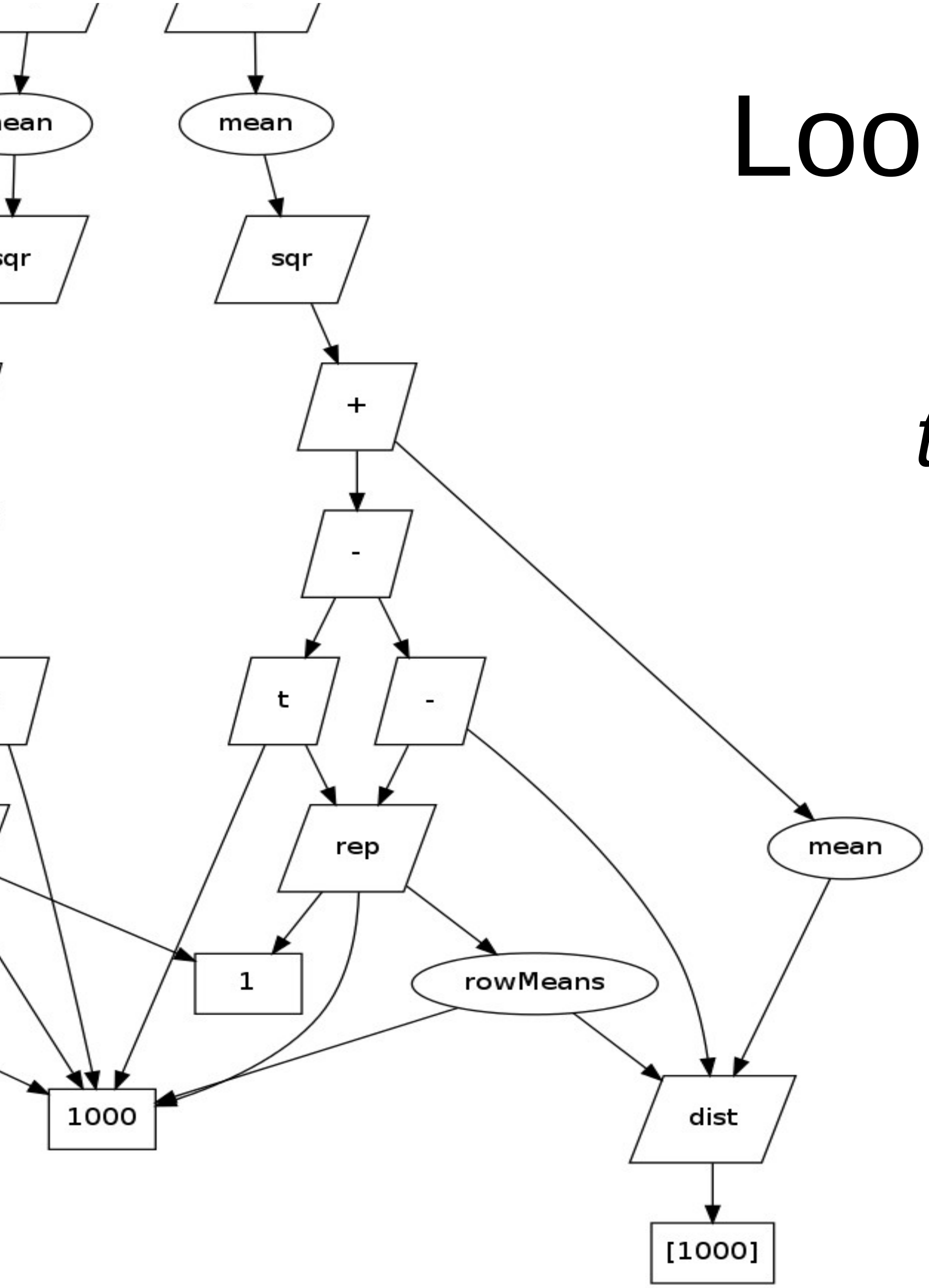


Query Planner

- Once evaluation is triggered: we have a better broad view of the calcuation to be completed
- Computation Graph is essentially a pure function
- We can reorder operations, and easily see which branches can be evaluated independently, in parallel

```
function (x, y, index = 1)
{
 x <- dist(x)
 y <- dist(y)
  x <- as.matrix(x)
  y <- as.matrix(y)
  n <- nrow(x)
  m <- nrow(y)
  dims <- c(n, ncol(x), ncol(y))
  Akl <- function(x) {
    d <- as.matrix(x)^index
    m <- rowMeans(d)
    M <- mean(d)
    a <- sweep(d, 1, m)
    b <- sweep(a, 2, m)
    return(b + M)
  }
  A <- Akl(x)
  B <- Akl(y)
  dCov <- sqrt(mean(A * B))
  dVarX <- sqrt(mean(A * A))
  dVarY <- sqrt(mean(B * B))
  V <- sqrt(dVarX * dVarY)
  if (V > 0)
    dCor <- dCov/V
  else dCor <- 0
    return(list(dCov = dCov, dCor = dCo
}
```





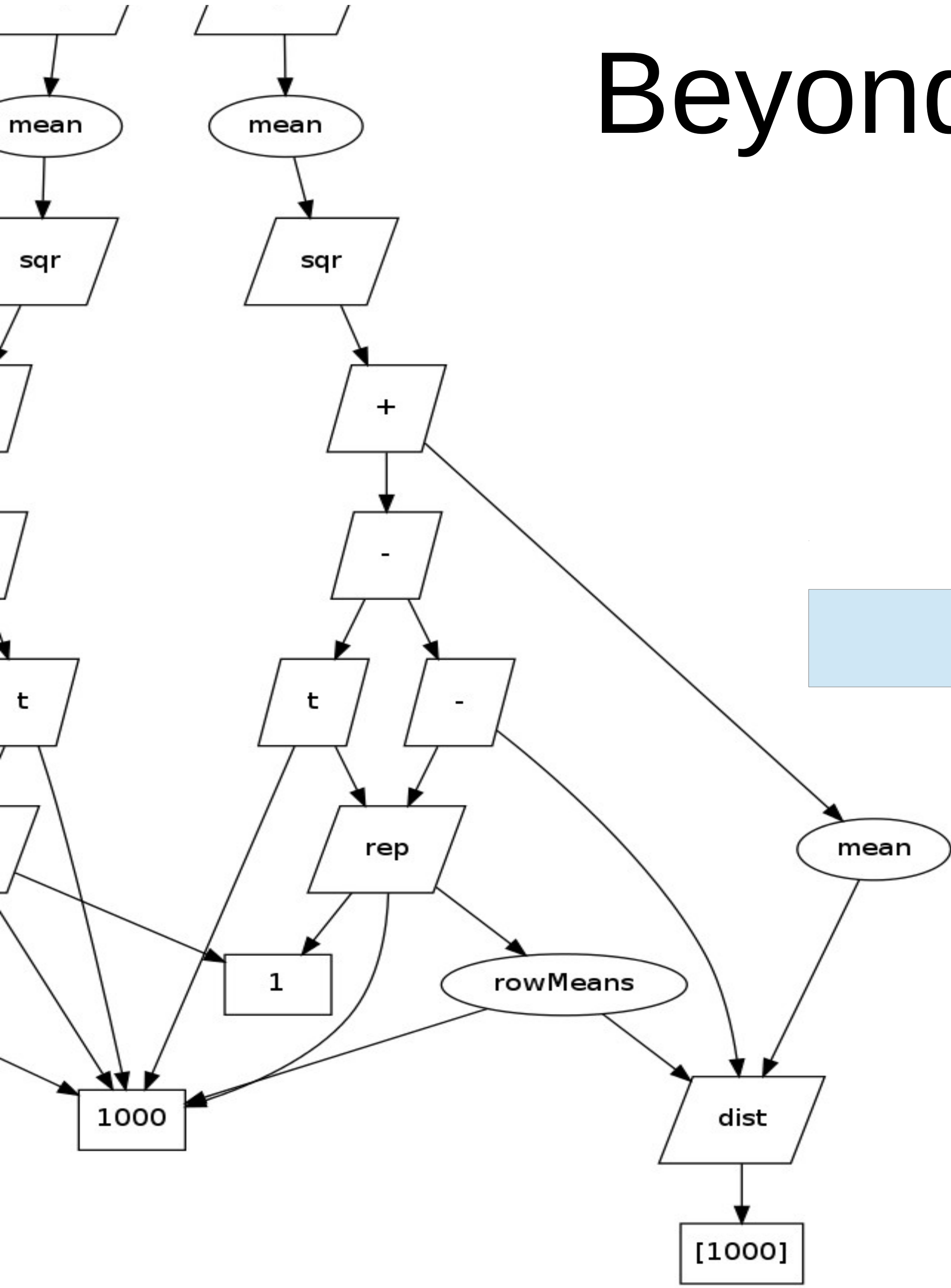
Loop Fusion

```
mean(op1(op2(op3(x)))
```

transformed to...

```
double sum = 0;
for(int i..1000) {
  sum += op1(op2(op3))
}
```

Beyond Bytecode

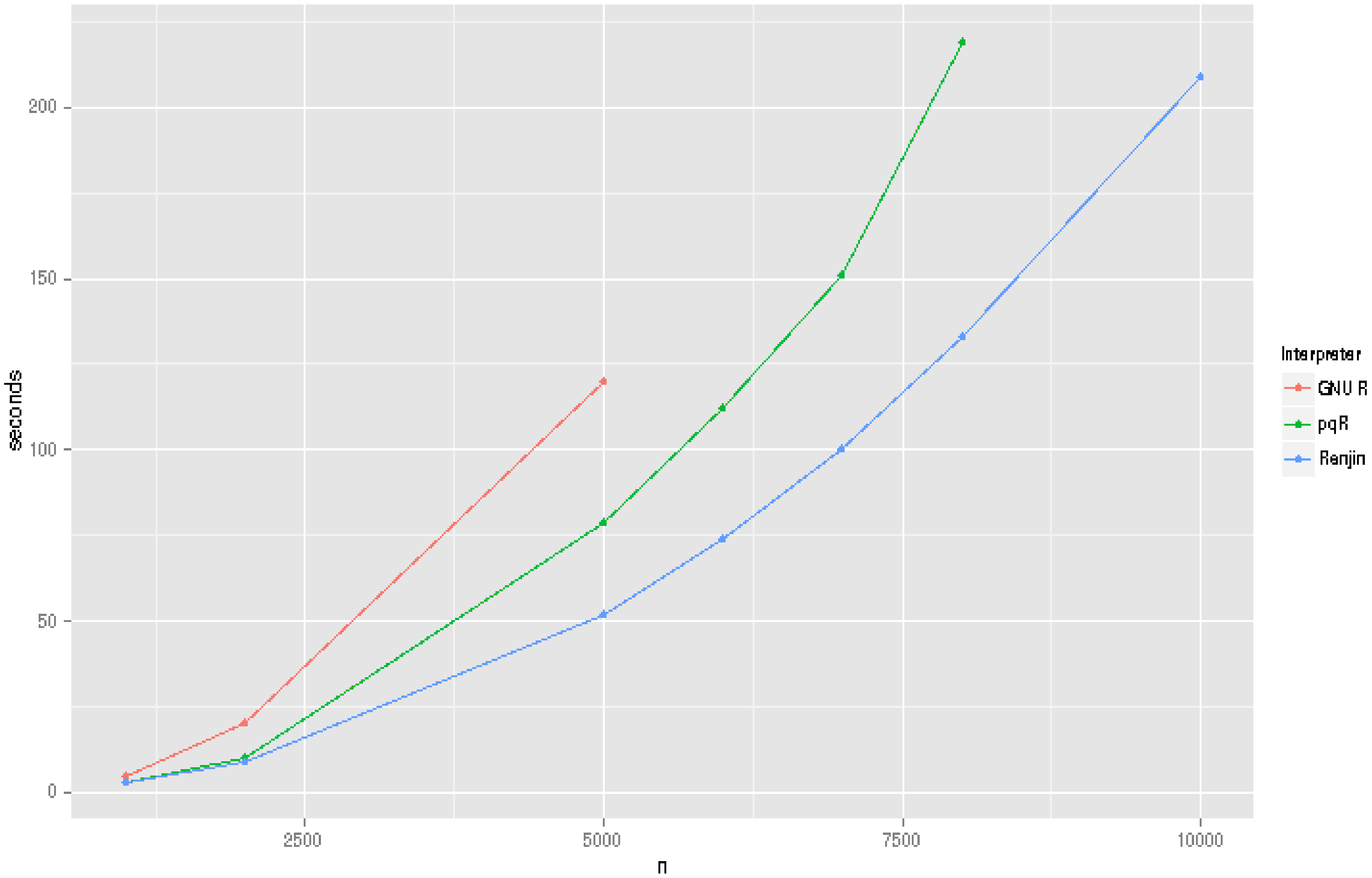


JVM Byte Code →
Native Machine Code

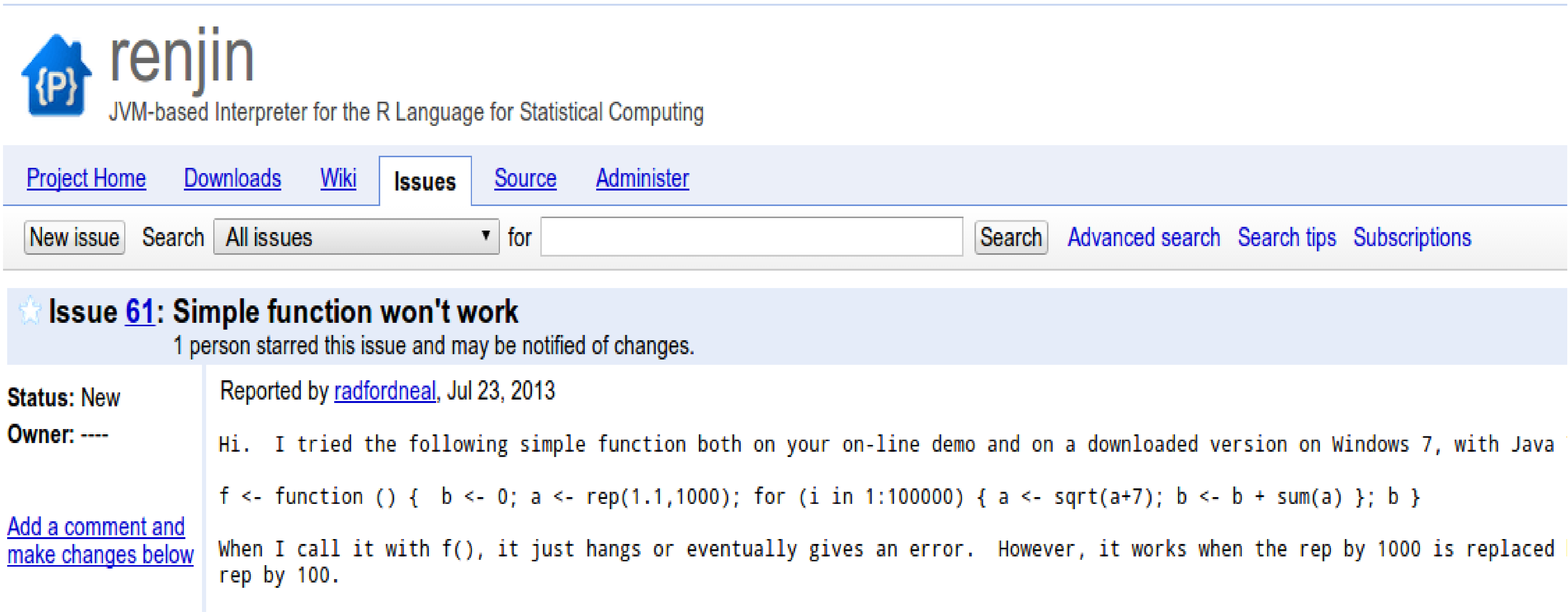
SQL Query

OpenCL

Results



seconds
200
150
100
50
0
2500
5000
7500
10000
n
Interpreter
GNU R
pqR
Renjin


renjin
JVM-based Interpreter for the R Language for Statistical Computing
Project Home
Downloads
Wiki
Issues
Source
Administer
New issue
Search
All issues
for
Search
Advanced search
Search tips
Subscriptions
Issue 61: Simple function won't work
1 person starred this issue and may be notified of changes.
Status: New
Owner: ----
Add a comment and make changes below
Reported by radfordneal, Jul 23, 2013
Hi. I tried the following simple function both on your on-line demo and on a downloaded version on Windows 7, with Java
f <- function () { b <- 0; a <- rep(1.1,1000); for (i in 1:100000) { a <- sqrt(a+7); b <- b + sum(a) }; b }
When I call it with f(), it just hangs or eventually gives an error. However, it works when the rep by 1000 is replaced
rep by 100.

Loops!

```
m <- 4
for (i in 1:m) {
  x = exp (tanh (a^2 * (b^2 + i/m)))
  r[i%%10+1] = r[i%%10+1] + sum(x)
}
```

Kaboom!

(thanks Radford!)

Loops!

- R gives you the flexibility to mix imperative with functional approaches
- In many dynamic languages (JS, Ruby), sophisticated runtime analysis is required to identify and compile hotspots in the code.
- In R, they're pretty easy to spot:

```
x <- 1:1e6
for(i in seq_along(x)) {
...
}
```

```
for (i in 1:m) {
  x = exp (tanh (a^2 * (b^2 + i/m)))
  r[i%%10+1] = r[i%%10+1] + sum(x)
}
```

BB1:
$\tau_3 \leftarrow$ (: 1.0d m_0)
$\Lambda 0_1 \leftarrow 0$
$\tau_2 \leftarrow$ length(τ_3)

BB2: [L0]
$r_1 \leftarrow \Phi(r_0, r_2)$
$\Lambda 0_2 \leftarrow \Phi(\Lambda 0_1, \Lambda 0_3)$
$i_1 \leftarrow \Phi(i_0, i_2)$
$x_1 \leftarrow \Phi(x_0, x_2)$
if $\Lambda 0_2$ >= τ_2 => TRUE:L3, FALSE:L1, NA:ERROR

BB3: [L1]
$i_2 \leftarrow \tau_3[\Lambda 0_2]$
$\tau_4 \leftarrow$ (^ a_0 2.0d)
$\tau_5 \leftarrow$ (^ b_0 2.0d)
$\tau_6 \leftarrow$ (/ i_2 m_0)
$\tau_7 \leftarrow$ (+ τ_5 τ_6)
$\tau_8 \leftarrow$ (* τ_4 τ_7)
$\tau_9 \leftarrow$ (tanh τ_8)
$x_2 \leftarrow$ (exp τ_9)
$\tau_{10} \leftarrow$ (%% i_2 10.0d)
$\tau_{11} \leftarrow$ (+ τ_{10} 1.0d)
$\tau_{12} \leftarrow$ ([r_1 τ_{11})
$\tau_{13} \leftarrow$ (sum x_2)
$\tau_{14} \leftarrow$ (%% i_2 10.0d)
$\tau_{15} \leftarrow$ (+ τ_{14} 1.0d)
$\tau_{16} \leftarrow$ (%% i_2 10.0d)
$\tau_{17} \leftarrow$ (+ τ_{16} 1.0d)
$r_2 \leftarrow$ ([<- r_1 τ_{17})

BB4: [L2]
$\Lambda 0_3 \leftarrow$ increment counter $\Lambda 0_2$
goto L0

BB5: [L3]
return NULL

Compared to other dynamic languages?

- **Argument:** Speculative specialization works very well for long-running code, but unnecessary for most statistical code with many loops:
  - Simulations
  - Iterative algorithms
  - ?
- **Needs to be tested...**

Home About Downloads Blog Packages Support Documentation

Packages

Package	Downstream	Languages	Problems	Description	
		A3		TF	A3: Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
	2	abc			Tools for Approximate Bayesian Computation (ABC)
		abcdeFBA		TF	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package.
		ABCExtremes		TF	ABC Extremes
		ABCp2		TF	Approximate Bayesian Computational model for estimating P2
		abctools	C		Tools for ABC analyses
		abd			The Analysis of Biological Data
	82	abind		TF	Combine multi-dimensional arrays
		aBioMarVsuit			A Biomarker Validation Suit for predicting Survival using gene signature.
		abn	C		Data Modelling with Additive Bayesian Networks
		AcceptanceSampling			Creation and evaluation of Acceptance Sampling Plans
		ACCLMA		TF	ACC & LMA Graph Plotting
		ACD		TF	Categorical data analisys with complete or missing responses
		Ace		TF	Assay-based Cross-sectional Estimation of incidence rates
	1	acepack	Fortran	TF	ace() and avas() for selecting regression transformations
		acer	C++		The ACER Method for Extreme Value Estimation
		aCGH.Spline			Robust spline interpolation for dual color array comparative genomic hybridisation data
		ACNE			Affymetrix SNP probe-summarization using non-negative matrix factorization
		aCRM		TF	Convenience functions for analytical Customer Relationship Management
		acs			Download and manipulate data from the US Census American Community Survey
		Actigraphy			Actigraphy Data Analysis
		actuar	C	TF	Actuarial functions

packages.renjin.org

cor2var-examples	OK

xch-examples [OK]

```
>
>     #Example
>     xch(5,0.17)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1 0.17 0.17 0.17 0.17
[2,] 0.17    1 0.17 0.17 0.17
[3,] 0.17 0.17    1 0.17 0.17
[4,] 0.17 0.17 0.17    1 0.17
[5,] 0.17 0.17 0.17 0.17    1
>
```

ar1-examples [OK]

```
>
>     # Example
>     ar1(5,0.75)
           [,1]       [,2]       [,3]       [,4]       [,5]
[1,]          1       0.75     0.5625   0.421875 0.31640625
[2,]       0.75          1       0.75     0.5625   0.421875
[3,]     0.5625       0.75          1       0.75     0.5625
[4,]   0.421875     0.5625       0.75          1       0.75
[5,] 0.31640625   0.421875     0.5625       0.75          1
>     ar1(3,0.25)
       [,1]   [,2]   [,3]
[1,]      1   0.25 0.0625
```

Developing CI + benchmarking system for testing optimizations

More Information

- http://www.renjin.org
- http://packages.renjin.org
- http://docs.renjin.org/en/latest/