Seeking the holy Graal

Presenter: David Lucas

Who am I ?

- Over 25+ years in software industry
- Working with Java since 1998
- IntelliJ / JetBrains introduced me to Kotlin
- Google made me realize …
  Kotlin is now the new Java
- I am a Kotlin Enthusiast
- Extensive production deployments
- Focus mostly on server side solutions
- Experience with Make, Ant, Maven, and new to Gradle

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

LSE

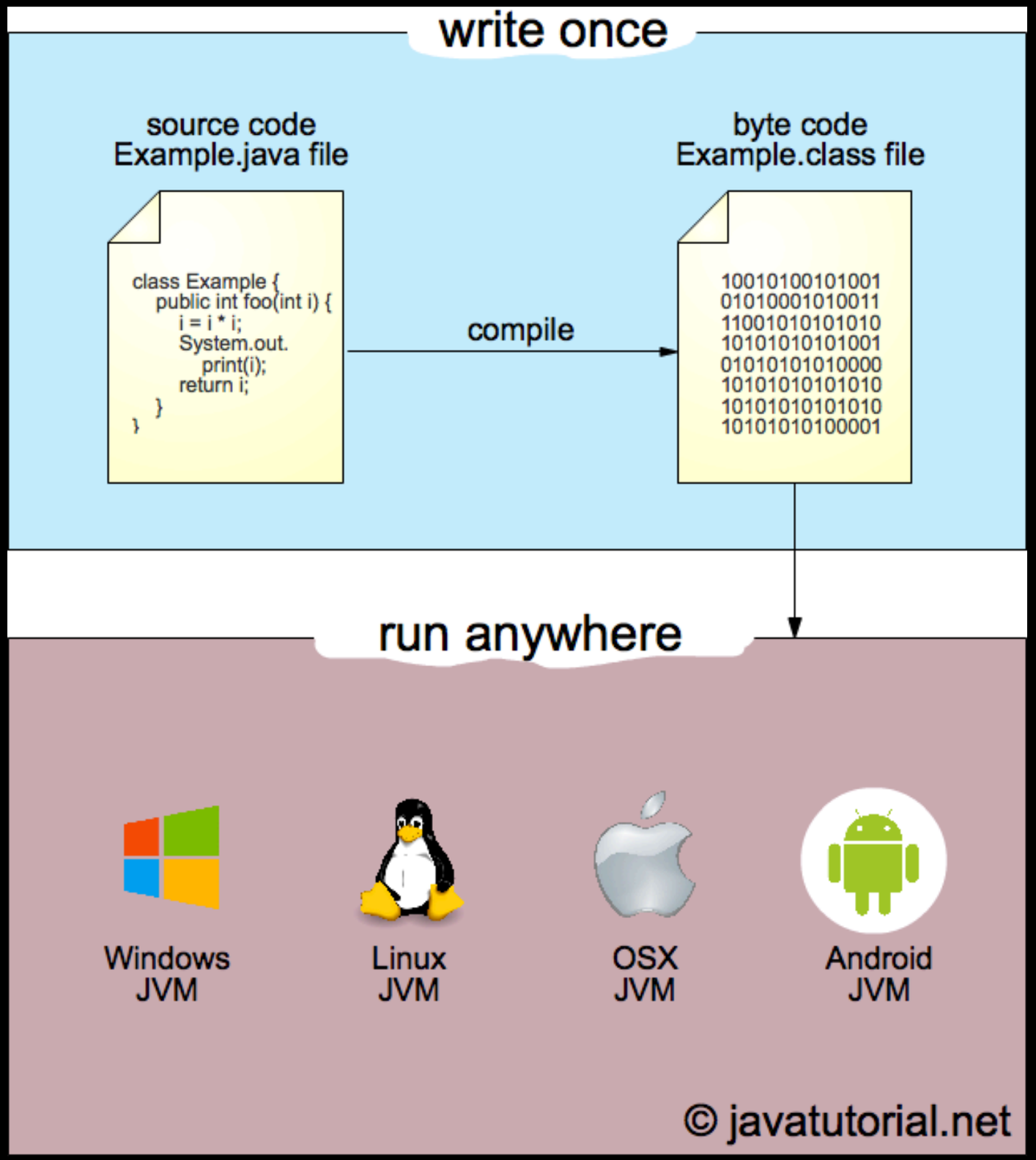
My Agenda

- Not a big fan of polyglot
  (but do realize data scientists care)
- Not a big fan of stored procedures
  (but do realize big data cares)
- My goal is to shrink resource usage for microservices (jar -> exec)
- Alternative for Kotlin Native ?

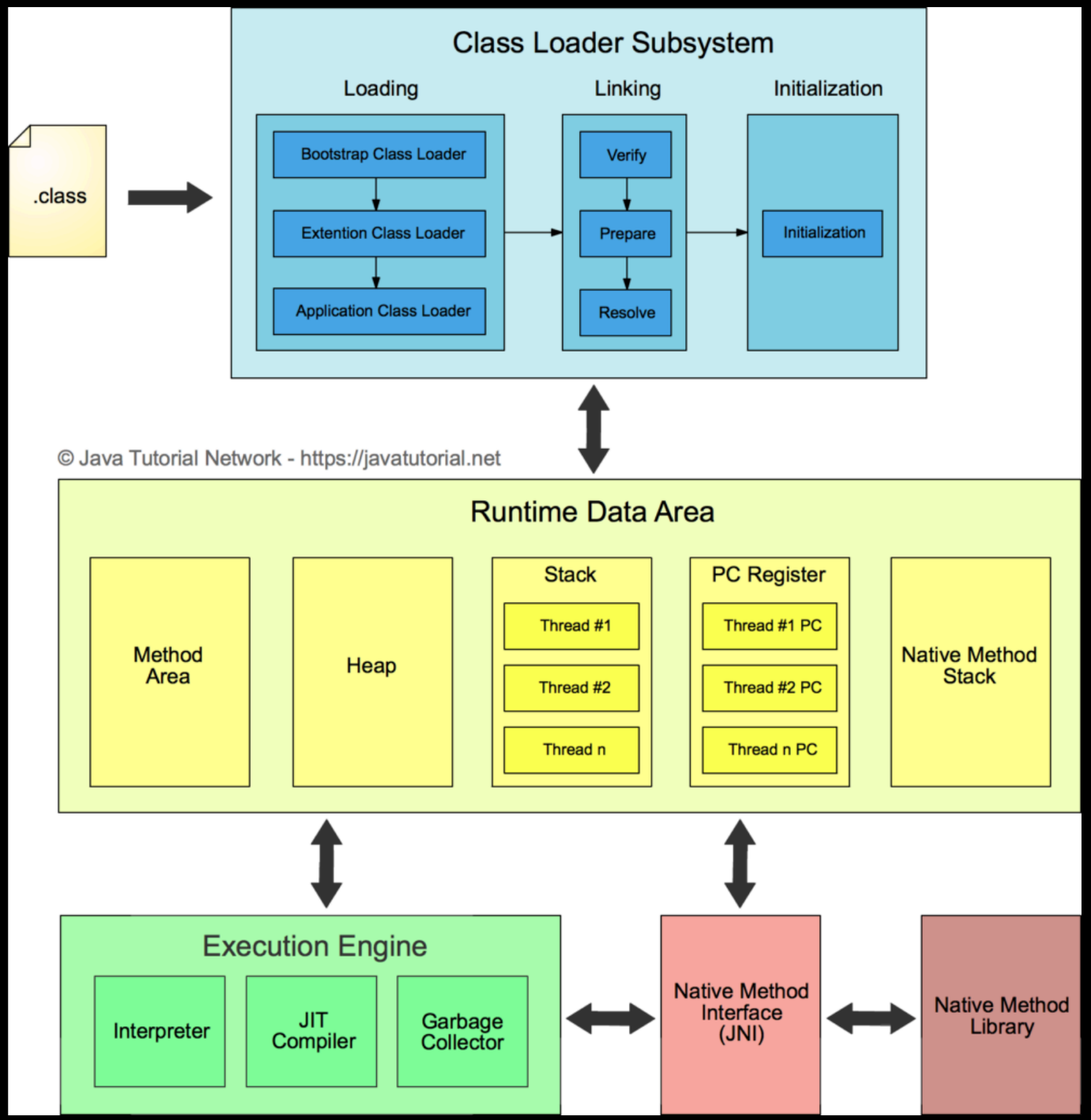
Goals

- Introduce GraalVM
- Demo some capabilities (js, R, rb, py)
- Demo LLVM Interpreter
- Demo Native Images
- Summary

GraalVM Intro: Java Compilation



GraalVM Intro: What is the JVM ?



GraalVM Intro

ORACLE

- Graal is a highly optimized Ahead Of Time (AOT) compiler (JIT=Just In Time)
- GraalVM: result of many projects over decade
- Community Edition is Open Source (GPL v2.0)
- Focus: improve resource usage & performance
- “Make development more productive and run programs faster anywhere” — @graalvm

http://www.graalvm.org/docs/why-graal/

Java™ Scala Kotlin JS Ruby R python™ C C++

Automatic transformation of interpreters to compiler

GraalVM™

Embeddable in native or managed applications

OpenJDK™ node.js ORACLE Database MySQL™ standalone >_

https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM Intro

ORACLE

- Platforms: JVM, Node.js, Native
- Compilers: JavaScript, R, Ruby, Python
- True Polyglot Runtime
  (shared data and functions)
- Easier than JNI
- SubstrateVM for native runtime
- Truffle API for scripting and creating new languages

http://www.graalvm.org/docs/why-graal/

LSE

GraalVM Intro

GraalVM Stack

python™ Ruby R JS Sulong

Scala Kotlin Java™

Truffle Framework

Graal Compiler

ORACLE

LSE

GraalVM Intro

ORACLE

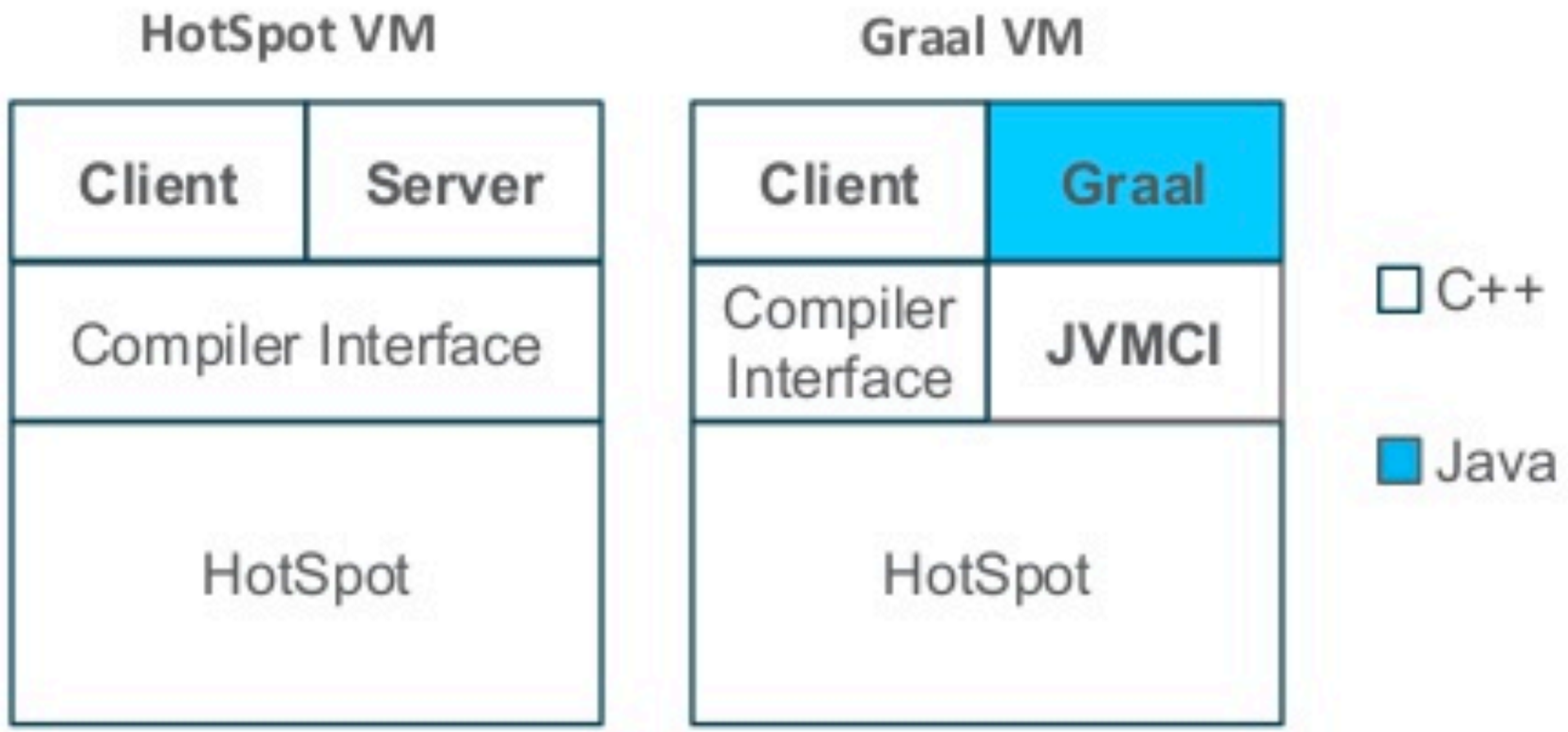
- LLVM bit code can be executed
- Generates shared libraries and executables
- Embedded in Databases
  - Oracle has had a JVM in their database since 2009 for Java Stored Procedures
  - Adding support in MySQL
- Twitter in PROD, tweet service in 2017 (embedded components of GraalVM)

http://www.graalvm.org/docs/why-graal/

LSE

ORACLE

Graal and Graal VM



https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

LSE

GraalVM Intro

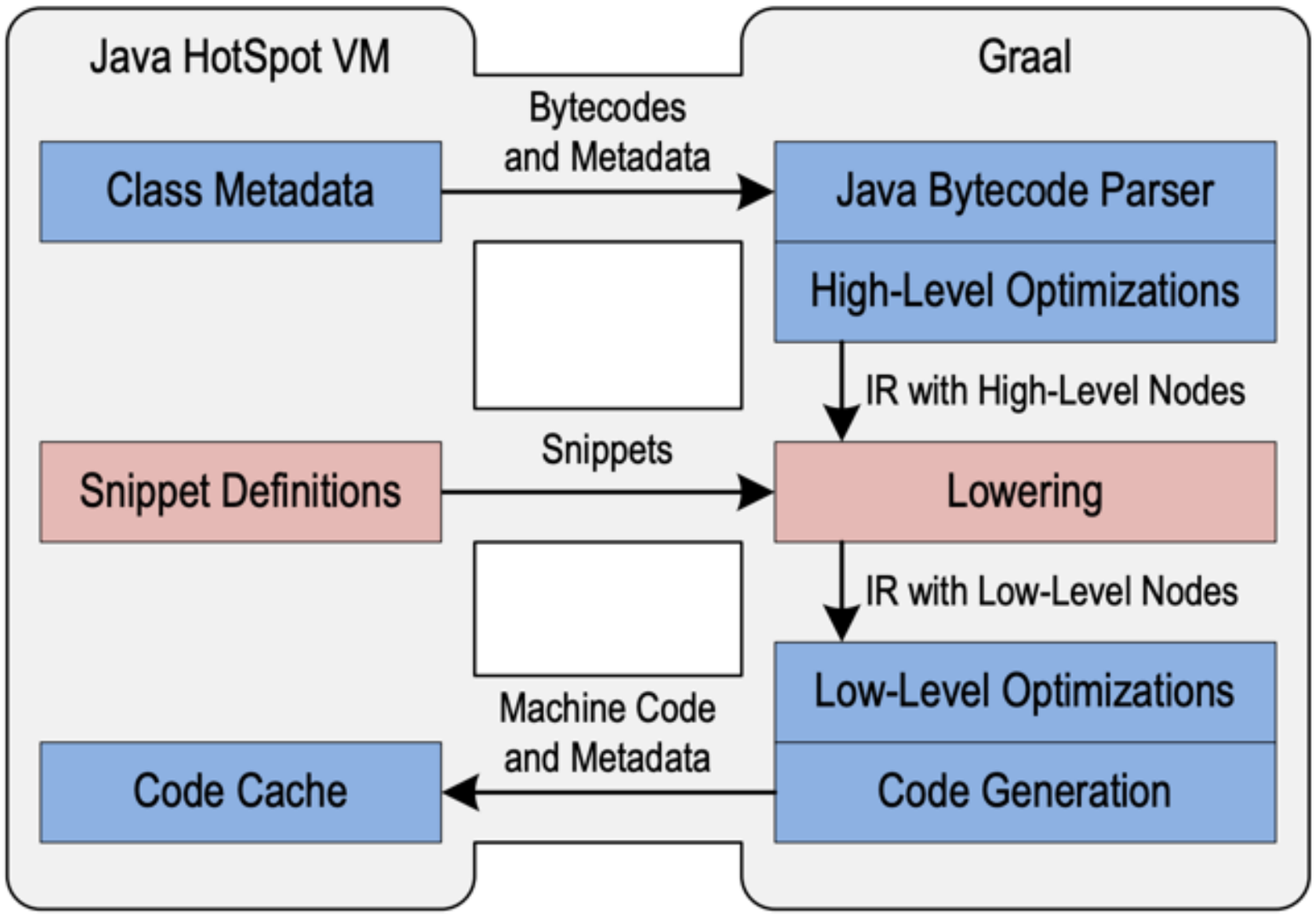
ORACLE

- Ahead-Of-Time (AOT) Compilation (static) into Intermediate Representation (IR)
- Convert IR to native in more optimized fashion
  - Speculates results and references
  - De-optimizes and Re-optimizes
  - Snippets (inlining)
- Performs advance escape analysis and initialization before execution
- Written in Java
- Details on how it creates a smart IR: http://lafo.ssw.uni-linz.ac.at/papers/2013_Onward_OneVMToRuleThemAll.pdf

http://www.graalvm.org/docs/why-graal/

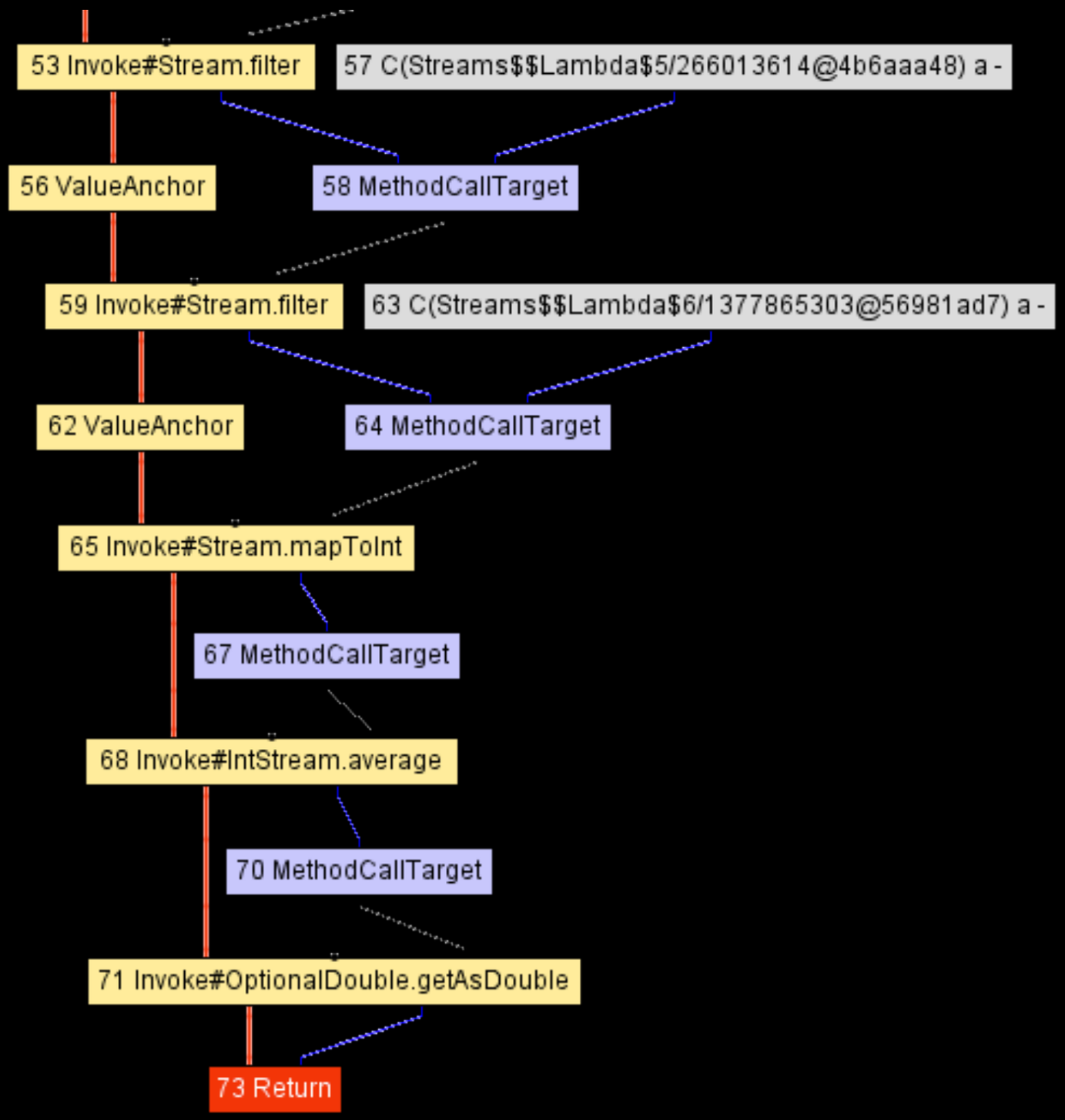
LSE

Compiler-VM Separation



https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM
Intermediate Representation (IR)

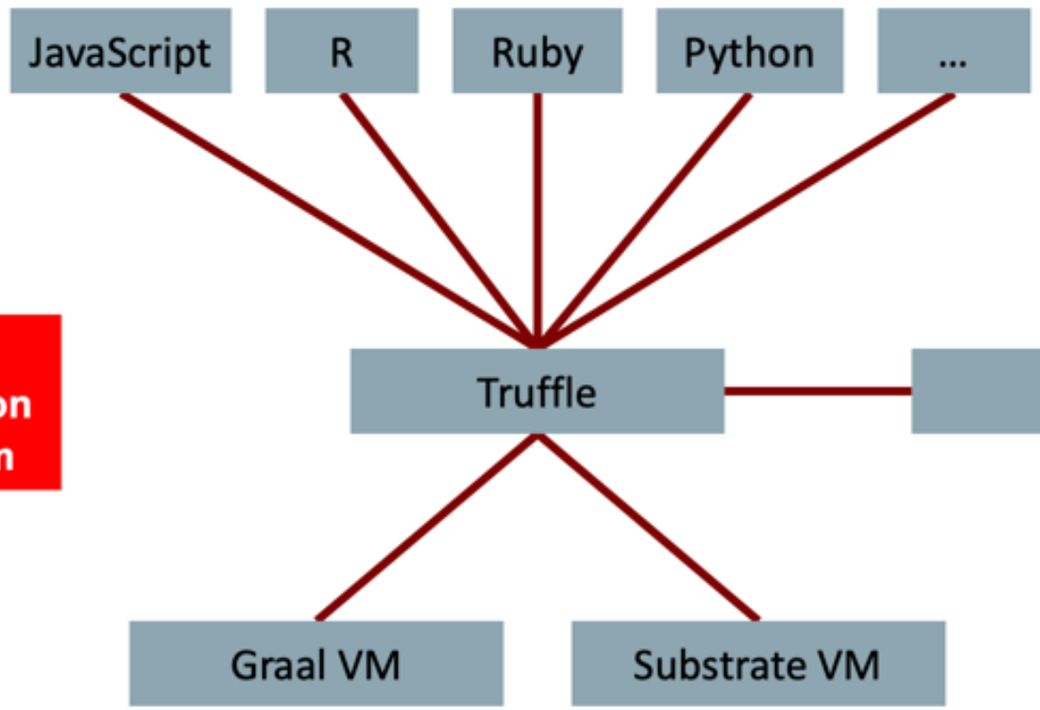


https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

ORACLE

Truffle System Structure

AST Interpreter for every language



Your language should be here!

Common API separates language implementation and optimization system

Language agnostic dynamic compiler

Integrate with Java applications

Low-footprint VM, also suitable for embedding

https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

LSE

GraalVM API

ORACLE

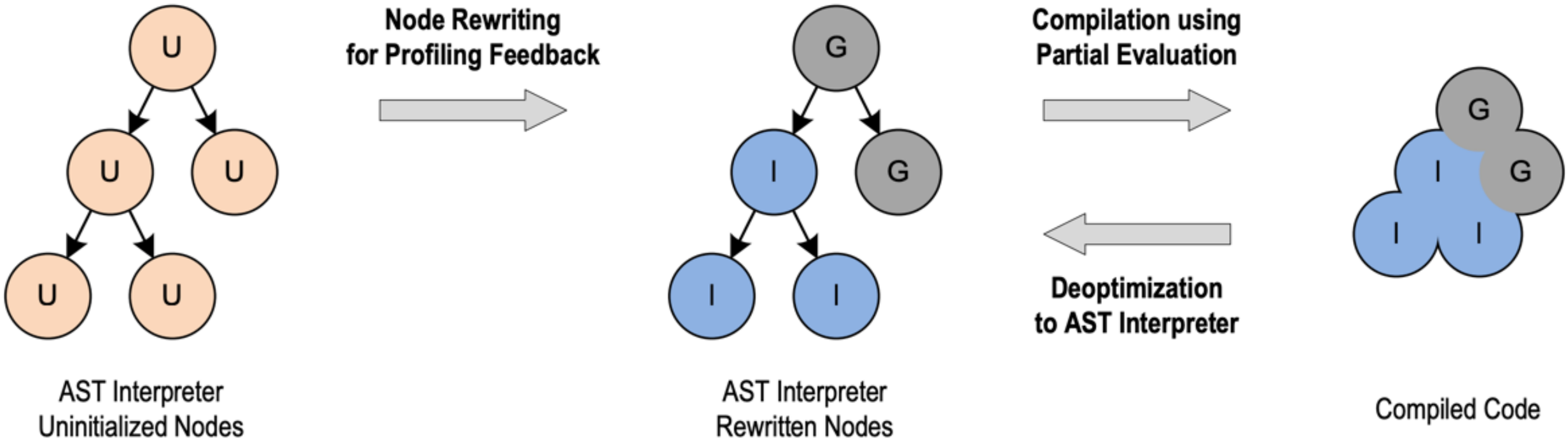
- Truffle API
  - Declarative
  - Abstract Syntax Tree (AST) representation
  - Convert AST into IR
- Written in Java
- Script Engines use Truffle to create AST
- Truffle AST used to generate IR
- IR used to create byte code or native code

http://www.graalvm.org/docs/why-graal/

LSE

ORACLE

Truffle Approach



https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

LSE

Substrate VM

Static Analysis and Ahead-of-Time Compilation using Graal

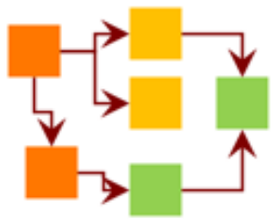
Static Analysis

Java Application

JDK

Substrate VM

All Java classes from application, JDK, and Substrate VM



Reachable methods, fields, and classes

Ahead-of-Time Compilation

Machine Code

Initial Heap

DWARF Info

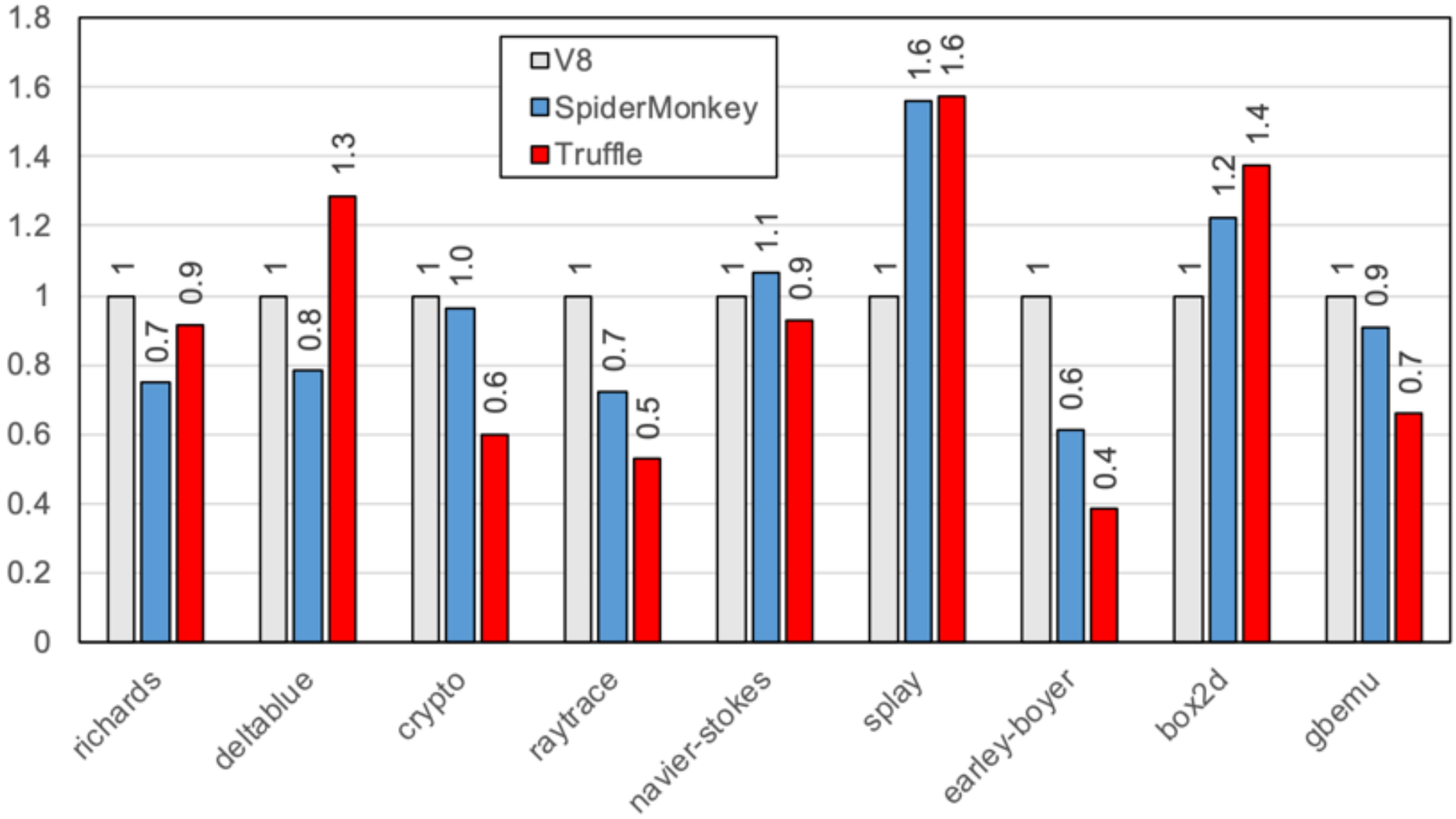
ELF / MachO Binary

Application running without dependency on JDK and without Java class loading

https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

ORACLE

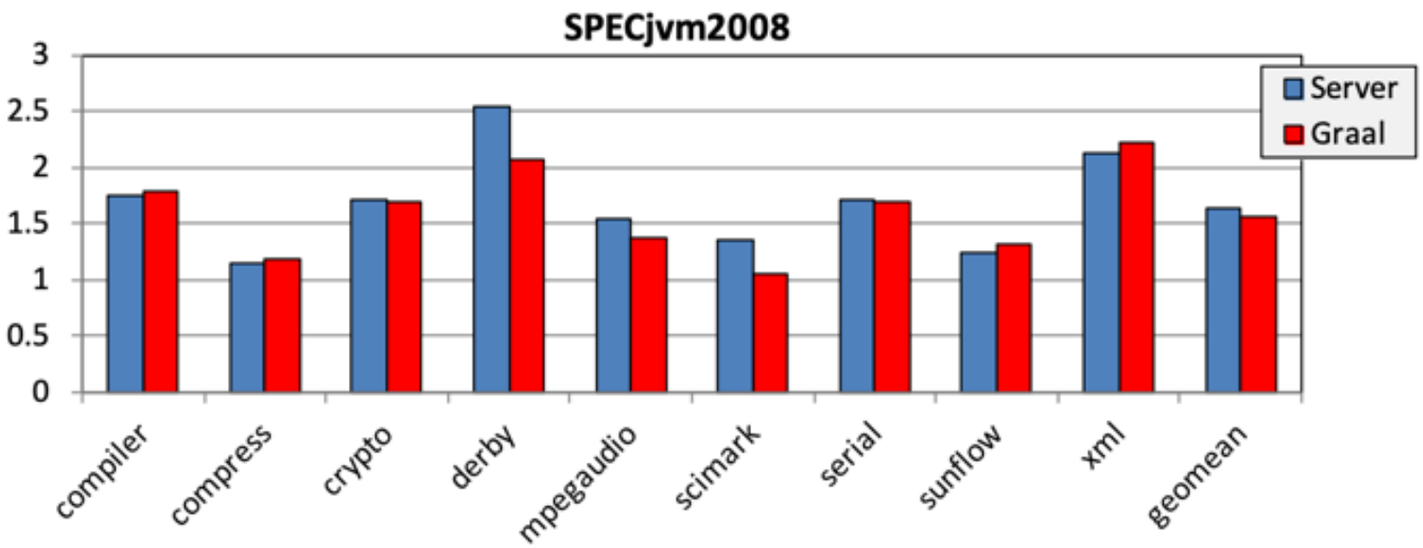
Performance: JavaScript

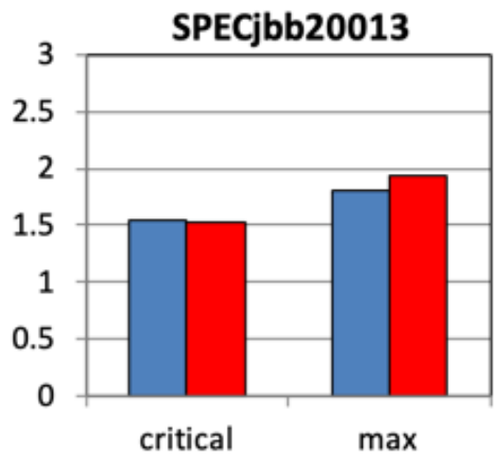


https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

LSE

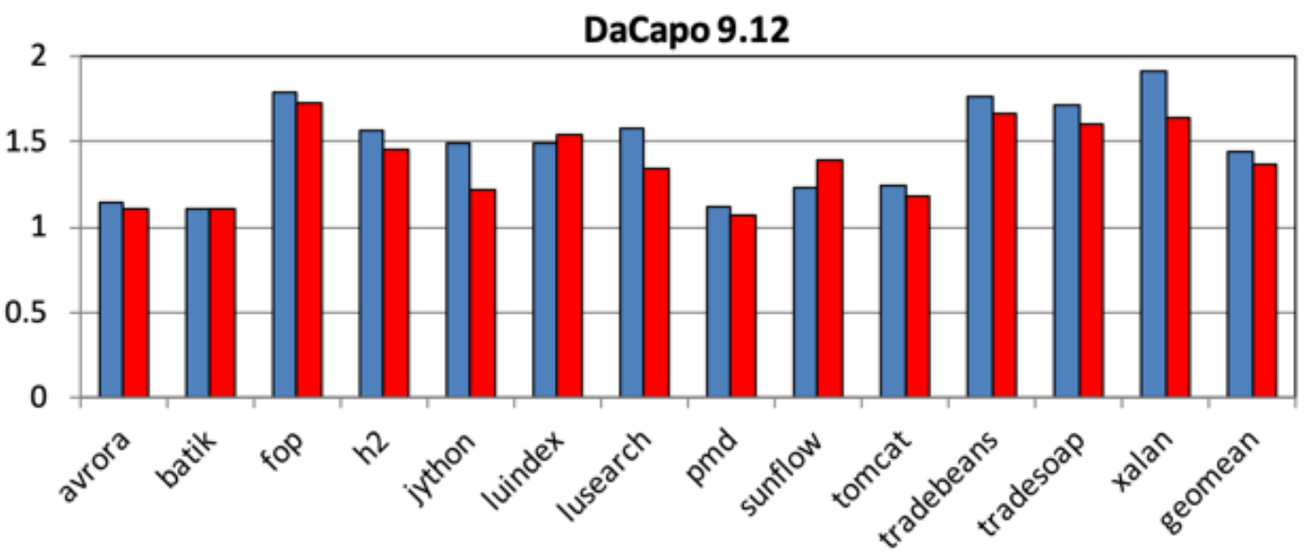
Graal Benchmark Results

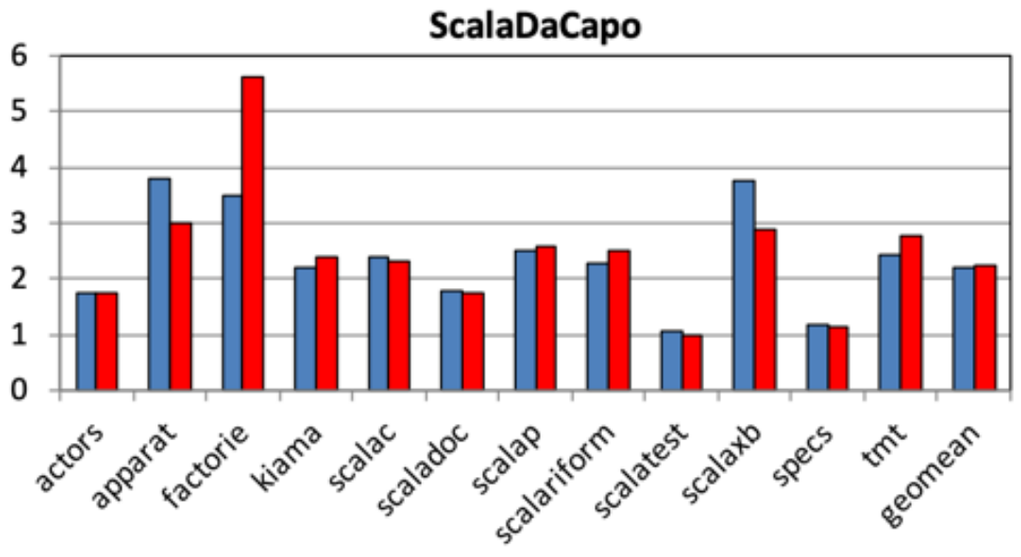




Higher is better, normalized to Client compiler.

Results are not SPEC compliant, but follow the rules for research use.





https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM Setup

GraalVM Home Docs **Downloads** Community ★ Star 9,175 TRY GRAALVM

The standard GraalVM bundle can run Java and JavaScript either via OpenJDK (Java 8u212), Node.js (v10.15.2) or standalone. **GraalVM 19.0.0** is available as Community Edition (CE) and Enterprise Edition (EE). The most notable changes in GraalVM can be found from the release notes.

GraalVM Updater tool included in the core distribution can add support for GraalVM Native Image and experimental support for the Ruby, R, or Python languages. It can also install third party languages and tools.

The Oracle Database Multilingual Engine with added JavaScript support via GraalVM is available here.

Community Edition (CE)

GraalVM CE is available for free for development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for GraalVM CE for Linux and Mac OS X on x86 64-bit systems.

DOWNLOAD FROM GITHUB

Enterprise Edition (EE)

GraalVM EE provides additional performance, security, and scalability relevant for running critical applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for GraalVM EE for Linux or Mac OS X on x86 64-bit systems.

DOWNLOAD FROM OTN

https://www.graalvm.org/downloads

GraalVM Setup

graalvm-ce-darwin-amd64-19.0.0.tar.gz	317 MB
graalvm-ce-linux-amd64-19.0.0.tar.gz	322 MB
graalvm-ce-windows-amd64-19.0.0.zip	179 MB
native-image-installable-svm-darwin-amd64-19.0.0.jar	10.2 MB
native-image-installable-svm-linux-amd64-19.0.0.jar	9.91 MB
Source code (zip)	
Source code (tar.gz)	

https://github.com/oracle/graal/releases

GraalVM Setup

- GraalVM CE (Linux, Mac, & Windows preview) http://www.graalvm.org/downloads/
- GraalVM EE https://www.oracle.com/technetwork/oracle-labs/program-languages/downloads/index.html
- Follow installation instructions
- Add GRAALVM_HOME to your PATH

```
Davids-MacBook-Pro-4:seeking-graal-cjug19 ddlucas$ java -version
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-20190420112649.buildslave.jdk8u-src-tar--b03)
OpenJDK GraalVM CE 19.0.0 (build 25.212-b03-jvmci-19-b01, mixed mode)
```

http://www.graalvm.org/docs/getting-started/

GraalVM Layout

Typical JDK Structure

- https://docs.gradle.org/current/userguide/

GraalVM Layout

New Members

```
R                jarsigner        jhat             jstatd           policytool       serialver
Rscript          java             jinfo            jvisualvm        polyglot         servertool
appletviewer     javac            jjs              keytool          rake             testrb
extcheck         javadoc          jmap             lli              rdoc             tnameserv
gem              javah            jps              native-image     ri               truffleruby
graalpython      javap            jrunscript       native2ascii     rmic             unpack200
gu               jcmd             js               node             rmid             wsgen
idlj             jconsole         jsadebugd        npm              rmiregistry      wsimport
irb              jdb              jstack           orbd             ruby             xjc
jar              jdeps            jstat            pack200          schemagen
```

- https://docs.gradle.org/current/userguide/

GraalVM Tooling

- Graal Updater (gu): installs languages
- Scripts can use Chrome Inspector (—inspect)
- Native Image: convert jar to exec or shared library
  - creates light JVM: Substrate VM
  - native-image -H:ReflectionConfigurationFiles=graal_config.json x.jar # provides explicit class loading

https://docs.gradle.org/current/release-notes.html

GraalVM DEMO Verify

- MAC: make sure you load xcode command line
  xcode-select --version # 2354 or higher
  xcode-select --install
  clang --version # Apple LLVM version 10.0.1 (clang-1001.0.46.4)
- verify GraalVM:
  export PATH= ...
  gu available
  gu install native-image python R ruby
  # follow instructions, contains experimental items

GraalVM DEMO Verify

- cd examples/hello-poly
  javac HelloPolyglotWorld.java
  java HelloPolyglotWorld

GraalVM DEMO

- examples/build.sh will pull down some git repos and place in tmp folder
- it also creates a docker image for Linux demo
- examples can run from Mac or Linux
- this is a subset of GraalVM samples

GraalVM DEMO FizzBuzz

- Shell into Linux Docker Image
- docker run -p3000:3000 -it graalvm-demo:0.01 bash
- source environment
  . /dockerjava/setup-gvm.env
- add experimental languages
  gu install R ruby python native-image
  cd /dockerjava/graalvm-ten-things
- which java && which R && which ruby && which graalpython && which native-image
- js fizzbuzz.js
- graalpython fizzbuzz.py
- Rscript fizzbuzz.r
- ruby fizzbuzz.rb

GraalVM DEMO Polyglot Node.js

- node --jvm polyglot.js
- which npm
- npm init
- npm install
- node --jvm polyglot.js
- Linux: curl http://localhost:3000/ # hard to see
- Mac: open http://localhost:3000/

GraalVM DEMO LLVM

- ls -l gzip.*
- clang -c -emit-llvm gzip.c
- file gzip.bc
- echo "HELLO WORLD" | lli gzip.bc -f -c - | zcat -

GraalVM DEMO Streams

- cd ../streams-example
- /opt/graalvm-ce-19.0.0/bin/java -jar target/benchmarks.jar mapReduce -f1 -wi 4 -i4
- /usr/lib/jvm/java-11-openjdk-amd64/bin/java -jar target/benchmarks.jar mapReduce -f1 -wi 4 -i4

GraalVM DEMO JDK 11 JS

- cd ../graal-js-jdk11-maven-demo/test
- mvn clean install
- mvn test

Adding some Ketchup

- So what about a complicated server ???
- Native can handle runtimes that have explicit resolution of types and dependencies
- Spring-FU project is working to create a functional API that avoids annotations, reflection, and runtime
- Working with Graal team to improve both
- https://spring.io/blog/2018/10/02/the-evolution-of-spring-fu

Spring FU (kofu)

```
import org.springframework.fu.kofu.web.server
import org.springframework.fu.kofu.webApplication
import org.springframework.web.reactive.function.server.ServerRequest
import org.springframework.web.reactive.function.server.ServerResponse.ok

val app = webApplication {
    beans {
        bean<SampleService>()
        bean<SampleHandler>()
    }
    server {
        port = if (profiles.contains("test")) 8181 else 8080
        router {
            val handler = ref<SampleHandler>()
            GET("/", handler::hello)
            GET("/api", handler::json)
        }
        codecs {
            string()
            jackson()
        }
    }
}
data class Sample(val message: String)
class SampleService {
    fun generateMessage() = "Hello world!"
}
class SampleHandler(private val sampleService: SampleService) {
    fun hello(request: ServerRequest)= ok().syncBody(sampleService.generateMessage())
    fun json(request: ServerRequest) = ok().syncBody(Sample(sampleService.generateMessage()))
}
fun main() {
    app.run()
}
```

GraalVM DEMO kofu

- cd tmp/spring-fu/samples/kofu-reactive-minimal
- ./gradlew build

- java -jar build/libs/kofu-reactive-minimal.jar

- native-image -jar build/libs/kofu-reactive-minimal.jar

- du -sm build/libs/*.jar kofu-reactive-minimal

GraalVM Notes
Native: Wait a minute !

WHAT	STATUS
Dynamic Class Loading / Unloading	Not supported
Reflection	Supported (Requires Configuration)
Dynamic Proxy	Supported (Requires Configuration)
Java Native Interface (JNI)	Mostly supported
Unsafe Memory Access	Mostly supported
Class Initializers	Supported
InvokeDynamic Bytecode and Method Handles	Not supported
Lambda Expressions	Supported
Synchronized, wait, and notify	Supported
Finalizers	Not supported
References	Mostly supported
Threads	Supported
Identity Hash Code	Supported
Security Manager	Not supported
JVMTI, JMX, other native VM interfaces	Not supported
JCA Security Services	Supported

GraalVM Notes

- Static / declarative dependencies works
- Brings a new AOT / JIT compiler to the table
- Static initialization and optimization saves in startup time
- Memory requirements are reduced
- Potential savings over multiple containers

GraalVM Notes

- Frameworks need to change how they do some things to work with Graal engine
- Spring FU is pushing to improve Graal
- Others supporting GraalVM include:
  https://micronaut.io
  https://quarkus.io
  https://helidon.io
- Graal Truffle allows for new scripting languages to take advantage of JVM and Native

GraalVM Summary

- Technology to keep an eye on
- Depends on goals
  - If you are in to polyglot…
  - Alternative to Node.js
  - Multi-platform support
  - Ready for production, maybe for Twitter
- Extensible
- Flexible
- Open Source
- I get to use Kotlin more
- Enterprise Edition provides even more

GraalVM Resources

- https://www.graalvm.org
- https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal
- https://www.slideshare.net/ThomasWuerthinger/2014-0424-graal-modularity
- https://www.slideshare.net/ThomasWuerthinger/graal-truffle-ethdec2013
- https://medium.com/graalvm/stream-api-performance-with-graalvm-be6cfe7fbb52
- https://medium.com/graalvm/graalvm-ten-things-12d9111f307d
- https://medium.com/graalvm/under-the-hood-of-graalvm-jit-optimizations-d6e931394797
- https://github.com/oracle/graal

Questions ?

https://github.com/lseinc/seeking-graal-cojug19.git

L
S
E

Thank You !

https://github.com/lseinc/seeking-graal-cojug19.git

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com
@DavidDLucas

L S E

LSE