GRAALVM / FASTR AND RENJIN

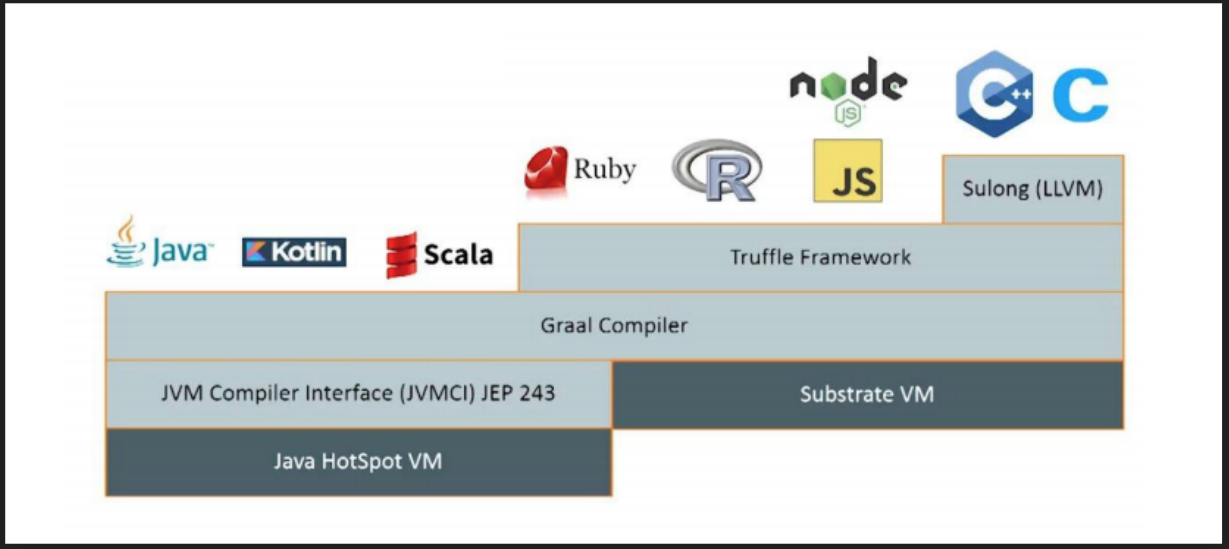
Twin Cities R Users Group

Mar 28, 2019

GRAALVM / FASTR

- R language implementation inside GraalVM
- GraalVM - polyglot VM
  - Java8 / JVM-languages, JavaScript - primary
  - Ruby, R, Python - "experimental"
  - Any LLVM language - C/C++, Fortran, Julia, etc.
- Able to create native images
  - Specify which language support is needed
  - Used by Twitter for their Docker-based services
- Embeddable in Oracle/MySQL DB
- Oracle as main sponsor - graal Github & fastr Github

GRAALVM ARCHITECTURE



https://www.tomitribe.com/codeone/dev6016/

WHY GRAALVM / FASTR?

- faster, secure, distributed, portable, tool support
  - JIT is slower to optimize but much faster
  - native images are fast and small
- polyglot capabilities - any to any
- FastR compatible with GNU R 3.5.1
  - provides R and Rscript as entry points
  - native R performance close to Java
  - C/C++/Fortran/rJava integration is faster
  - Some (few) packages known not to work
  - IDE support varies (no RStudio)

INSTALLING GRAALVM / FASTR

- Linux and MacOS - from Oracle (CE/EE)
  - unzip, set `GRAALVM_HOME` and `JAVA_HOME`
  - add `GRAALVM_HOME/bin` to `PATH`
  - JDK tools, JS tools (`js, node, npm`), and
    - `gu, native-image, lli, polyglot`
- Install FastR with `gu install R`
  - Installs to `GRAALVM_HOME/jre/languages/R`
  - requires OpenMP library and C/C++/Fortran compilers
  - Use `configure_fastr` script to setup `GRAALVM_HOME/jre/languages/R/etc/Makecon`

USING FASTR - PACKAGES

- `install.packages` works likes normal
- `getOption("repos")` points to MRAN 2019-02-13
- `.libPaths()` points to FastR install dir
- `install.fastr.packages` for special packages
  - `data.table, rJava, refcmp, graalvm`
- There is an online package compatibility check
- grid-based graphics works (lattice, ggplot2,...)
- parallel has SHARED cluster type for multi-threading

USING FASTR - INTEROP

- Polygot API available to all languages
- All examples (from/to) can be found online

```
import org.graalvm.polyglot.*;

class Polyglot {
  public static void main(String[] args) {
    Context polyglot = Context.newBuilder().
                       allowAllAccess(true).build();
    Value array = polyglot.eval("R", "c(1,2,42,4)");
    int result = array.getArrayElement(2).asInt();
    System.out.println(result);
  }
}
```

USING FASTR - R CALLING *

- polyglot features
  - eval.polyglot('lang','code')
  - eval.polyglot(path = '/path/to/code.ext')
  - export('polyglot-value-name', rObject)
  - import('polyglot-value-name')
- JVM features - java.type - to get class and new method

```
x <- eval.polyglot('ruby', '[1,2,3]')
print(x[[1]])
# [1] 1

x <- eval.polyglot(path='r_example.js')
print(x$a)
# [1] "value"

export('robj', c(1,2,3))
eval.polyglot('js', paste0(
    'rvalue = Polyglot.import("robj"); ',
    'console.log("JavaScript: " + rvalue.length);'))
# JavaScript: 3
# NULL -- the return value of eval.polyglot
```

RENJIN

- R language implementation on a JVM
  - Compiles R to JVM bytecode
  - Packages as jars, Maven-based repository
- Compiles C/C++/Fortran to JVM bytecode
  - Makes use of Gimple & the GCC-Bridge
- Makes R a JVM language (jar+JVM)
- BeDataDriven as main sponsor
  - Commercial support and used by them
  - Hosts the package repository & renjin Github

WHY RENJIN?

- Similar reasons to GraalVM / FastR
  - more portable than FastR
  - almost always embedded in a JVM language app
  - not as much support and tooling
- Public Maven repository for R packages
  - Supports multiple versions of packages
  - Build packages via Maven with a pom.xml file
  - If package doesn't work - you're a bit stuck
- Support for both CRAN and BioConductor
- `renjin` package for use on GNU R for optimization

INSTALLING RENJIN

- Typically added to a Java application
  - Can add to a project via Maven, Gradle, sbt
  - Can include in a Spark über-jar
  - Same for all "package" jars
- Linux and MacOS - from renjin.org
  - unzip, set `RENJIN_HOME`, add to `PATH`
  - install via apt (Debian / Ubuntu)
  - `renjin` starts the REPL
- Renjin Studio is an executable JAR

USING RENJIN - PACKAGES

- `install.packages` does not work
- `library` will download if not there
- Your library is your local Maven repository
- You can also install via Maven
- Package versions get a Maven ID
  - "groupId" is `org.renjin.cran`
  - "artifactId" is R package name
  - "version" is R package version and Renjin build
- There is an online package portal

USING RENJIN - EXECUTING R

- Makes use of `javax.scripting` API/SPI
- Renjin provides ways to send data in and out

```
RenjinScriptEngineFactory factory = new RenjinScriptEngineFactory();
ScriptEngine engine = factory.getScriptEngine();

engine.eval("df <- data.frame(x=1:10, y=(1:10)+rnorm(n=10))");
engine.eval("print(df)");
engine.eval("mdl <- lm(y ~ x, df)");

Vector gammaVector = (Vector)engine.eval("mdl$gamma");
double gamma = gammaVector.getElementAsDouble(0);
```

USING RENJIN - EXECUTING JAVA

- Renjin adds an `import` function
- Renjin converts JavaBean syntax to R

```
import(java.util.HashMap)
import(beans.Customer)

ageMap <- HashMap$new()
ageMap$put("Bob", 33)
ageMap$put("Carol", 41)
print(ageMap$size())

age <- ageMap$get("Carol")
cat("Carol is ", age, " years old.\n", sep = "")

bob <- Customer$new(name = "Bob", age = 36)
carol <- Customer$new(name = "Carol", age = 41)
cat("bob$name = ", bob$name, "\n", sep = "")
```