FastR: Status and Outlook

Michael Haupt
Tech Lead, FastR Project
Virtual Machine Research Group, Oracle Labs
June 2014
Safe Harbor Statement

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.
R Roundup

• Things cool about R
  – Open-source code and libraries
  – Ease of use, great DSL for statistics

• Bottlenecks
  – Performance out of the box
  – Database interaction

• Challenges and possibilities
  – “Big data” contexts
  – Heterogeneous computing resources
Oracle R Enterprise (ORE)

Transparency Layer

```r
> aggdata <- aggregate(ONTIME_S$DEST, +   by=list(ONTIME$DEST), FUN=length)
> class(aggdata)
[1] “ore.frame”
Attr(,”package”)
[1] “OREbase”
> head(aggdata)
  Group.1 x
  0  ABE 237
  1  ABI  34
  2  ABQ 1357
...
```
Oracle R Enterprise (ORE)

Parallel Execution

```
modList <- ore.groupApply(X=ONTIME_S, INDEX=ONTIME_S$DEST, function(dat) {
  lm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
})
modList_local <- ore.pull(modList)
summary(modList_local$BOS) # return model for BOS
```
Considerations

• R is a great language for statistics. Why resort to C and Fortran?
• R features inherent parallelism. Why implement it on top?
FastR

- **Open-source R implementation**
  - GPL 2
  - [https://bitbucket.org/allr/fastr](https://bitbucket.org/allr/fastr)
  - Research prototype
  - Linux, Mac

- **Characteristics**
  - Implemented in “100 % Java”
  - With *Truffle* (interpreter) and *Graal* (dynamic compiler)

- **Collaborations**
  - Purdue U (Jan Vitek)
  - JKU Linz (Hanspeter Mössenböck)
  - TU Dortmund (Peter Marwedel)
  - UC Davis (Duncan Temple Lang)
  - U Edinburgh (Christophe Dubach)
Truffle and Graal

Node Transitions: Specializing for Types

Node Rewriting to Update Profiling Feedback

Node Rewriting for Profiling Feedback

Compilation using Partial Evaluation

Deoptimization to AST Interpreter

Compiled Code
FastR: Shifting Performance and Linguistic Boundaries
FastR Deployment Models

**Stock HotSpot™**
- Purely interpreted, no compilation
- Performance drawbacks

**GraalVM**
- Interpretation + compilation
- Full performance advantages

**Substrate VM**
- Bootstrap to get stand-alone binary or shared library
- Interpretation + compilation
- Performance advantages
- Embeddable R execution environment

---

**Diagram:**
- Stock HotSpot: R -> FastR -> HotSpot
- GraalVM: R -> FastR -> Truffle -> GraalVM (HotSpot+Graal)
- Substrate VM: FastR -> Truffle -> Graal -> SVM -> Graal -> HotSpot

---

*Oracle*
FastR: Status and Outlook

• Details
  – Ca. 51k LOC (and growing)
  – 4870 tests, 651 failing (13 %)
  – 7580 bulk arithmetic tests, none failing

• This year: completeness
  – Load selected CRAN packages
  – Execute “real-world” code

• Next year: transparent scalability
  – Threads, GPUs

Acknowledgments

Oracle Labs
Michael Haupt (tech lead)
Mick Jordan
Roman Katerinenko
Gero Leinemann (intern)
Adam Welc
Christian Wirth
Mario Wolczko
Thomas Würthinger

Oracle
Mark Hornick

JKU Linz
Christian Humer
Hanspeter Mössenböck
Andreas Wöß

TU Dortmund
Ingo Korb
Helena Kotthaus
Peter Marwedel

Purdue University
Rohan Barman
Dinesh Gajwani
Prahlad Joshi
Cameron Kachur
Di Liu
Leo Osvald
Simon Smith
Roman Tsegelskyi
Jan Vitek
Adam Worthington

UC Davis
Duncan Temple Lang
Nicholas Ulle

University of Edinburgh
Christophe Dubach
Juan José Fumero
Hardware and Software
Engineered to Work Together