The Not Quite R (NQR) Project: Explorations Using the Parrot Virtual Machine

Michael J. Kane\textsuperscript{1} and John W. Emerson\textsuperscript{2}

\textsuperscript{1}Yale Center for Analytical Sciences, Yale University

\textsuperscript{2}Department of Statistics, Yale University
Outline

1  The sky isn’t falling... is it?

2  Overview: New directions for R

3  Explorations with the Parrot Virtual Machine
The R Project

• 18 years and counting... with roots extending 35+ years
• Over 4200 packages on Bioconductor and CRAN
• Tiobe Programming Community Index rank of 34, just behind Matlab (26) and SAS (28): http://www.tiobe.com/
• Aguably the most popular language for research in statistical computing
• R does what we need 99% of the time
So, why are we having this session?
Topics at the frontier of statistical computing

- Bytecode support
- Threading models
- Reflection
- 64-bit indexing
- Performance
Bytecode and Virtual Machine Support

Unless he was delayed by a committee meeting earlier this morning, we just heard from Luke about the R bytecode compiler.

Progress has been made!
Threading

We can’t spawn threads in R

We can make use of threaded code
  • via multi-threaded BLAS and LAPACK [4]
Do we need threads?

- Process parallelism often trumps thread parallelism [8], because processes control:
  - memory allocation locks
  - mmap page-fault locks
- Parallel computing is well-supported in R: 16 different packages on CRAN address parallel computing via process parallelism

Maybe we don’t need it, but thread support would be nice.
Reflection

Objects that are external pointers (or non-native R objects) can’t be copied directly:

```r
> library(bigmemory)
> x <- big.matrix(2, 2, init=0)
> y <- x
> x[1,1] <- 99
> y[,]

   [,1]  [,2]
[1,]  99   0
[2,]   0   0
```
Updating R for 64-bit indexing

- About 430,000 lines of code in R src directory (~370,000 lines in .c files, ~60,000 lines in .h files)
- Very difficult and time-consuming [5]
- Limited or no academic currency
Performance: Very good (more later)!

However, low-level benchmarks performed by Justin Talbot in late 2010 [7] show that:

- Scalar operations are inefficient compared to Lua via Riposte:
  - About 40% of overhead comes from traversing the abstract syntax tree (AST)
  - About 60% comes from memory management

- Sequences of vectorized operations in R are about 10 times slower than hardware capabilities (e.g. \(2 \times (x + y)\))
However:

The sky isn’t falling. Yet.
Current efforts on the frontier represented today

- Continue development of R (Luke and R Core)

- Migrate R to C++, preserve the syntax (Andrew)

- Start from scratch, preserve the syntax (Simon, Mike and Jay)
Some efforts not directly represented here today

- Omegahat (Duncan Temple Lang [10])
- Start from scratch, change the grammar (Duncan Temple-Lang and Ross Ihaka [3])
- R on the JVM (Alexander Bertram, Peter Robinette, Michael Williams [1])
- Riposte, a Lua-based interpreter for R (Justin Talbot [9])
- A JIT for R (Jan Vitek [13])
- A new R-like language (Ross Ihaka and Brendan McCardle [2])
- A Lisp-based system (Tony Rossini [6])
The Not Quite R (NQR) Project:
Explorations Using the Parrot Virtual Machine
What is the Parrot Virtual Machine?

- “Parrot is a virtual machine designed to efficiently compile and execute bytecode for dynamic languages.” [http://www.parrot.org](http://www.parrot.org)

- Formally started by the Perl community around 2001

- Parrot Foundation created in 2008

- Includes a suite of tools for quickly developing new high-level languages and compilers

- Provides high-level language interoperability
Which languages are currently supported on Parrot?

Actively developed and stable:

• Rakudo Perl 6
• Parrot Lua
• Winxed
• nqp
• C/C++ through a native call interface (NCI)

About 25 other languages in various stages of development
Why develop a language for Parrot?

- Language interoperability

- Provides a full-featured assembly language:
  - No need to re-implement arrays, hashes, ...
  - Parrot collects the garbage for us

- Implementing a language is a matter of mapping the grammar to the existing constructs of the compiler toolkit

- JIT on the horizon

- An active, friendly, and helpful developer community
Exploration with the Parrot Virtual Machine

- We designed and Jay implemented a system supporting a subset of the S syntax using the Parrot Virtual Machine http://www.parrot.org.

- Code available: https://github.com/NQRCRCore/

- NQR stands for “Not Quite R” where “Not Quite” is an understatement.

- Some things don’t work: It’s Jay’s fault.
NQR on the Parrot Virtual Machine

- Support the core S syntax including vectors of Integer, Float, and String.
- Leverage existing libraries like the GNU Scientific Library or R’s libRmath.so
- Make initial design decisions consistent with a longer-term “scalable” model
Benchmark: a trivial while loop

Note from Jay: I haven’t yet mastered the for loop, sorry.

```
N <- 1000
while (N > 0) {
  N = N - 1
}
```
Benchmark: a trivial while loop
Benchmark: mean of random exponentials

Notes from Jay:

- `rexp()` in NQR is not vectorized as in R and pays the price of a non-optimized loop
- `mean()` uses the GNU Scientific Library implementation.

```r
N <- 1000
set.seed(1,2)
foo <- mean(rexp(N, 1.0))
```
Benchmark: mean of random exponentials
NQR’s Potential on Parrot VM

- Bytecode support: yes
- Threading models: in Parrot pipeline
- Reflection: yes, by design, with language interoperability
- 64-bit indexing: yes
- Performance: Naive NQR won’t beat the compiled C performance of R, but will continue to improve as Parrot evolves.
Future mucking about in the sandbox

• Refine/debug core language with vectors only

• Add memory-mapped files for larger-than-RAM objects for seamless scalability

• Add lists (a basic hash already exists)

• Add classes for `matrix` and `data.frame`

• Add `read.csv()` so we can use real data

• Explore graphics
Want to play with it?

https://github.com/NQRCore

You’ll need:

• **Parrot** [http://www.parrot.org](http://www.parrot.org)
• libffi (this dependency will be phased out)
• Jay has only tested in Linux; MacOS should be fine.
• Windows? In theory, yes (Parrot attempts to support Windows and more).
Appendix: NQR syntax examples

```
jay@bayesman:~:/Desktop/NQR$ ./installable_nqr

Not Quite R for Parrot VM, Version 0.0.7, July 29, 2011.

To exit, use <ctrl>-D.
Please see t/00-sanity.t for currently-supported syntax.

> a <- 1000 + 100 * rexp(10, 1.2345)
> print(c(mean(a), sd(a), min(a), max(a)))

1110.205677 122.5590509 1003.827809 1334.90338

> b <- sort(a)
> print(paste("Min two different ways:",
             a[which.min(a)], b[0]))

Min two different ways: 1003.827809 1003.827809
```
References I


References II


References III

The pnmath package for R.  
www.stat.uiowa.edu/~luke/R/experimental/.

Implicit and explicit parallel computing in R  

JIT grant.  
http://www.cs.purdue.edu/people/faculty/jv/.