
Scaling Data Analytics
Jan Vitek

Challenges

- How do we program big data?
- What are the tools?
- What are the abstractions?
- How do we debug, visualize, tune big data?

Some big data infrastructures

Hadoop

MapReduce

X10

RHIPE

Pig

Hive

Flume/Java

4 Myths

- Big data is big.
- Big data is speed.
- Big data is storage.
- Big data is hard.

Requirements

- Scale up vs. Scale down
- Rapid feedback, interaction with data, partial results
- Familiarity, ease of development
- Ease of deployment
- Portability and heterogeneity
- Robustness
- Efficiency

A tale of two communities

- **Computer Scientists**: Fixed programs, transient data.
  i.e. there will always be another input
- **Data Scientists**: Fixed data, transient programs.
  i.e. there will always be another query.

- *This dichotomy leads to a different world view in terms of design. In CS, languages/tools are built around static code abstractions. In DS, everything is dynamic and lightweight.*

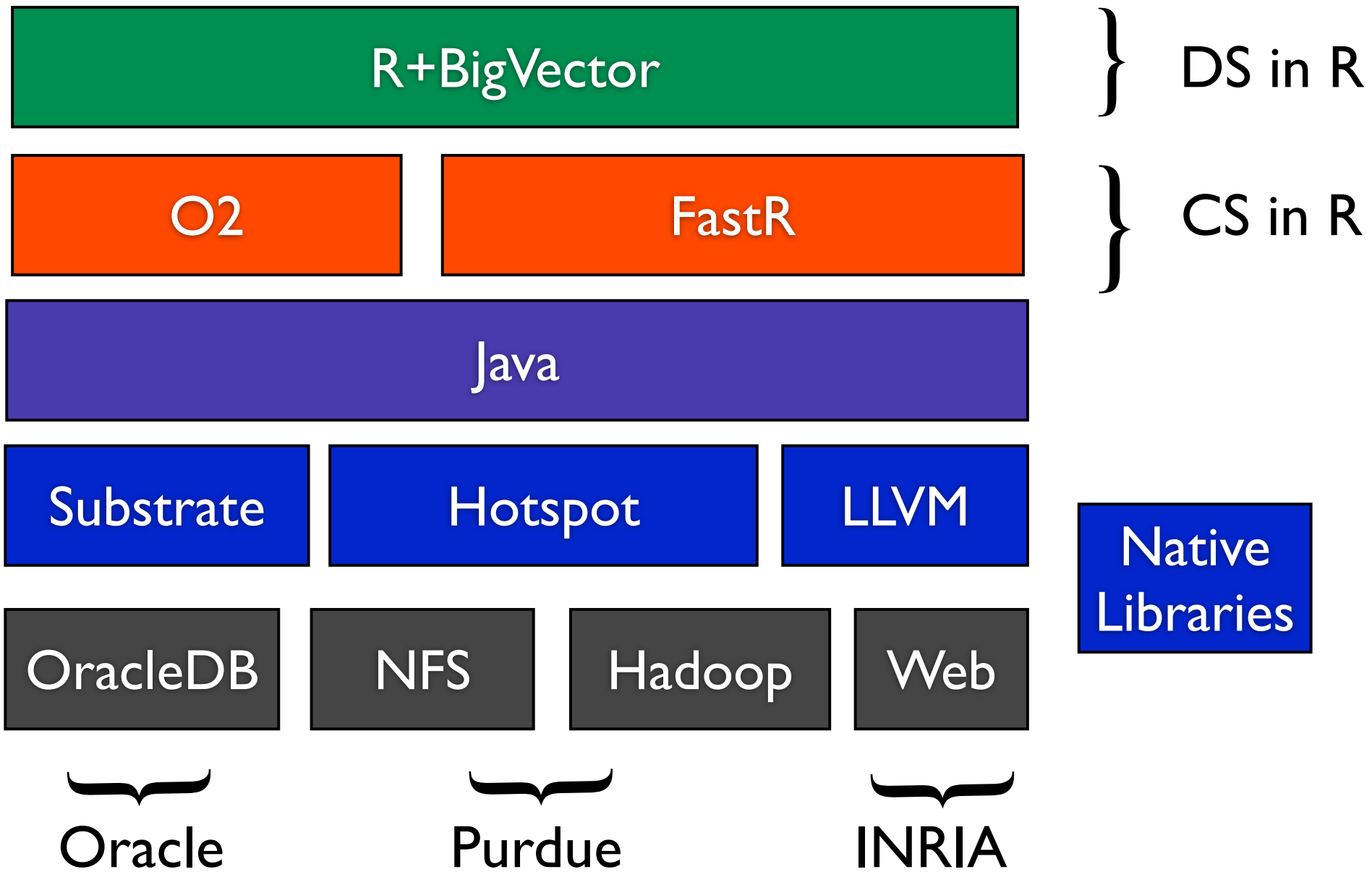
High-level dynamic languages

- Programming is simplified by the language virtual machine
  - memory management
  - threading
  - platform heterogeneity
- At a cost
  - Performance
  - Footprint

ReactoR…

- … create an open source platform for data analytics at scale
- … built in collaboration by Purdue, INRIA, Stanford & Oracle

ReactoR Overview



Why R?

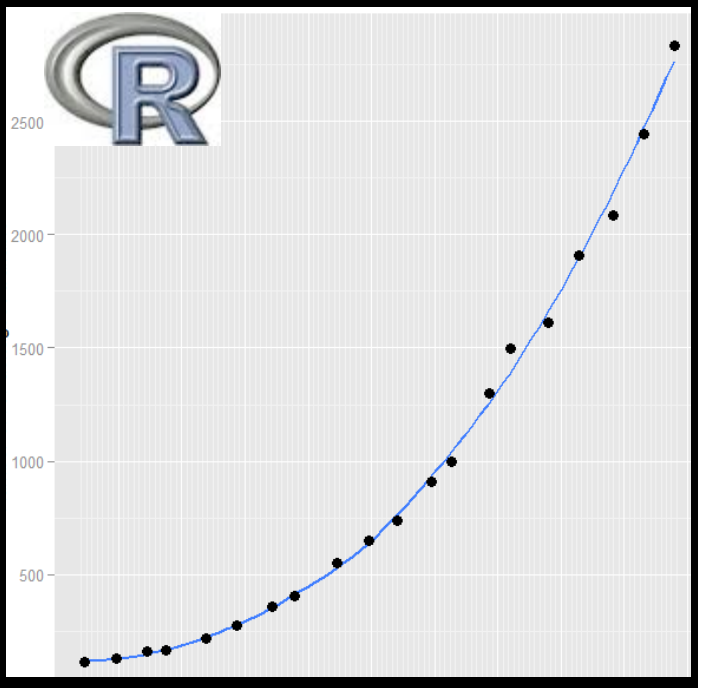
… language for data analysis and graphics

… open source

… books, conferences, user groups

… 4K+ packages

… 3mio users



Scripting data

read data into variables

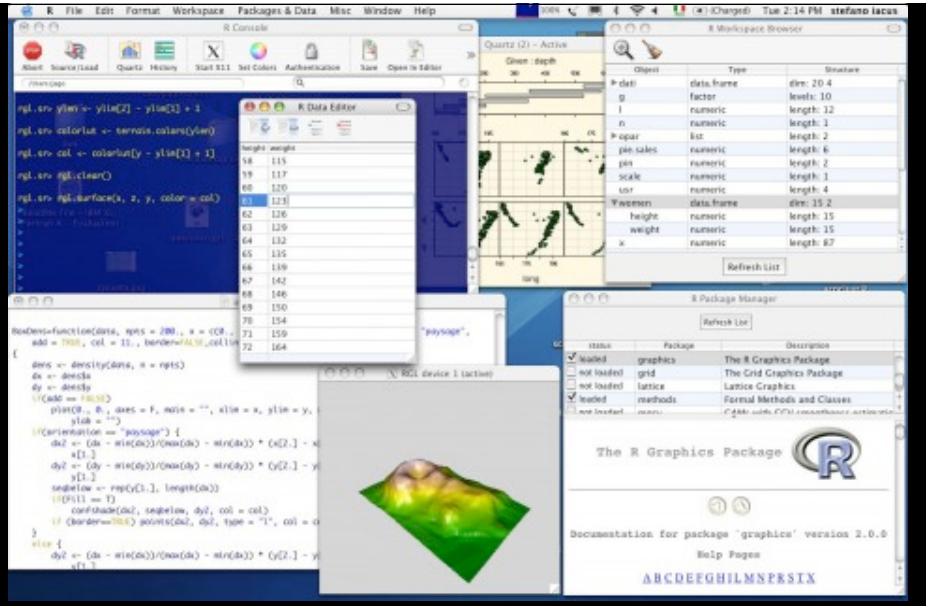
make plots

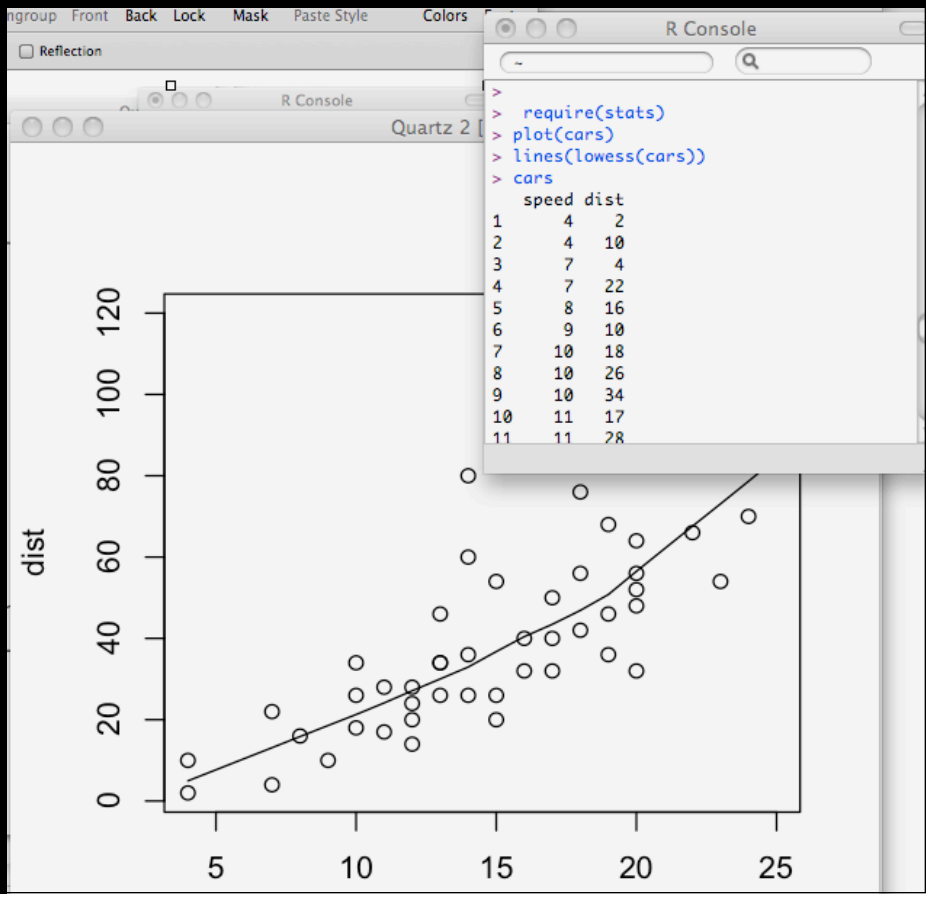
compute summaries

more intricate modeling

develop simple functions to automate analysis

…





Why Java?

… portable

… supports heterogenous platforms

… concurrent

… robust and stable

… fast enough

… books, conferences, user groups

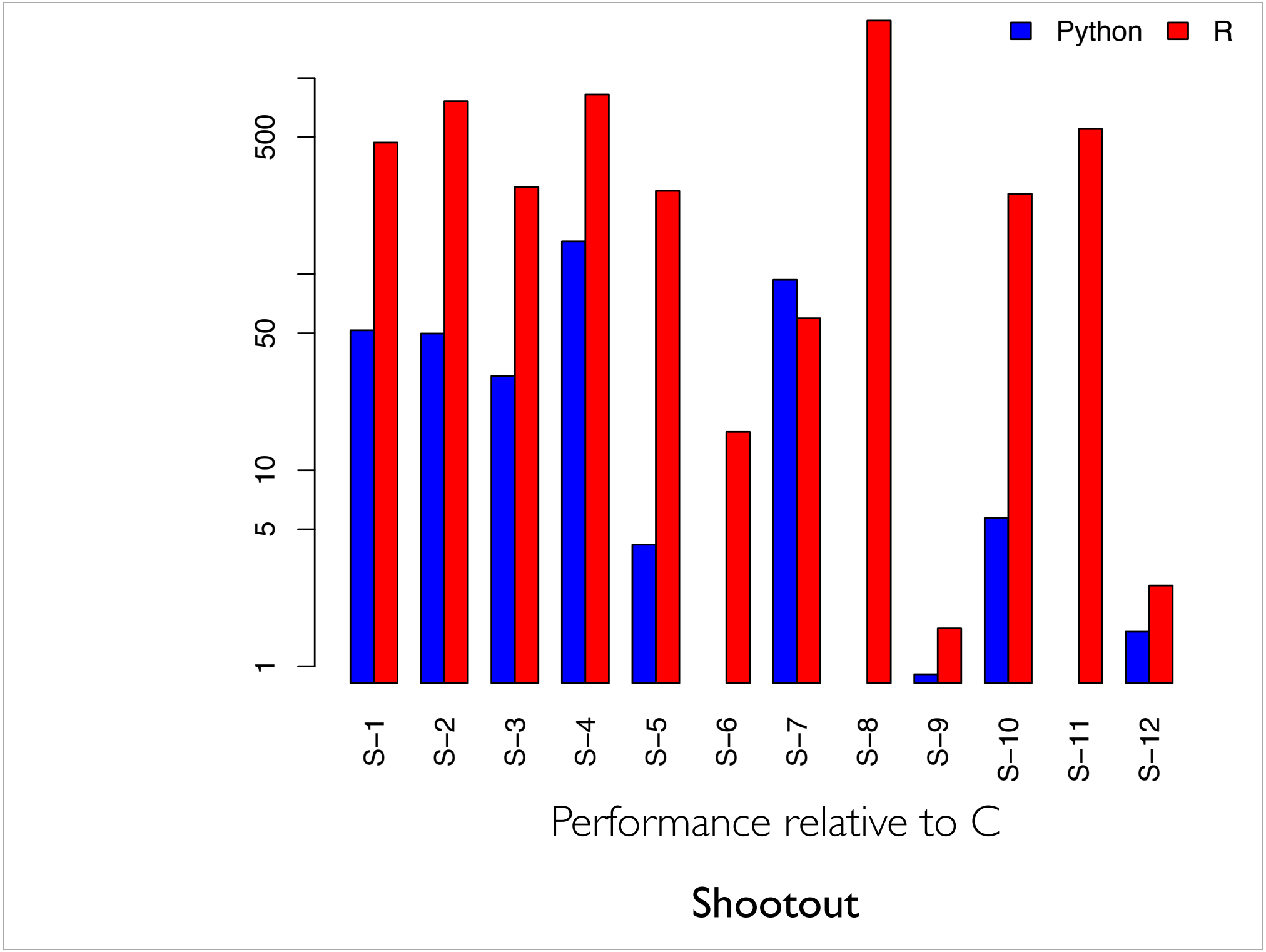
… thousands of packages

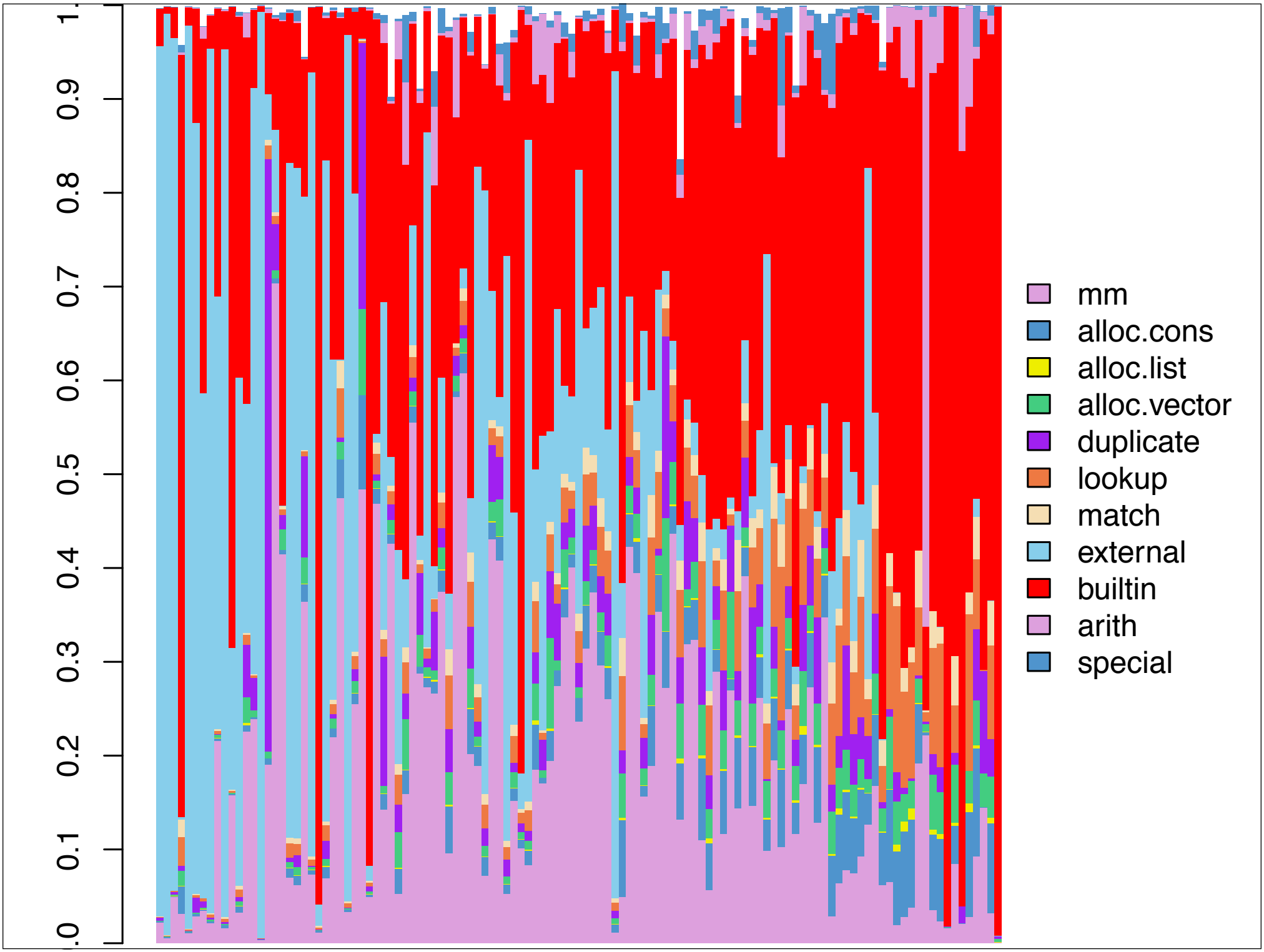
… millions of developers

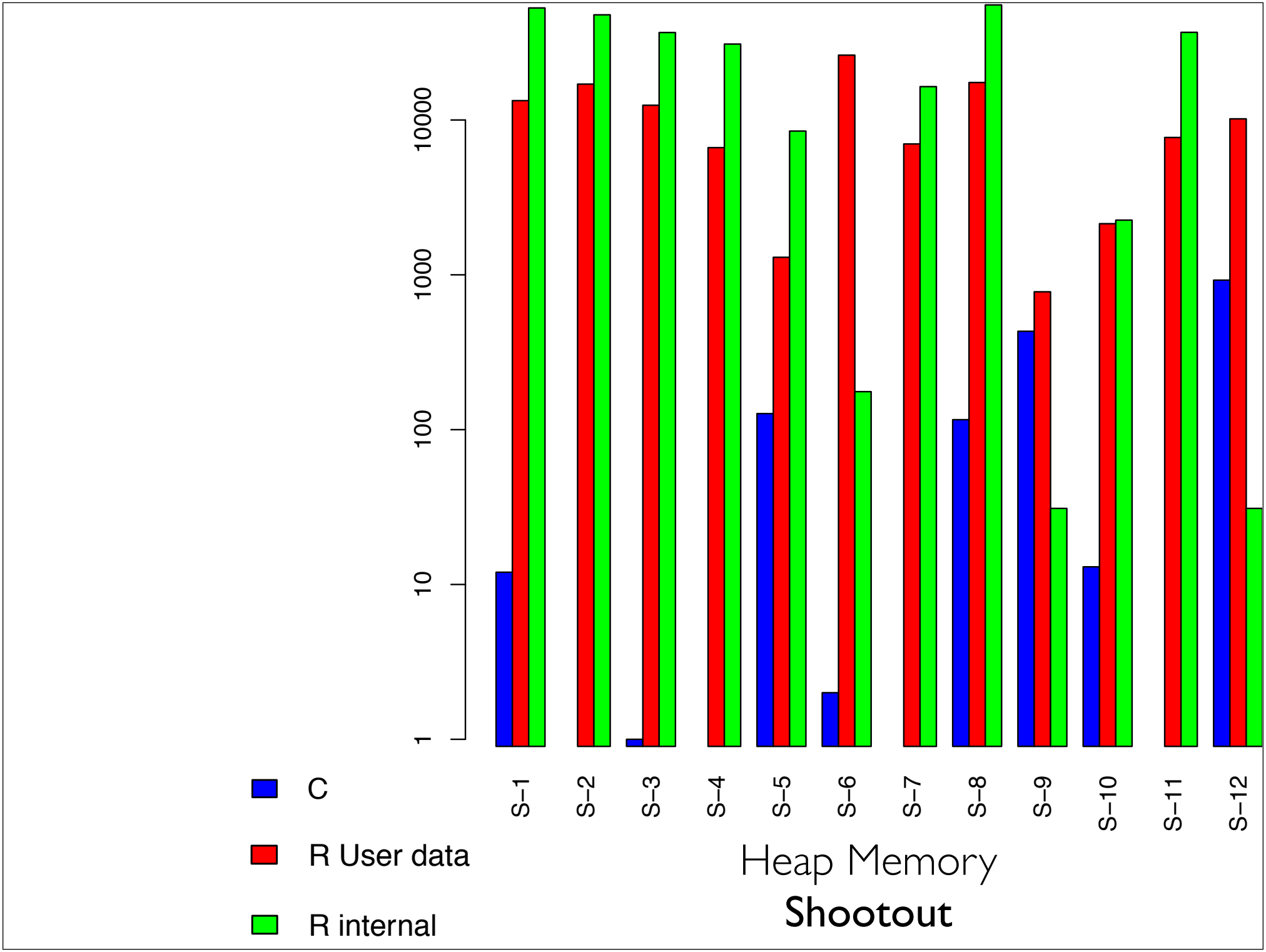
Scaling up…

Current limitations of R on a single node:

- Speed
- Memory footprint
- Limited support for concurrency


Python
R
500
50
10
5
1
S–1
S–2
S–3
S–4
S–5
S–6
S–7
S–8
S–9
S–10
S–11
S–12
Performance relative to C
Shootout

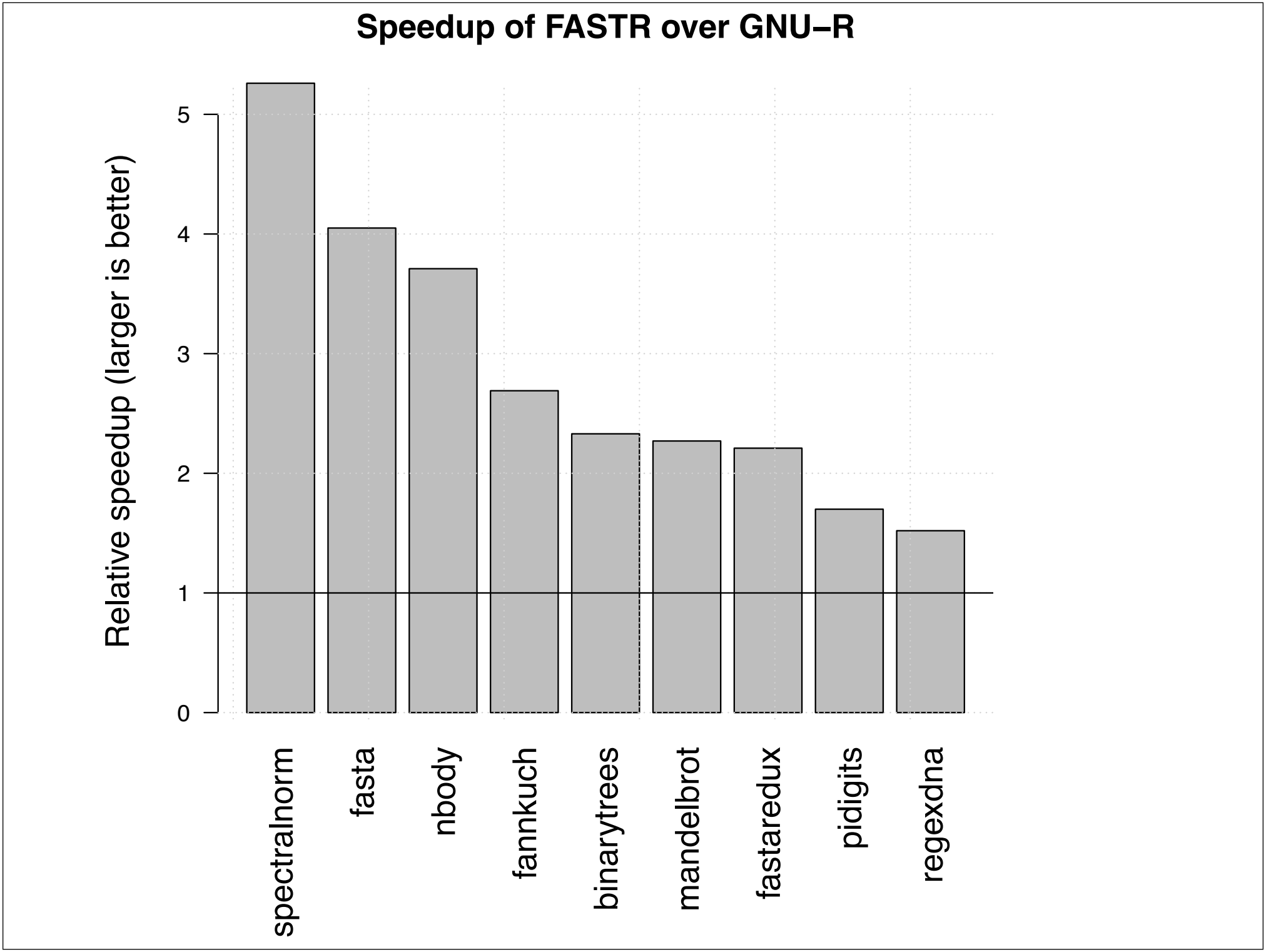

0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
mm
alloc.cons
alloc.list
alloc.vector
duplicate
lookup
match
external
builtin
arith
special


1
10
100
1000
10000
S–1
S–2
S–3
S–4
S–5
S–6
S–7
S–8
S–9
S–10
S–11
S–12
C
R User data
R internal
Heap Memory
Shootout

FastR status

FastR is a new R virtual machine written in Java

- Aims for compatibility & completeness
- Abstract syntax tree interpreter (80% complete for core language)
- LLVM JIT compiler (30% complete)
- Substrate VM (10% complete)


Speedup of FASTR over GNU–R
Relative speedup (larger is better)
0
1
2
3
4
5
spectralnorm
fasta
nbody
fannkuch
binarytrees
mandelbrot
fastaredux
pidigits
regexdna

O2

O2 is self-organizing computational cloud for analytics.

- Written in Java for portability and ease of deployment
- Provides BigVectors as arraylets that can be distributed, moved, and swapped to disk
- Provides a Distributed Fork/Join framework for both local and remote concurrent computation

Distributed F/J

```
for (int i : ntrees)
 trees[i] = new Tree(_data,maxDepth,...);
DRemoteTask.invokeAll(trees);
print("Trees done in "+ timer);
```

Single node Random Forest (O2 v Fortran/R)

Tree build time

data	rows	size	avg tree sz	F	J
iris	.15K	8KB	8	2ms	8ms
chess	196K	3.7MB	8	140ms	200ms
stego	7.5K	11MB	557	440ms	2.4s
kaggle/cs	100K	4.3MB	5321	420ms	1s
kaggle/as	580K	1.7GB	45894	--	25s
covtype	8.7M	72MB	95393	--	3s

Distributed random forest in 3K lines of Java on O2

Conclusions

- Scaling data analytics is about making it easier to turn idea into software
- It requires an integrated infrastructure that leverage advances in programming languages and compilers technology with a deep understanding of the domain.
- Interactive exploration and time to solution are the most important factors