ORACLE®

FastR

Michael Haupt
Tech Lead, FastR Project
Virtual Machine Research Group, Oracle Labs
September 2014

ORACLE

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 is ...

- A programming language
  - Many built-in statistical functions and data types
  - Very high abstraction for common statistical tasks
  - A DSL for statistics
  - Also, a general-purpose array programming language: ability to implement algorithms, analyses

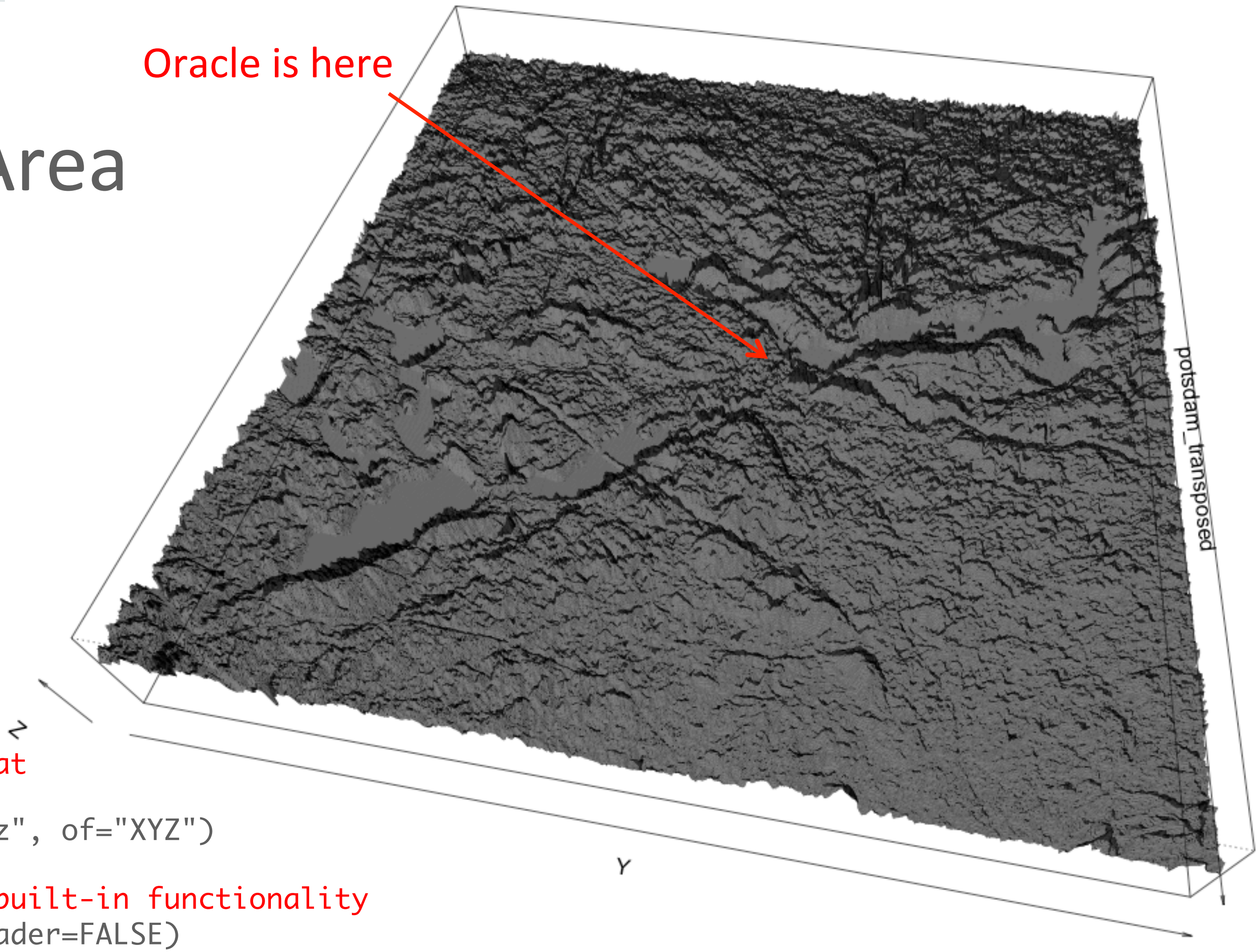
- A statistics workbench
  - Data exploration and manipulation
  - Graphics capabilities for visualizing data
  - Interactions with typesetting systems and web servers for data presentation

- A data science ecosystem
  - Vast open-source package repositories for multiple purposes
  - Roughly 5,800 packages in CRAN, 850 in Bioconductor
  - Application areas: statistics, geoscience, bioinformatics, health sciences, machine learning, ...

R is the *lingua franca* for data science.

Visualizing the Potsdam Area

geospatial data (potsdam_dem.tif)




```
> # use package to transform data into tabular format
> library(gdalUtils)
> gdal_translate("potsdam_dem.tif", "potsdam_dem.xyz", of="XYZ")
NULL
> # no package support needed from here on – all R built-in functionality
> potsdam_table <- read.table("potsdam_dem.xyz", header=FALSE)
> names(potsdam_table) <- c("longitude", "latitude", "elevation")
> n_rows <- length(levels(as.factor(potsdam_table$longitude)))
> potsdam_matrix <- matrix(potsdam_table$elevation, nrow=n_rows)
> potsdam_transposed <- t(potsdam_matrix)
> persp(potsdam_transposed, r=0.5, col=grey(.25), border=NA, expand=.15, theta=100, phi=50, ltheta=70, lphi=10, shade=3)
```

Common R Workflows

Explorative Data Analysis

1. Load the data
   - From a flat file, database, or other source
   - Possibly store them in R-specific format for later reuse
2. Massage the data
   - Identify the relevant subset: search, filter, "ask questions"
   - Bring the data into the right shape
3. Visualize the data
   - Choose the right kind of visualization and apply it (it's usually directly possible)
4. Interpret the results
   - Possibly go back to step 2 to ask more questions and visualize again

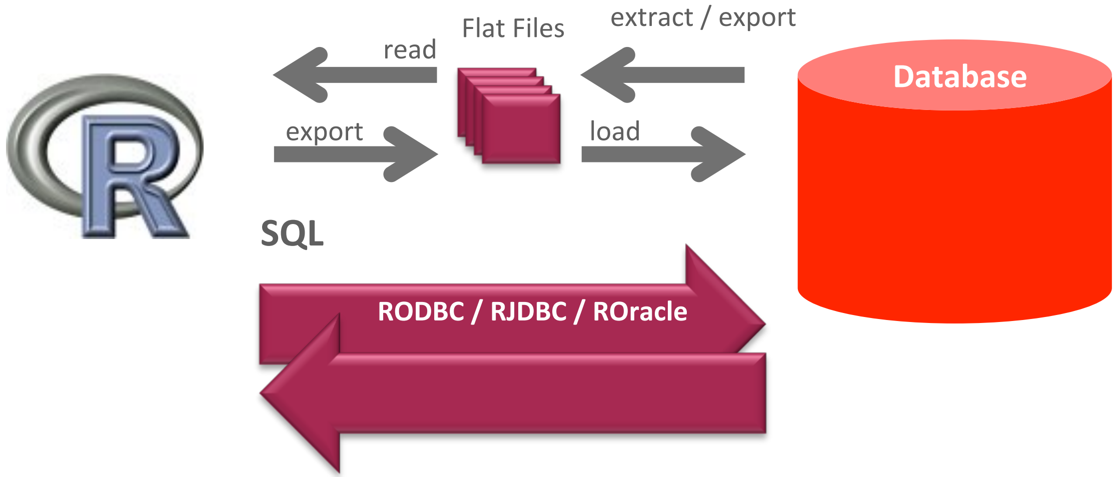
Implementing Analytic Algorithms

1. Implement algorithm in R
   - Performance problems will arise
2. Address critical paths
   - Identify the "expensive inner loops" that are performance bottlenecks
   - Port them to C/C++ and call them through the R FFI
3. Possibly parallelize
   - Identify parts of the code that is parallelizable
   - Adopt existing R libraries for multithreading, GPU usage

ORACLE

R Roundup

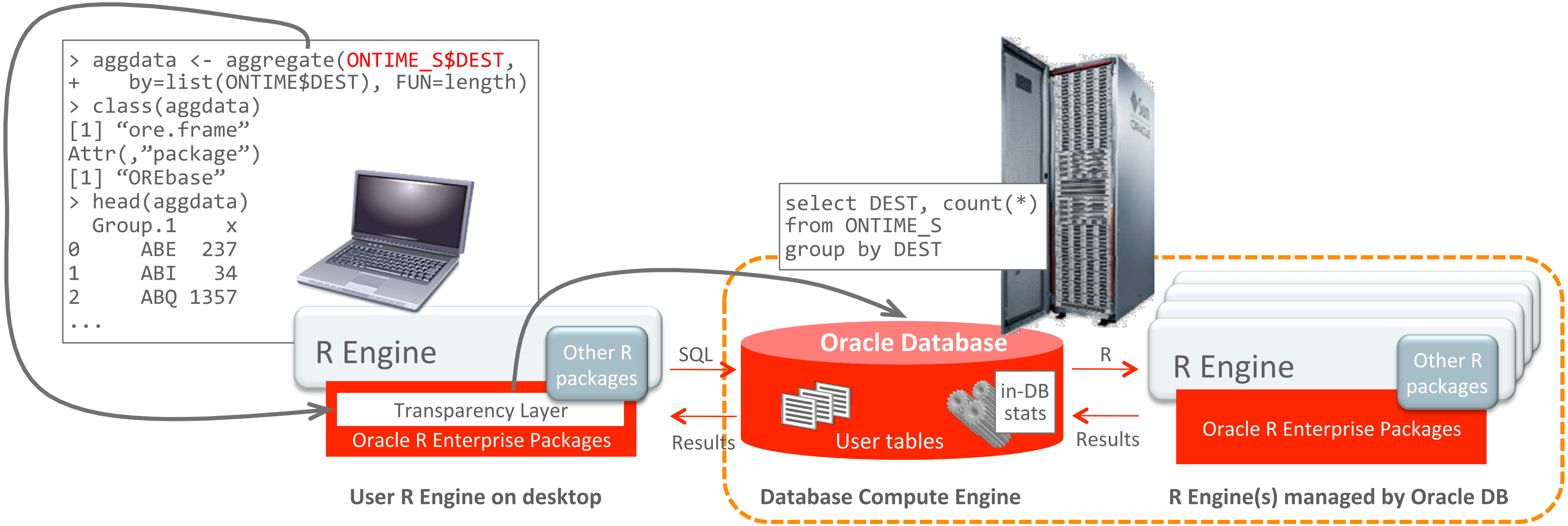
- Things cool about R
  - Open-source code and libraries
  - Ease of use, DSL for statistics
- Bottlenecks
  - Performance out of the box
  - Database interaction
- Challenges and possibilities
  - "Big data" contexts
  - Heterogeneous computing resources



R logo copyright © R Foundation, from http://www.r-project.org/

Oracle R Enterprise (ORE)

Transparency Layer

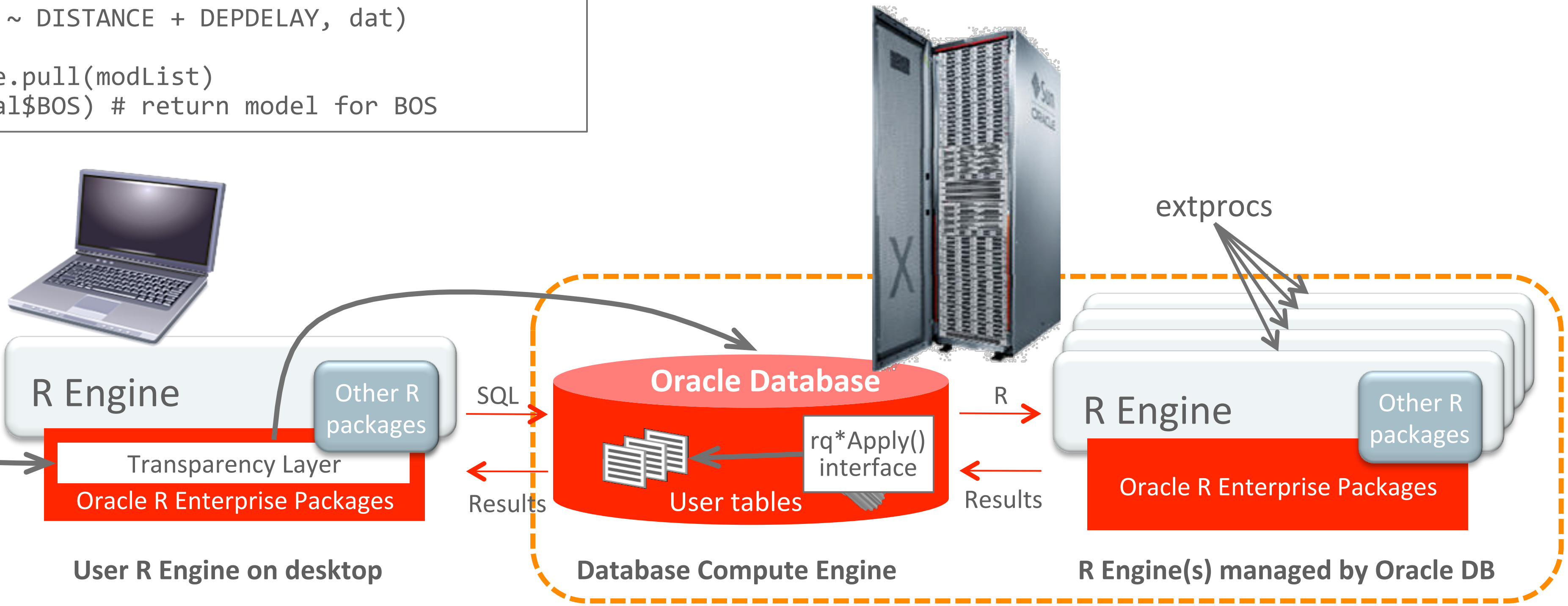


ORACLE

Oracle R Enterprise (ORE)

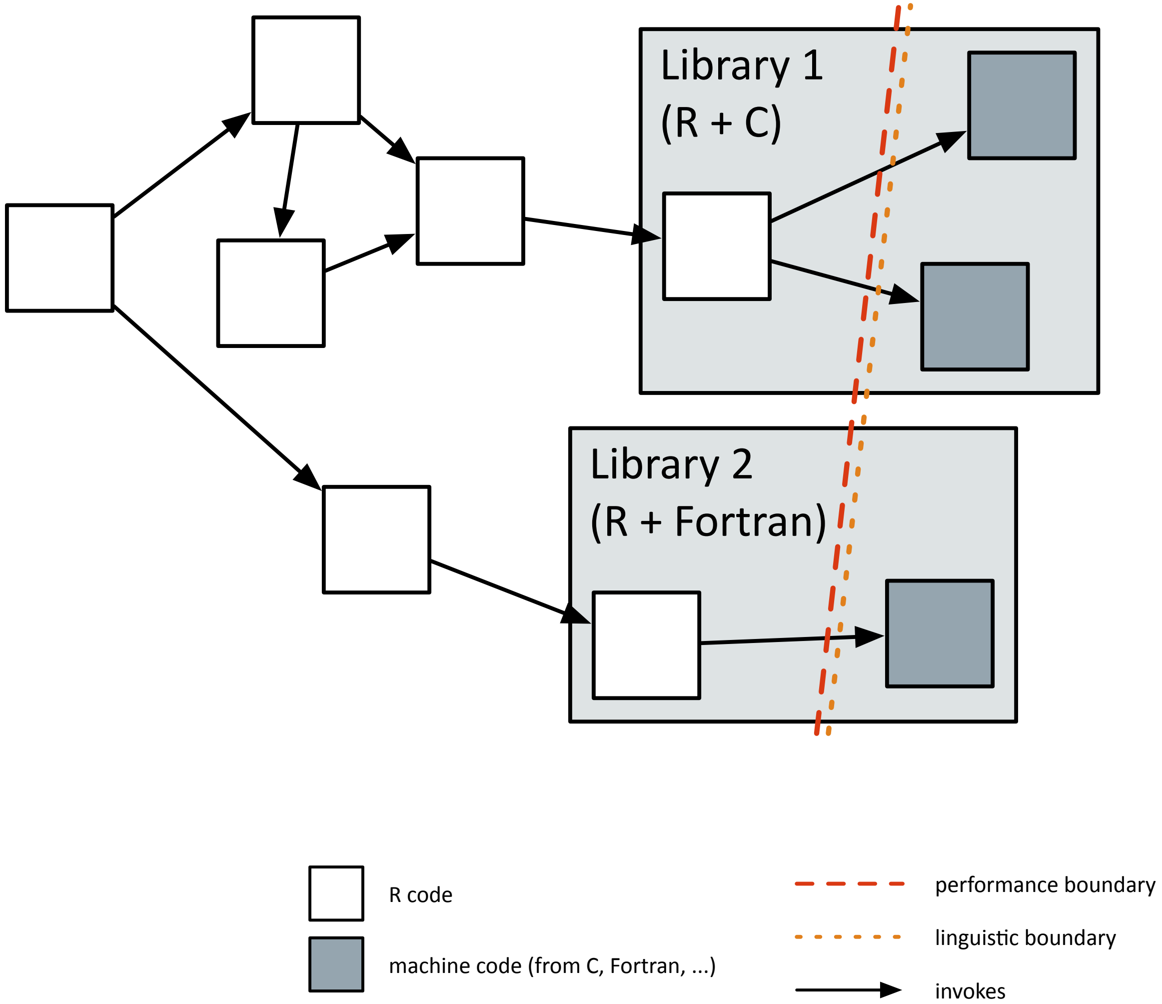
Parallel Execution

```
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?

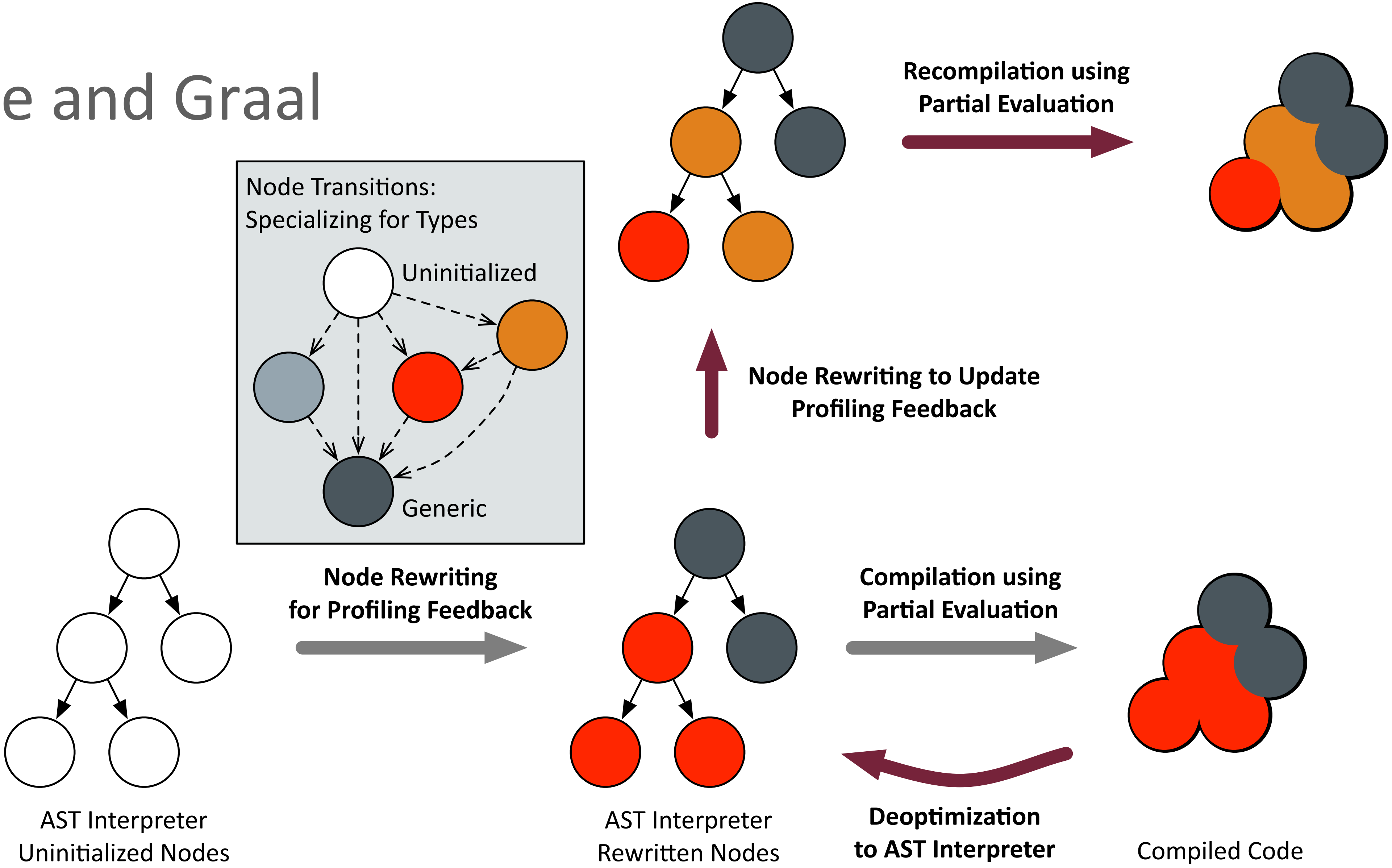


FastR

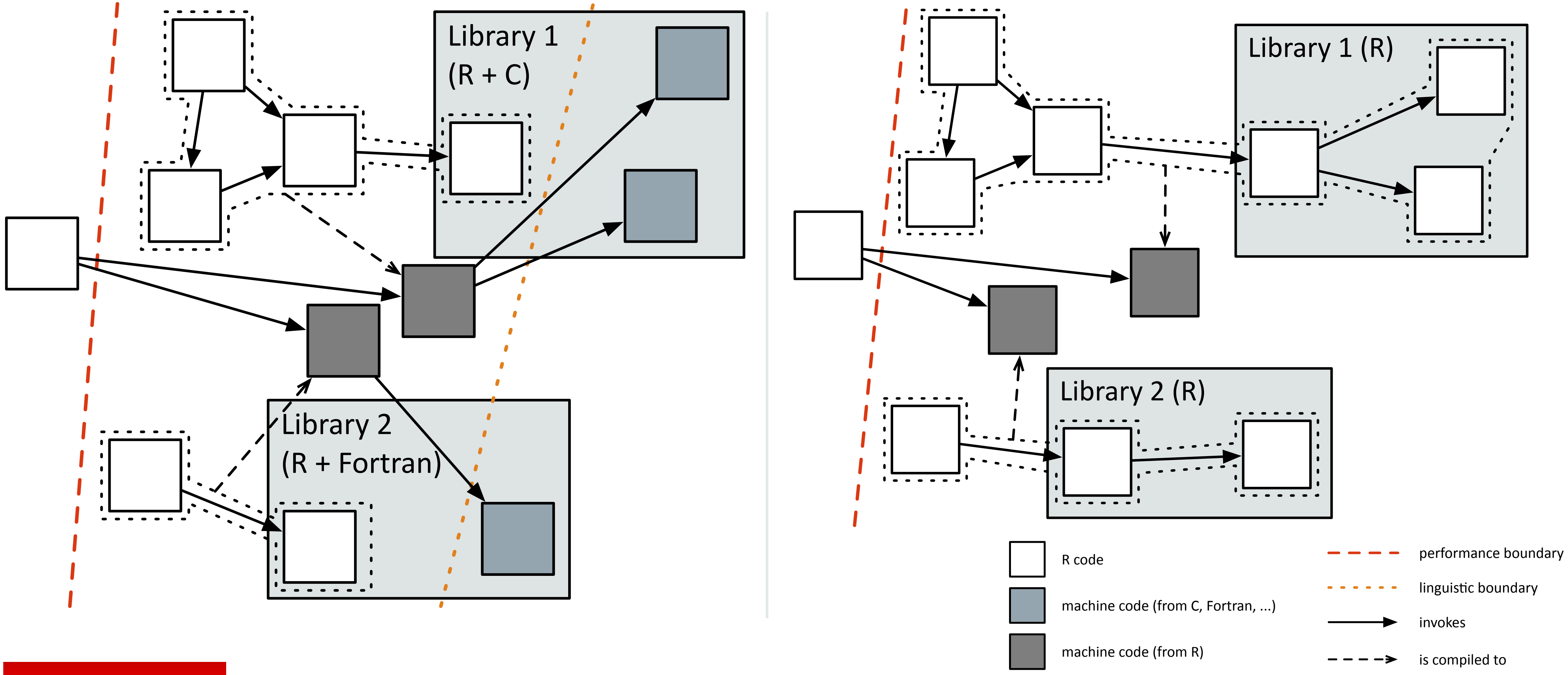
- Open-source R implementation
  - GPL 2
  - https://bitbucket.org/allr/fastr
  - Research prototype
  - Linux, Mac
- Characteristics
  - Implemented in “100 % Java”
  - With *Truffle* (interpreter)
    and *Graal* (dynamic compiler)
- Collaborations
  - Purdue U (Jan Vitek, Tiark Rompf)
  - JKU Linz (Hanspeter Mössenböck)
  - TU Dortmund (Peter Marwedel)
  - UC Davis (Duncan Temple Lang)

ORACLE

Truffle and Graal



FastR: Shifting Performance and Linguistic Boundaries

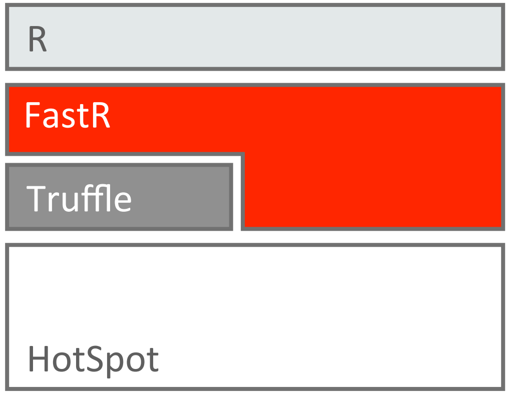


ORACLE

FastR Deployment Models

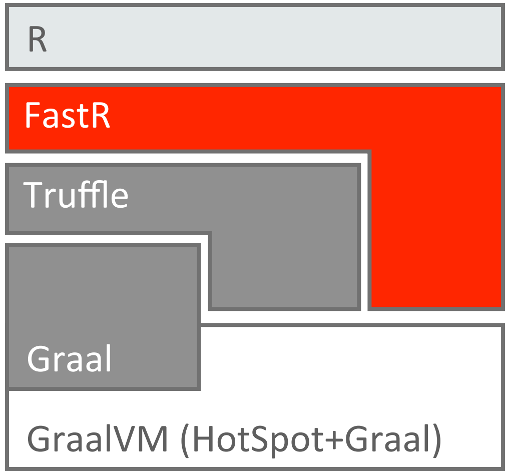
Stock HotSpot™

- Purely interpreted, no compilation
- Performance drawbacks



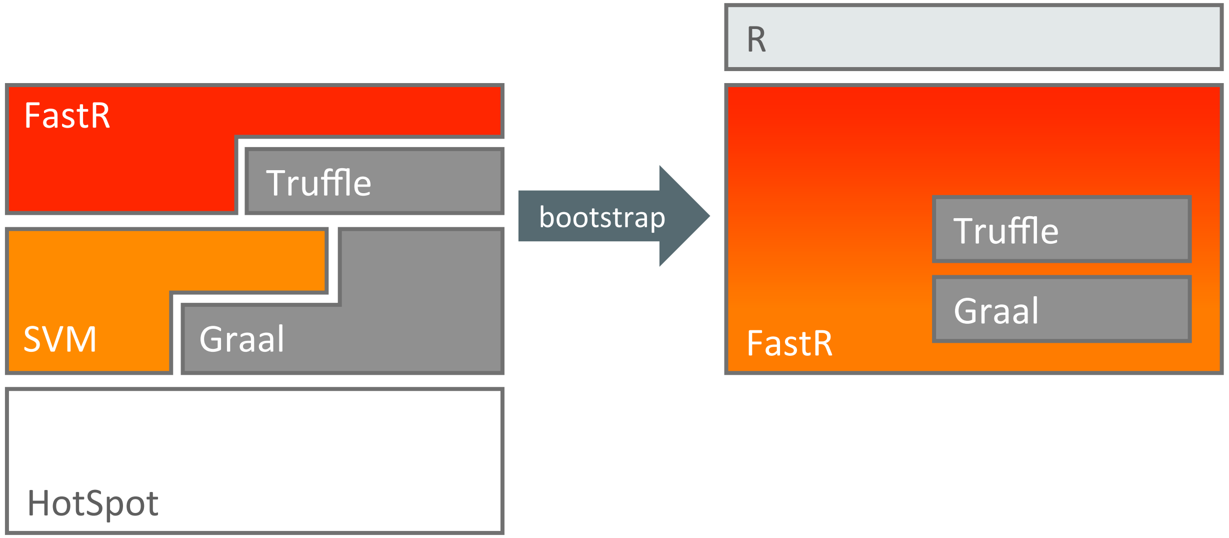
GraalVM

- Interpretation + compilation
- Full performance advantages



Substrate VM

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



FastR: Status and Outlook

- Details
  - Ca. 57k LOC (and growing)
  - Ca. 9,000 tests, 66 % pass
  - Plus ca. 7,600 bulk arithmetic tests, all pass
- 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)
Lukas Stadler
Adam Welc
Christian Wirth
Mario Wolczko
Thomas Würthinger

Oracle
Mark Hornick

JKU Linz
Christian Humer
Hanspeter Mössenböck
Andreas Wöß

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

TU Dortmund
Ingo Korb
Helena Kotthaus
Peter Marwedel

UC Davis
Duncan Temple Lang
Nicholas Ulle

ORACLE

Hardware and Software

Engineered to Work Together

ORACLE®