Seeking the holy Graal

Presenter: David Lucas

L S E

Who am I ?

- Over 25+ years in software industry
- Working with Java since 1998
- Continuous Delivery
- Continuous Learner
- Focus mostly on server side solutions
- I am a Kotlin Enthusiast

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

My Agenda

- Show some of what GraalVM can do
- Show what GraalVM can not do (yet)
- Discuss where it might be useful
- My goal is to shrink resource usage for microservices (jar -> exec)
- Alternative to JIT ?

Goals

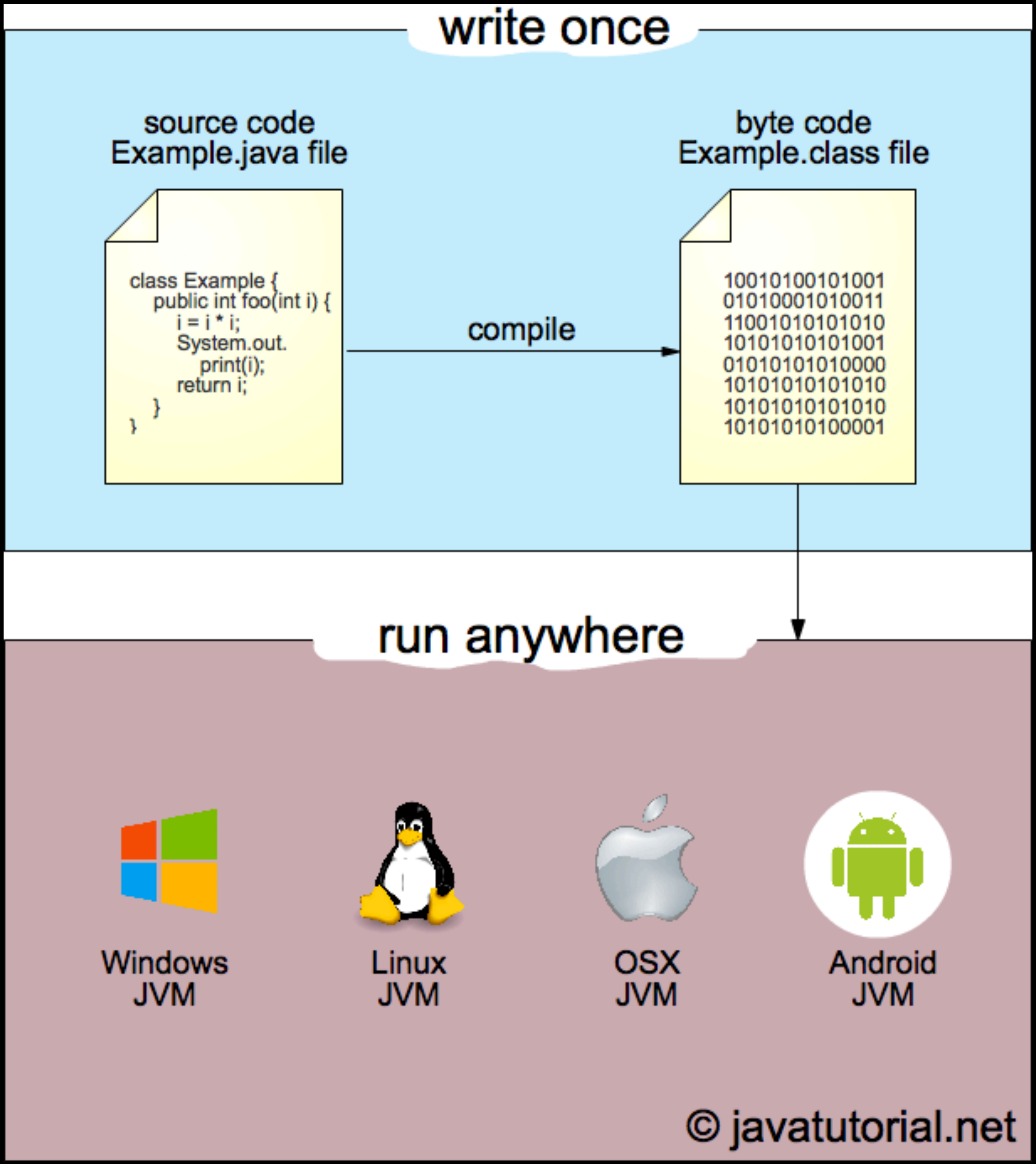
- Introduce GraalVM
- Demo some capabilities (js, R, rb, py)
- Demo running mixed environment
- Demo LLVM Interpreter
- Demo Native Images
- Summary

GraalVM Intro

- Graal is a highly optimized Ahead Of Time (AOT) compiler (versus 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/

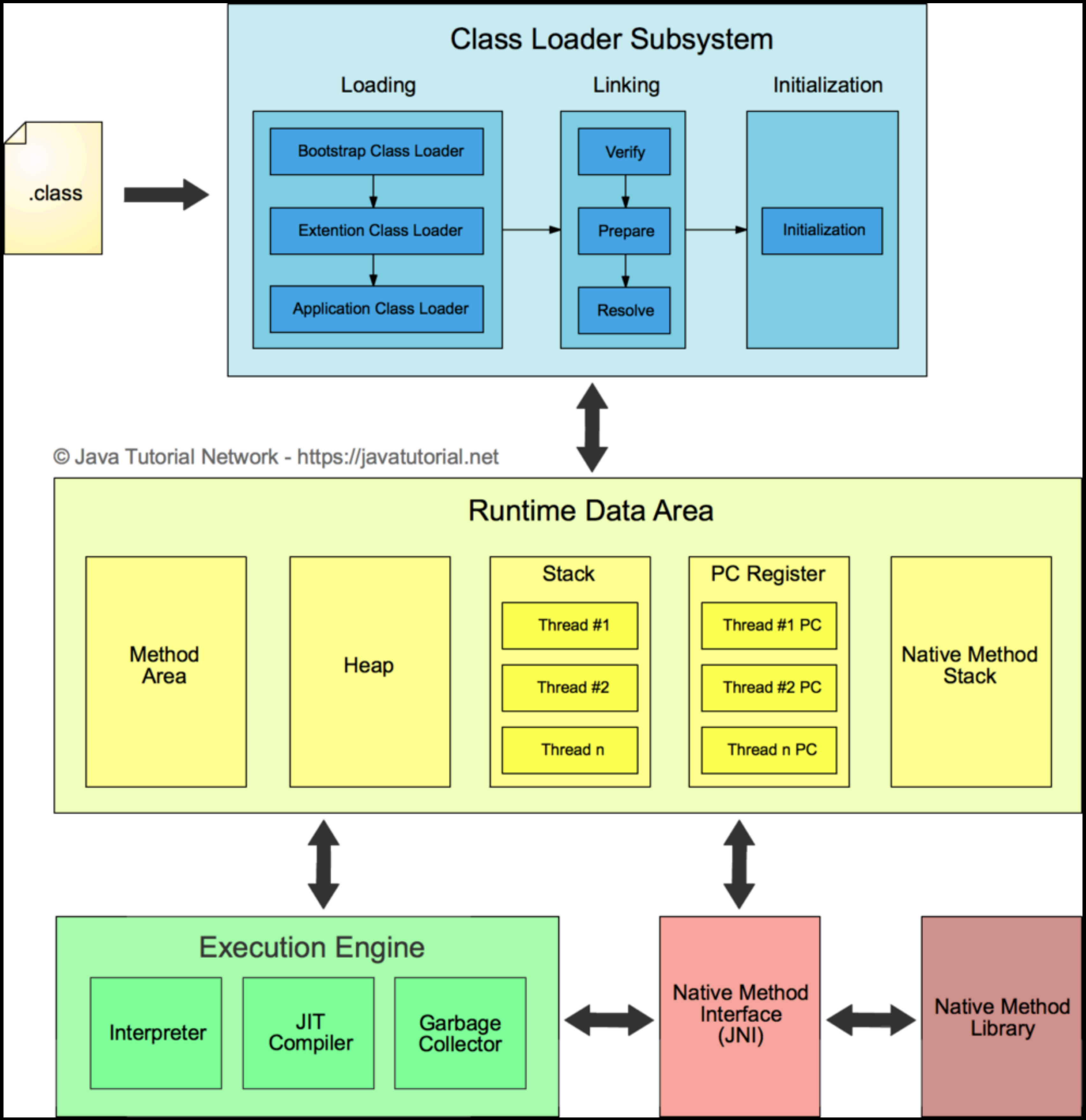
GraalVM Intro: Java Compilation

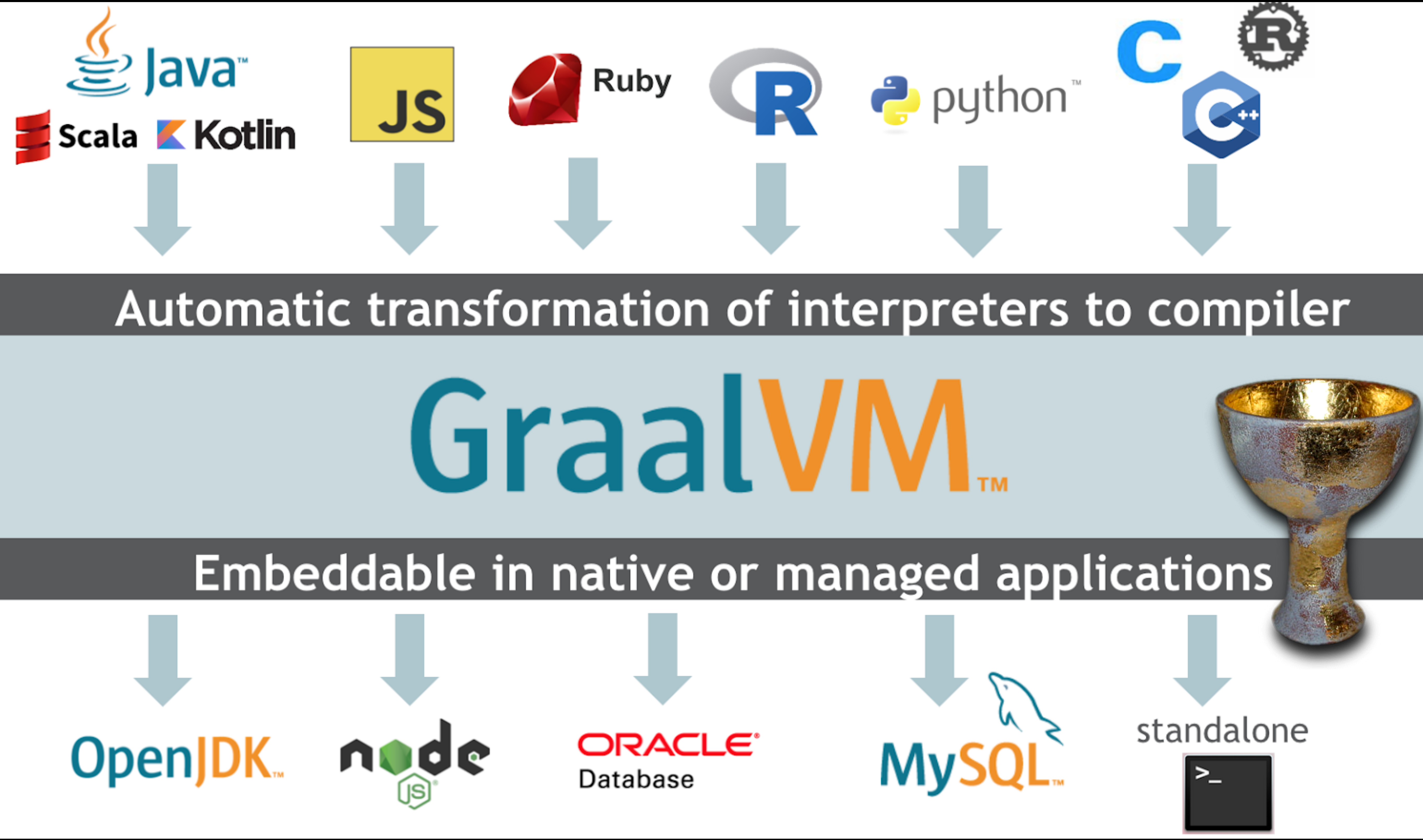


```
class Example {
    public int foo(int i) {
        i = i * i;
        System.out.
            print(i);
        return i;
    }
}
```

```
10010100101001
01010001010011
11001010101010
10101010101001
01010101010000
10101010101010
10101010101010
10101010100001
```

GraalVM Intro: What is the JVM ?




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, LLVM
- True Polyglot Runtime (shared data and functions)
- Easier than JNI
- SubstrateVM for native runtime
- Truffle API for scripting and creating new languages

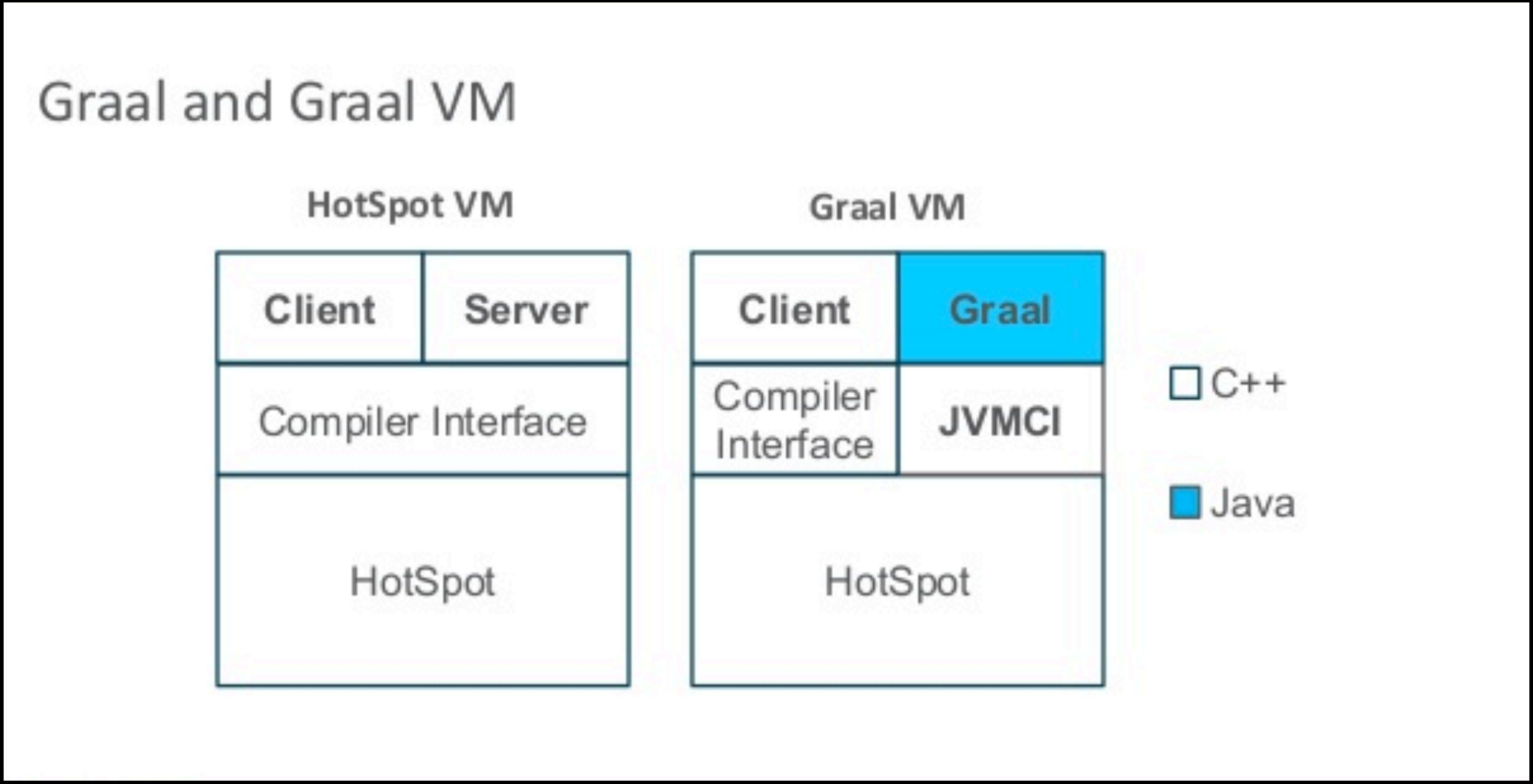
http://www.graalvm.org/docs/why-graal/

L S E

GraalVM Intro

- 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/



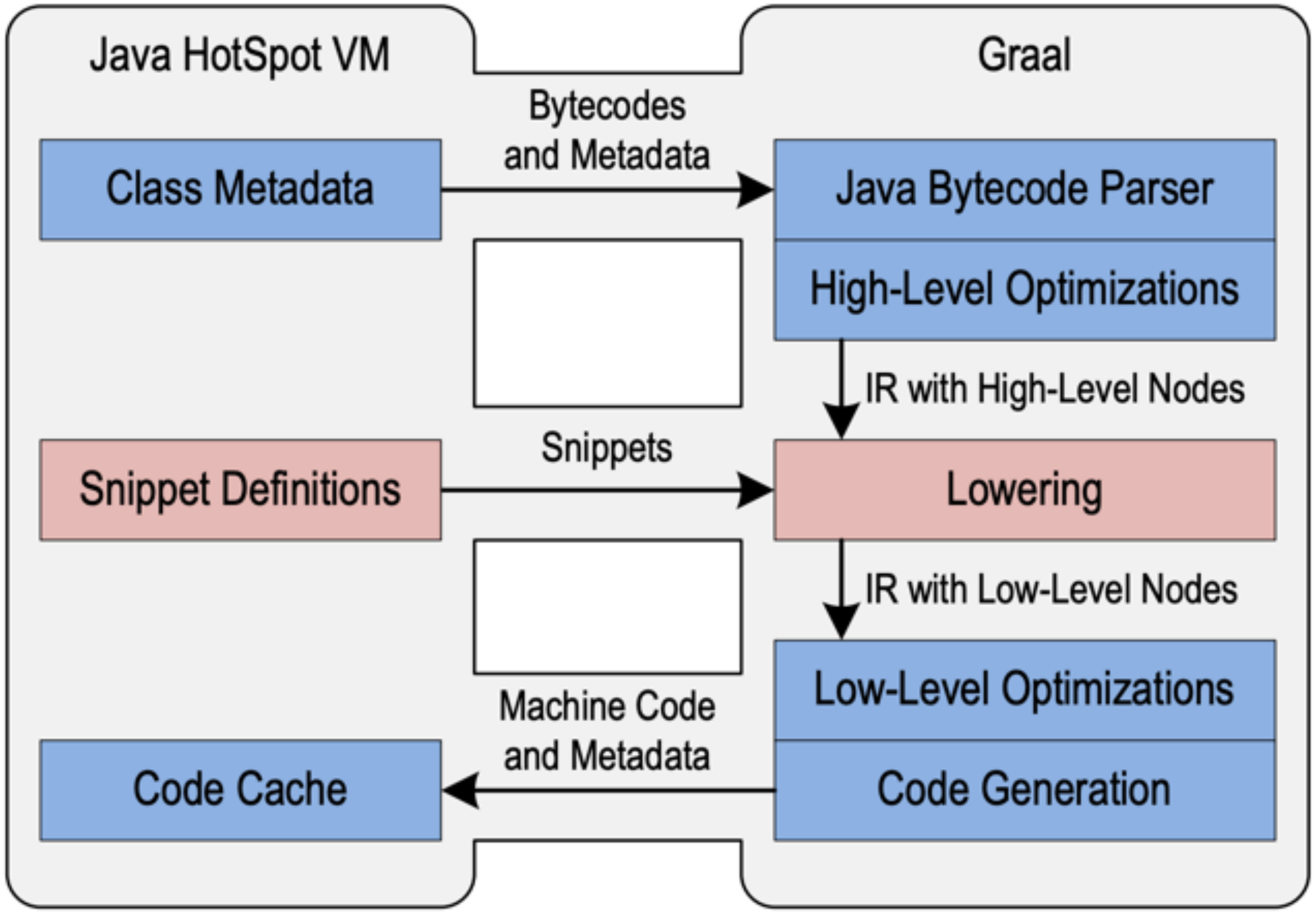
https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM Intro

- 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

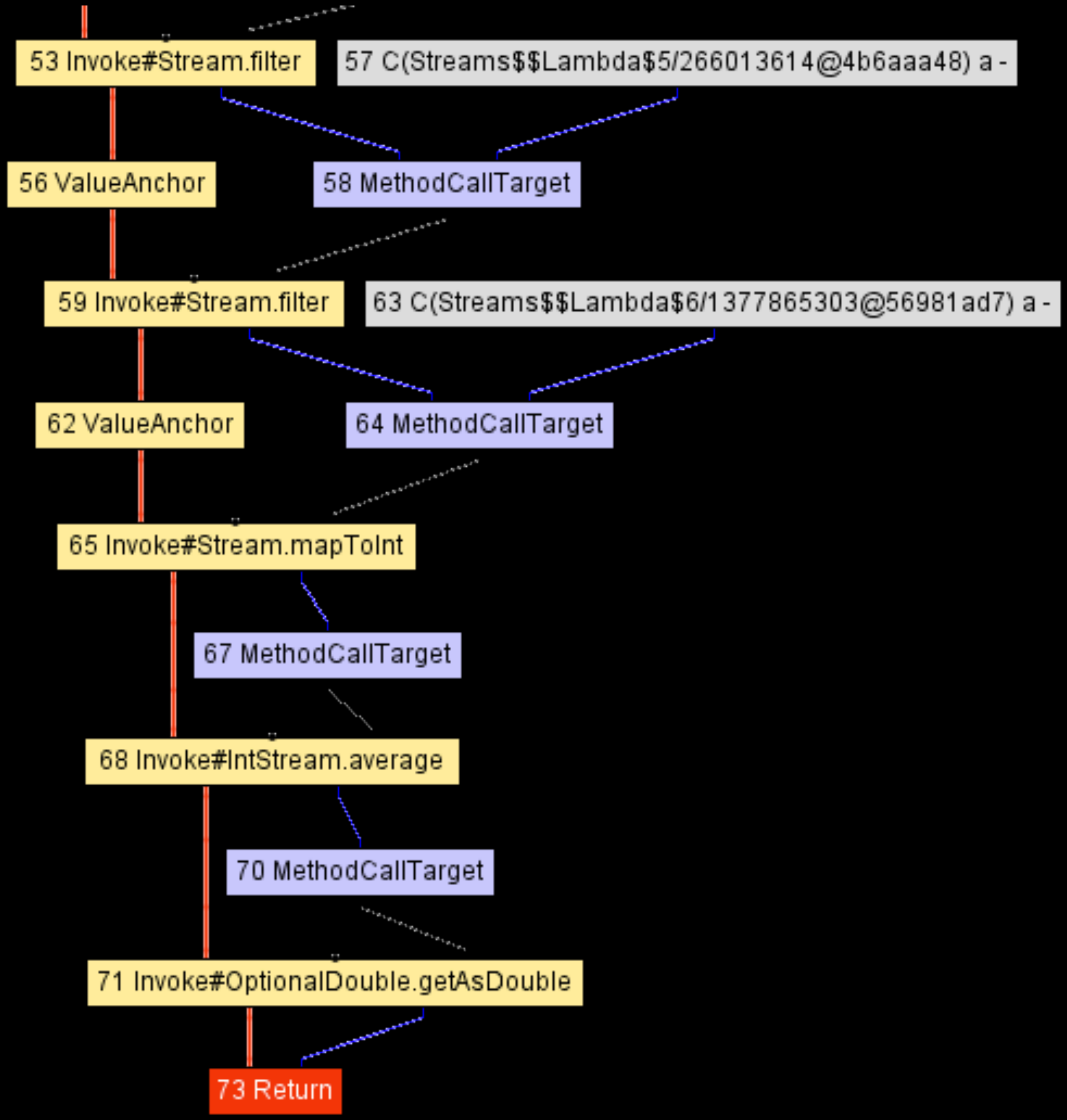
http://www.graalvm.org/docs/why-graal/

Compiler-VM Separation



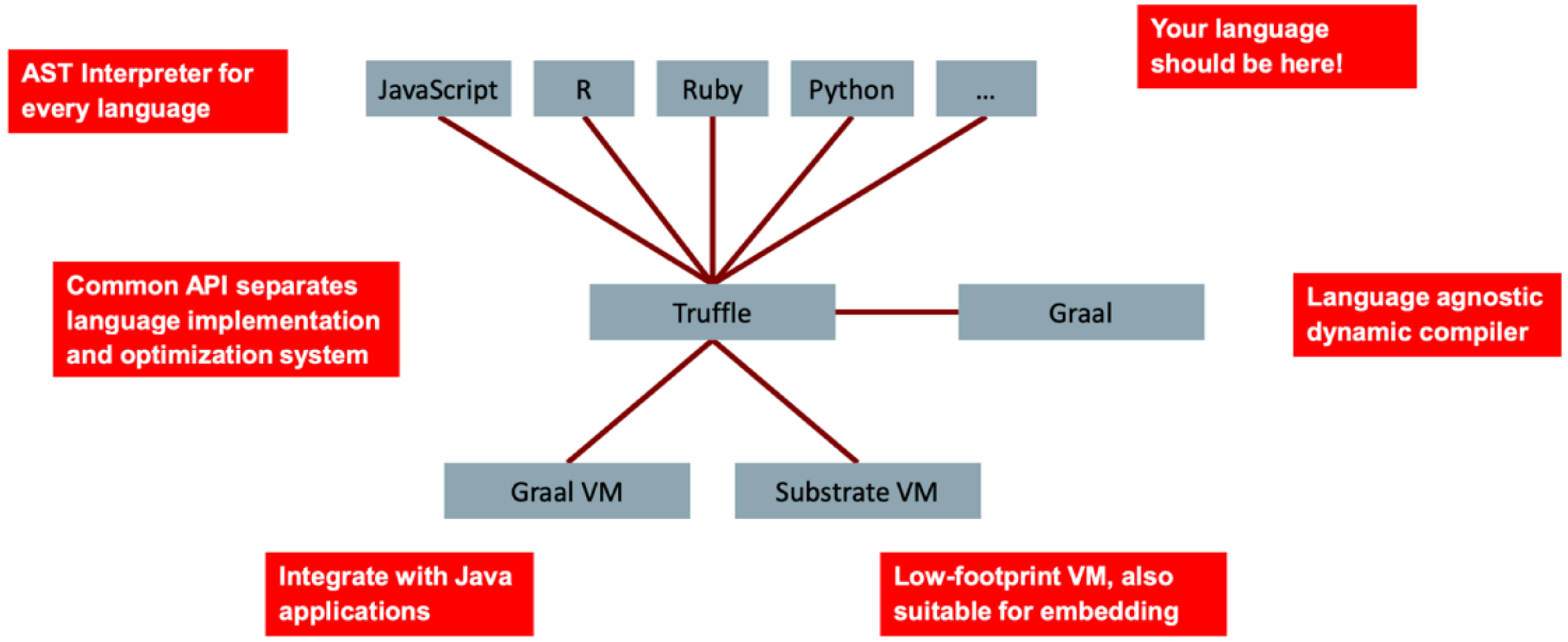
https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM
Intermediate Representation (IR)


53 Invoke#Stream.filter
57 C(Streams$$Lambda$5/266013614@4b6aaa48) a -
56 ValueAnchor
58 MethodCallTarget
59 Invoke#Stream.filter
63 C(Streams$$Lambda$6/1377865303@56981ad7) a -
62 ValueAnchor
64 MethodCallTarget
65 Invoke#Stream.mapToInt
67 MethodCallTarget
68 Invoke#IntStream.average
70 MethodCallTarget
71 Invoke#OptionalDouble.getAsDouble
73 Return

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

Truffle System Structure


AST Interpreter for every language
JavaScript
R
Ruby
Python
...
Your language should be here!
Common API separates language implementation and optimization system
Truffle
Graal
Language agnostic dynamic compiler
Graal VM
Substrate VM
Integrate with Java applications
Low-footprint VM, also suitable for embedding

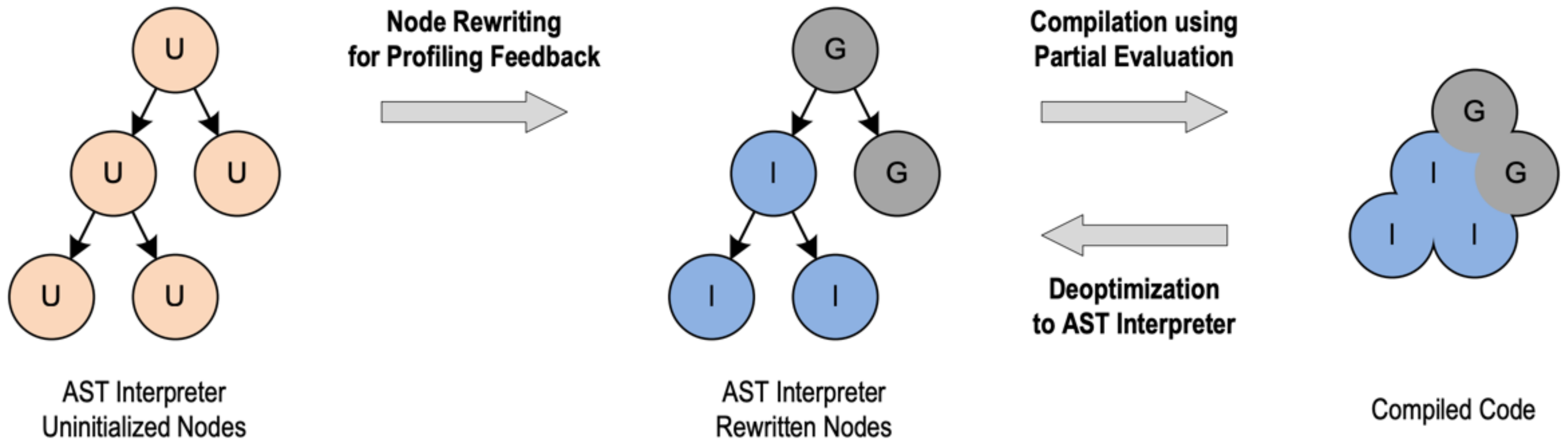
https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

GraalVM API

- 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/

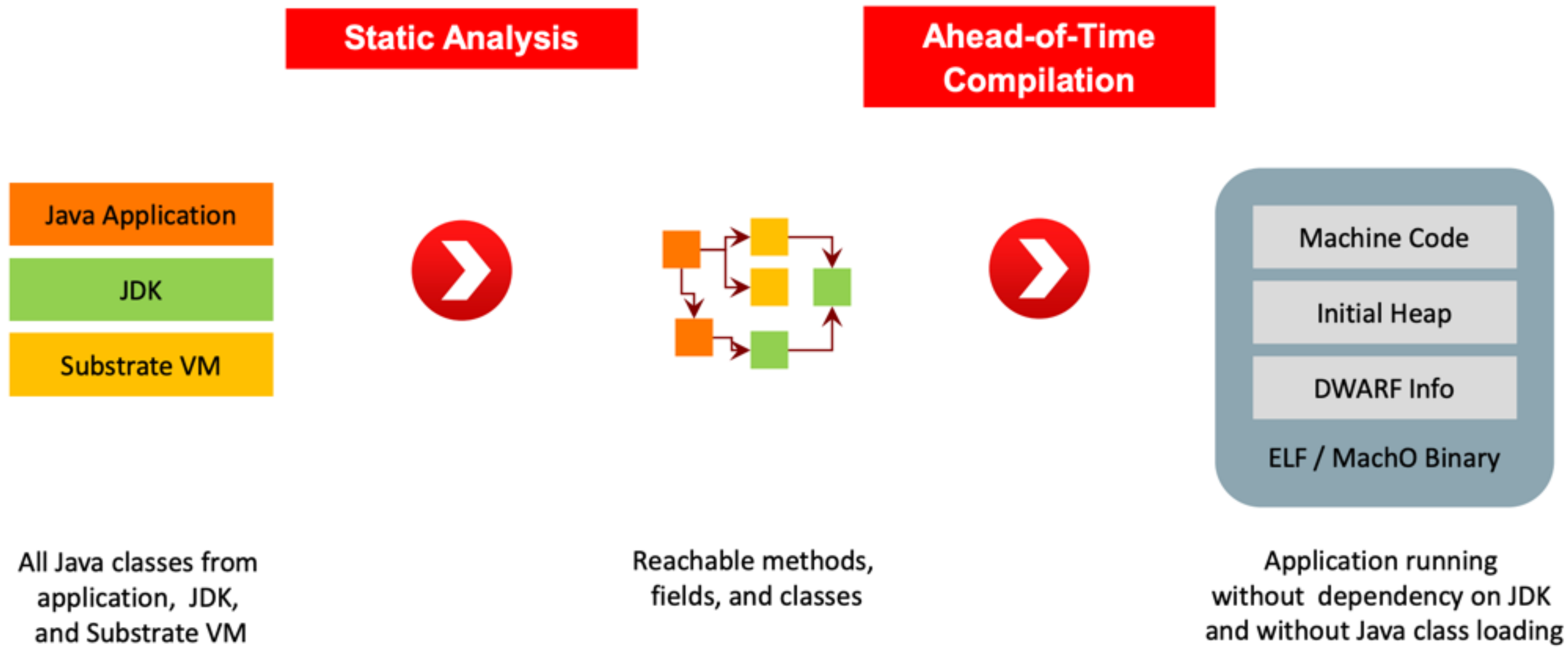
Truffle Approach



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

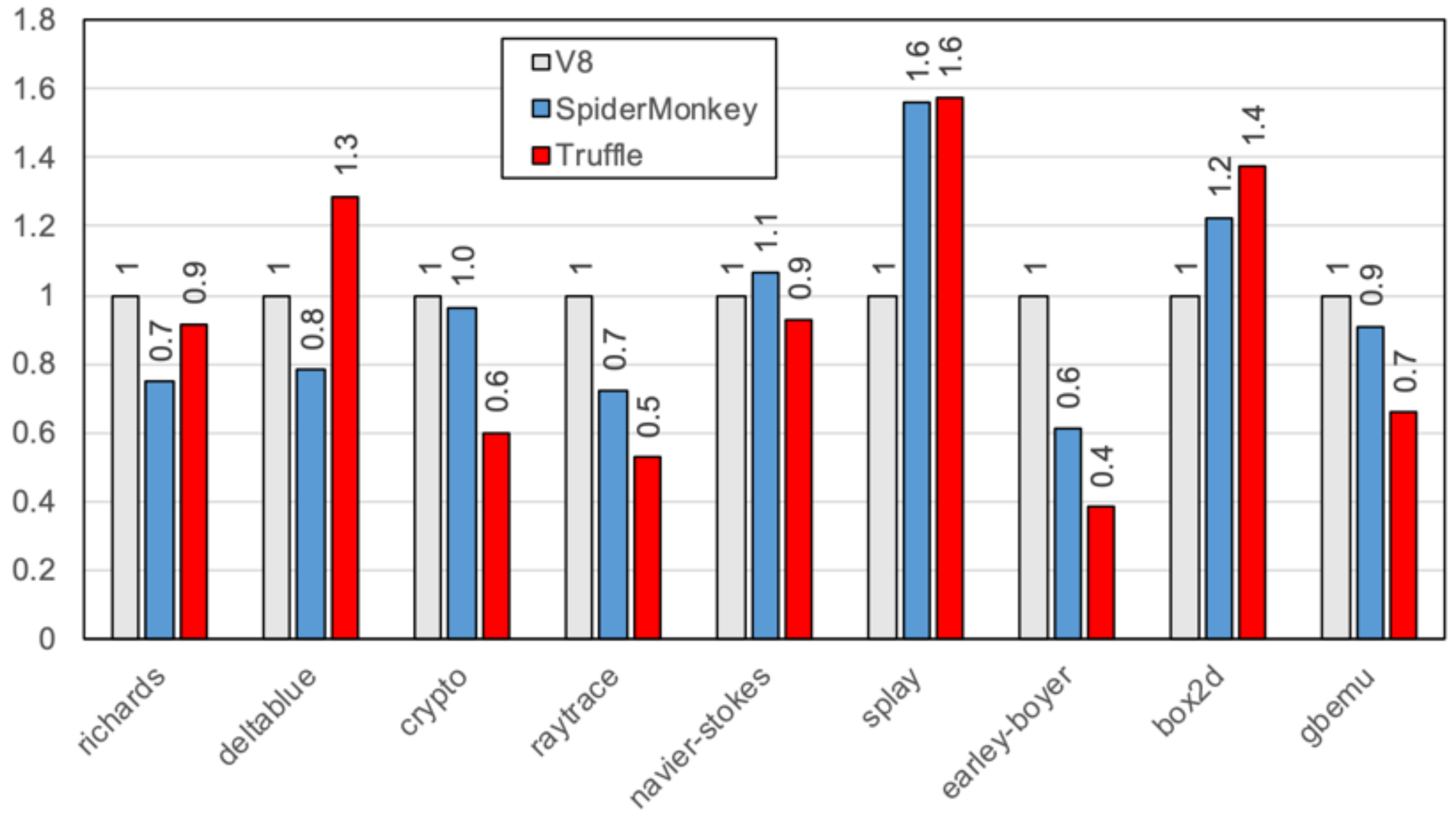
Substrate VM

Static Analysis and Ahead-of-Time Compilation using Graal


Static Analysis
Ahead-of-Time Compilation
Java Application
JDK
Substrate VM
Machine Code
Initial Heap
DWARF Info
ELF / MachO Binary
All Java classes from application, JDK, and Substrate VM
Reachable methods, fields, and classes
Application running without dependency on JDK and without Java class loading

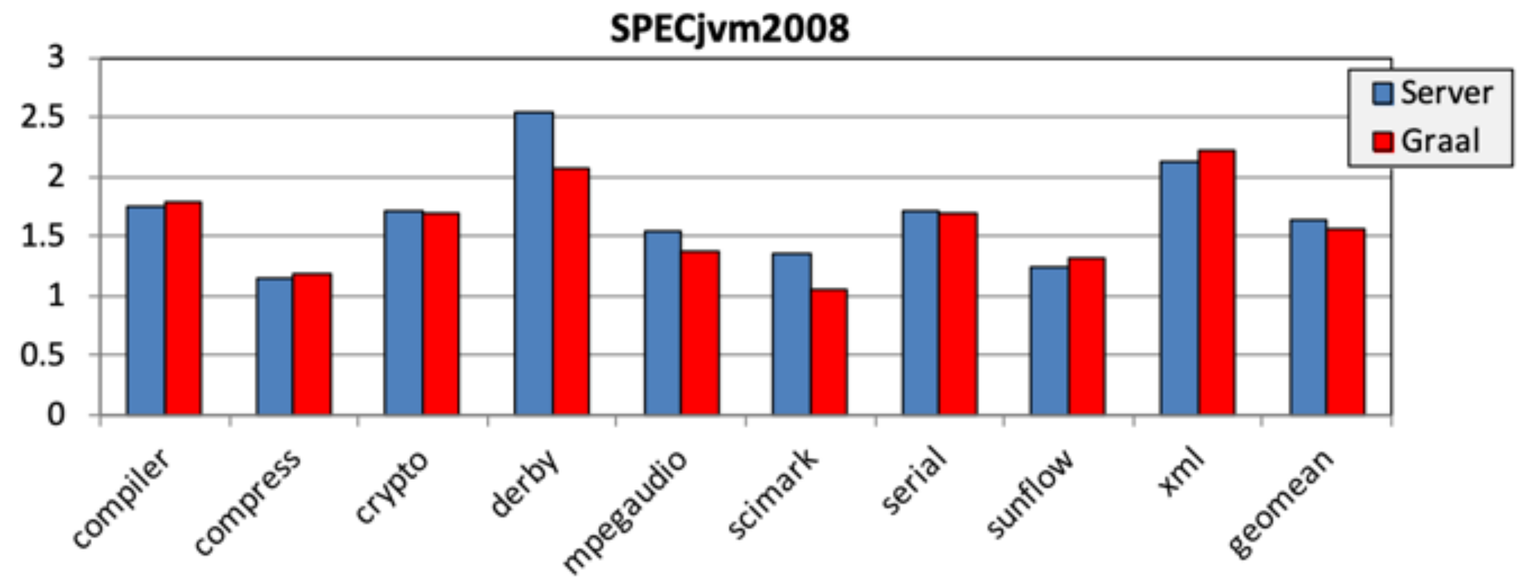
https://www.slideshare.net/ThomasWuerthinger/2015-cgo-graal

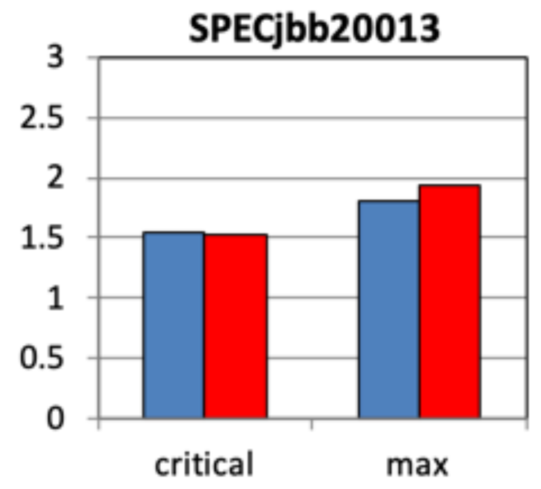
Performance: JavaScript



	V8	SpiderMonkey	Truffle
richards	1	0.7	0.9
deltablue	1	0.8	1.3
crypto	1	1.0	0.6
raytrace	1	0.7	0.5
navier-stokes	1	1.1	0.9
splay	1	1.6	1.6
earley-boyer	1	0.6	0.4
box2d	1	1.2	1.4
gbemu	1	0.9	0.7

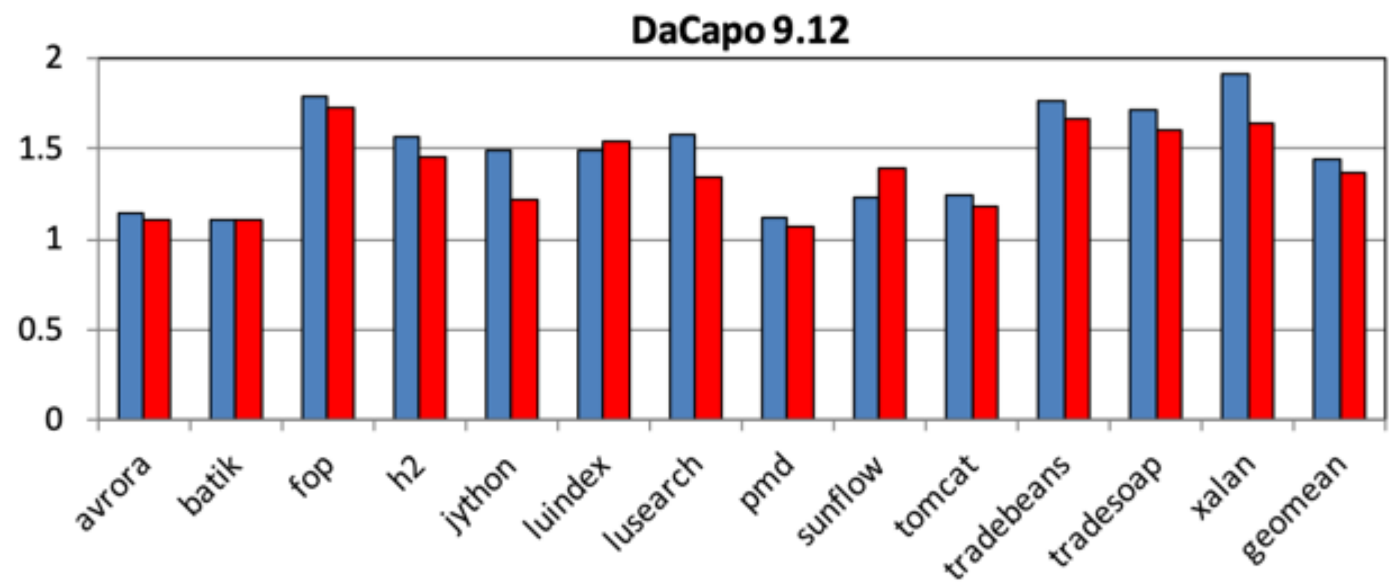
Graal Benchmark Results

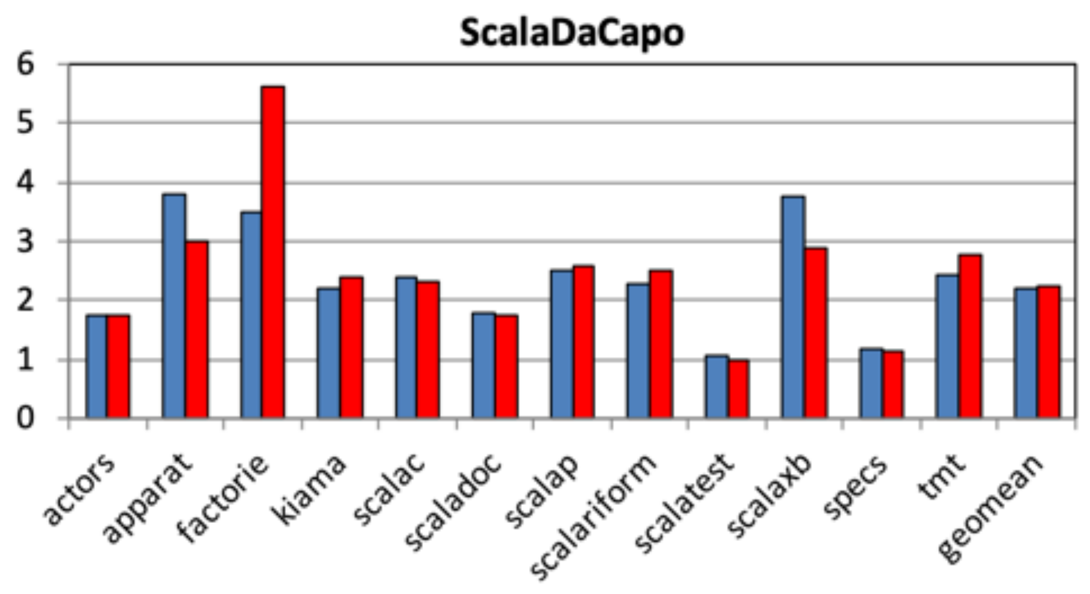




Higher is better, normalized to Client compiler.

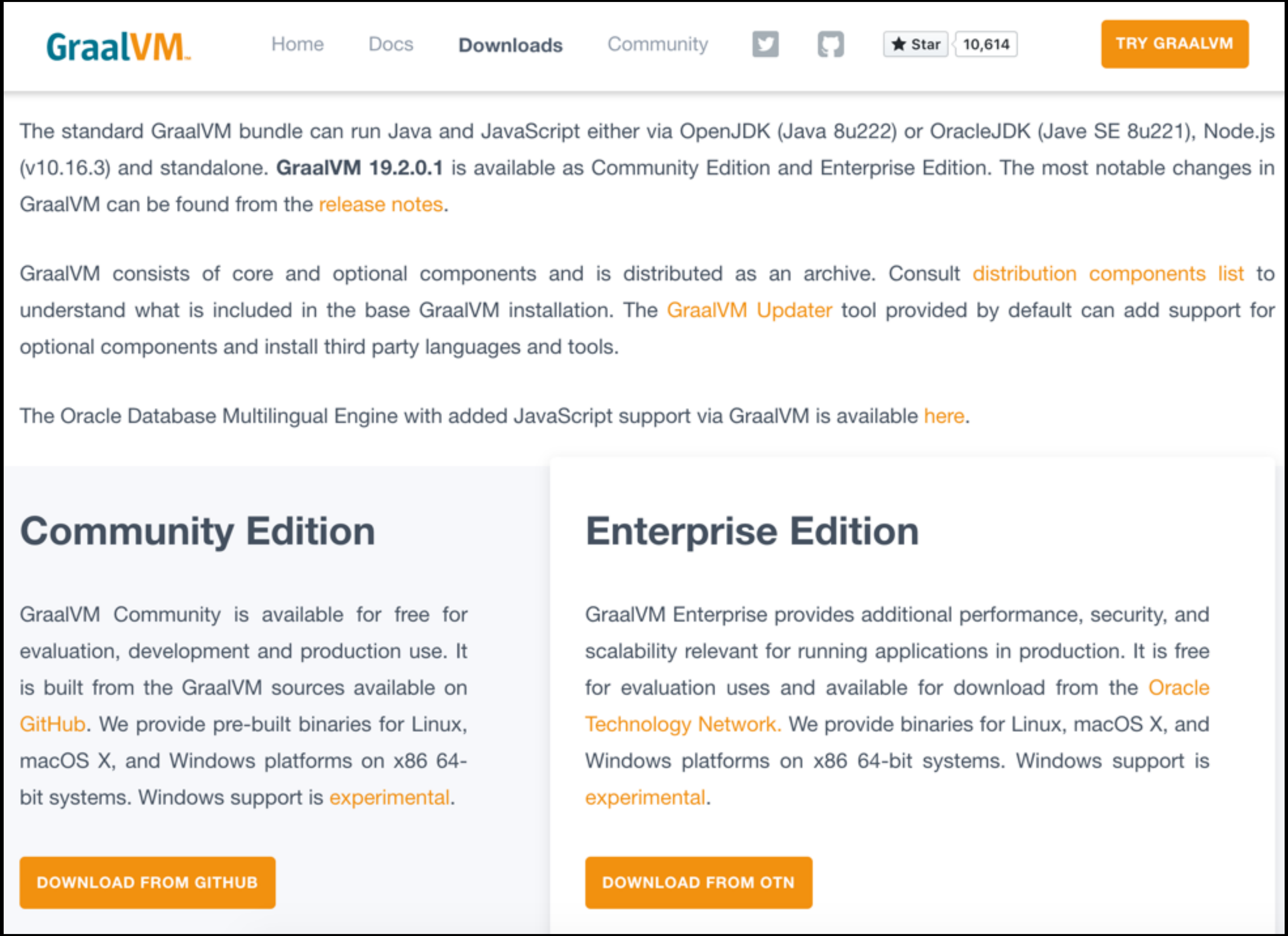
Results are not SPEC compliant, but follow the rules for research use.





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

GraalVM Setup



The standard GraalVM bundle can run Java and JavaScript either via OpenJDK (Java 8u222) or OracleJDK (Jave SE 8u221), Node.js (v10.16.3) and standalone. **GraalVM 19.2.0.1** is available as Community Edition and Enterprise Edition. The most notable changes in GraalVM can be found from the release notes.

GraalVM consists of core and optional components and is distributed as an archive. Consult distribution components list to understand what is included in the base GraalVM installation. The GraalVM Updater tool provided by default can add support for optional components and install third party languages and tools.

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

Community Edition

GraalVM Community is available for free for evaluation, development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for Linux, macOS X, and Windows platforms on x86 64-bit systems. Windows support is experimental.

DOWNLOAD FROM GITHUB

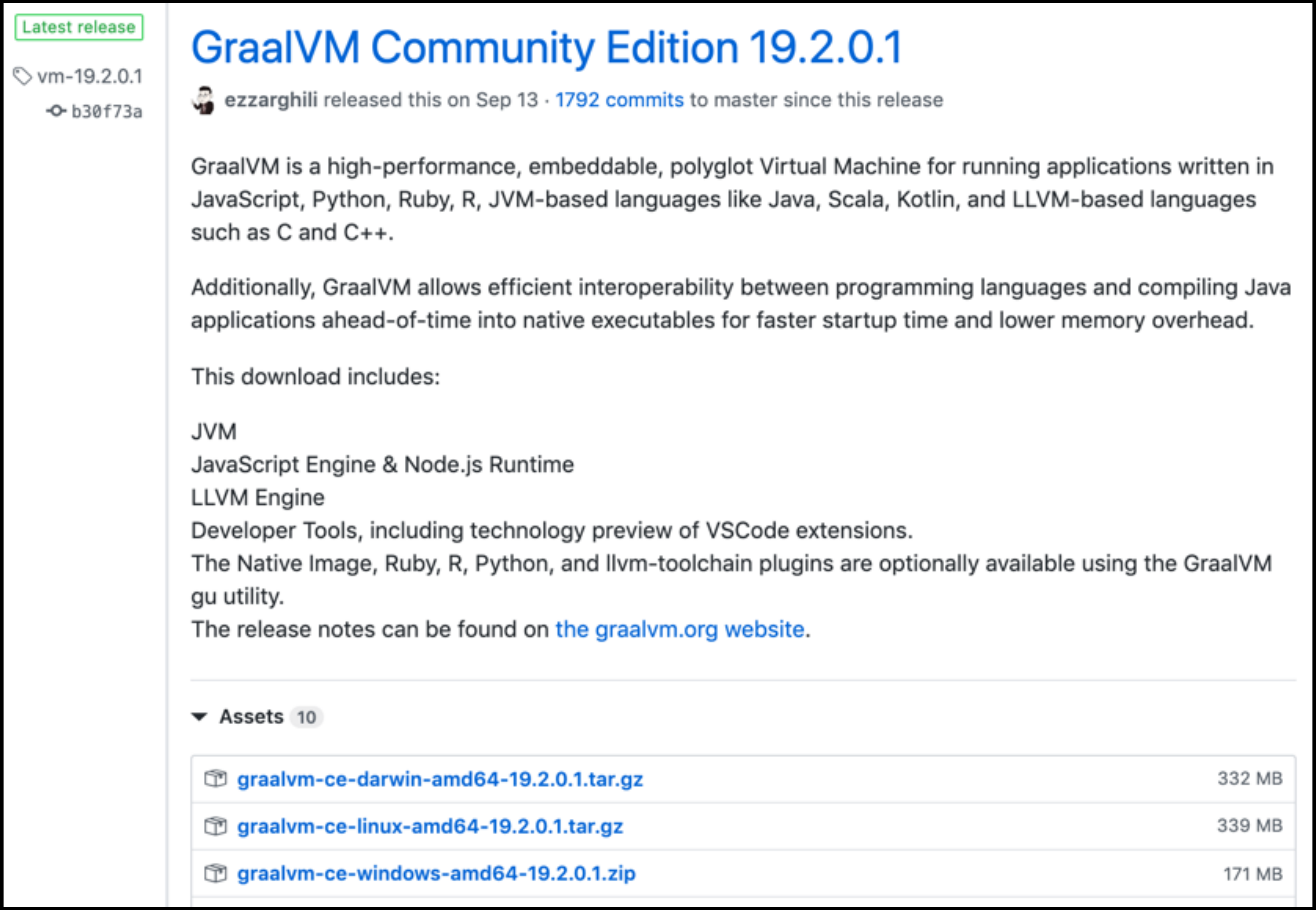
Enterprise Edition

GraalVM Enterprise provides additional performance, security, and scalability relevant for running applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for Linux, macOS X, and Windows platforms on x86 64-bit systems. Windows support is experimental.

DOWNLOAD FROM OTN

https://www.graalvm.org/downloads

GraalVM Setup



Latest release

vm-19.2.0.1

b30f73a

GraalVM Community Edition 19.2.0.1

ezzarghili released this on Sep 13 · 1792 commits to master since this release

GraalVM is a high-performance, embeddable, polyglot Virtual Machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Kotlin, and LLVM-based languages such as C and C++.

Additionally, GraalVM allows efficient interoperability between programming languages and compiling Java applications ahead-of-time into native executables for faster startup time and lower memory overhead.

This download includes:

JVM
JavaScript Engine & Node.js Runtime
LLVM Engine
Developer Tools, including technology preview of VSCode extensions.
The Native Image, Ruby, R, Python, and llvm-toolchain plugins are optionally available using the GraalVM gu utility.
The release notes can be found on the graalvm.org website.

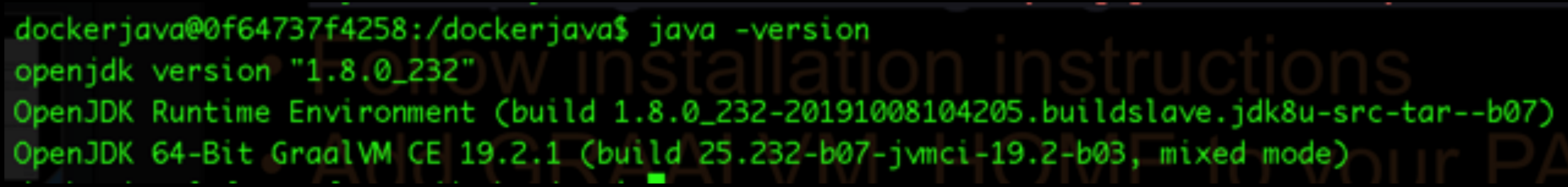
Assets 10

graalvm-ce-darwin-amd64-19.2.0.1.tar.gz	332 MB
graalvm-ce-linux-amd64-19.2.0.1.tar.gz	339 MB
graalvm-ce-windows-amd64-19.2.0.1.zip	171 MB

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

GraalVM Setup

- Binaries on Linux, Mac, & Windows
- GraalVM CE (OpenJDK based)
  http://www.graalvm.org/downloads/
- GraalVM EE (Oracle JDK based)
  https://www.oracle.com/technetwork/oracle-labs/program-languages/downloads/index.html
- Follow installation instructions
- Add $GRAALVM_HOME/bin to your PATH
- verify

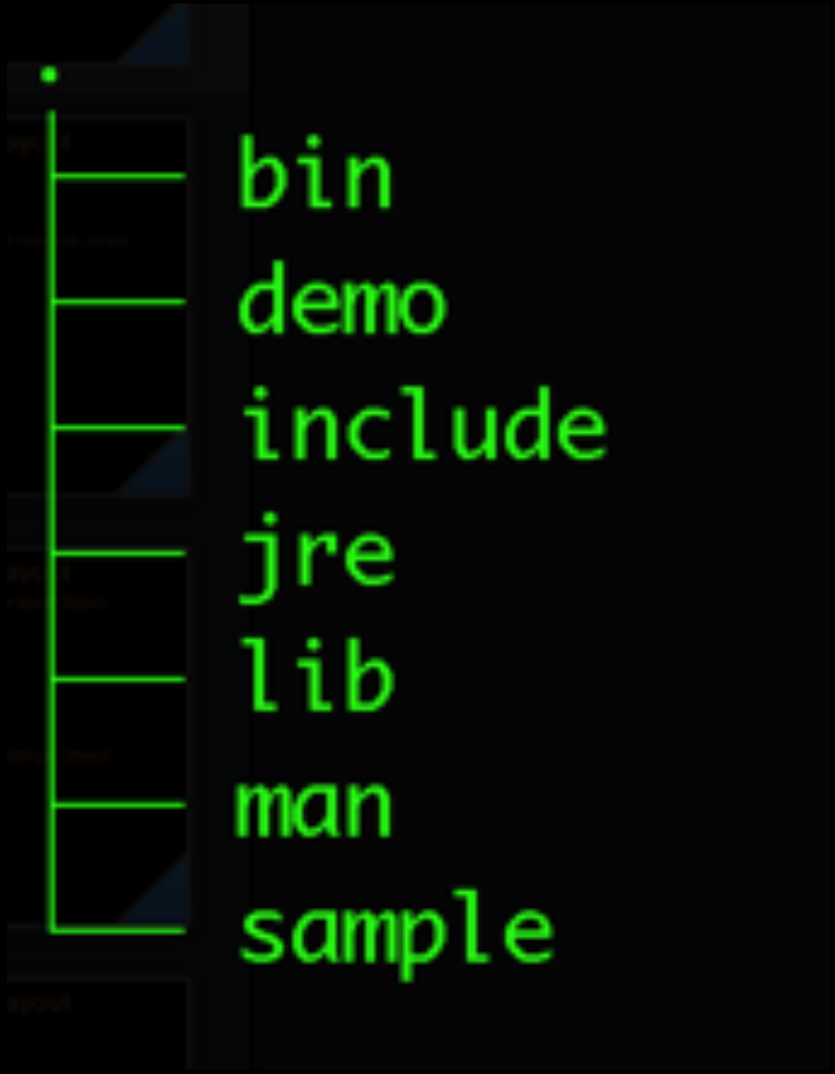


```
dockerjava@0f64737f4258:/dockerjava$ java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-20191008104205.buildslave.jdk8u-src-tar--b07)
OpenJDK 64-Bit GraalVM CE 19.2.1 (build 25.232-b07-jvmci-19.2-b03, mixed mode)
```

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

GraalVM Layout

Typical JDK Structure

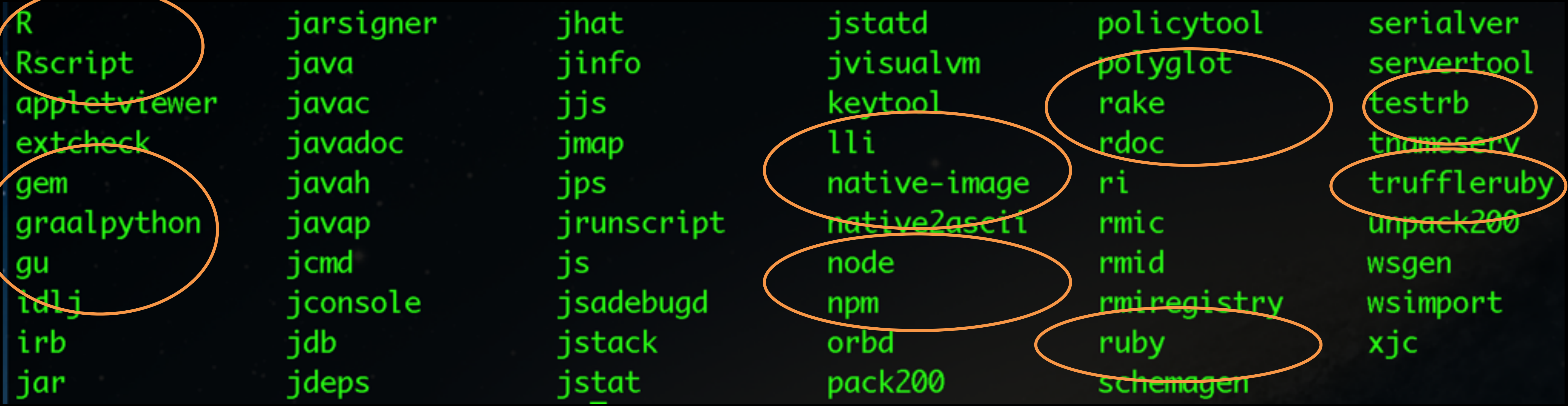


```
.
├── bin
├── demo
├── include
├── jre
├── lib
├── man
└── sample
```

- 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 lite JVM: Substrate VM)

```
native-image -H:ReflectionConfigurationFiles=graalvm-config.json
-cp x.jar  io.example.Application    # 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

```
$ cd examples
$ export GRAALVM_HOME=/opt/graalvm-ce-19.2.1/Contents/Home
$ export PATH=$GRAALVM_HOME:$PATH
$ which gu
/opt/graalvm-ce-19.2.0.1/Contents/Home/bin/gu
```

GraalVM DEMO Verify

```
$ cd examples/01-hello-poly

$ javac HelloPolyglotWorld.java

$ java HelloPolyglotWorld
```

GraalVM DEMO FizzBuzz

```
$ cd examples/02-fizzbuzz

$ js           --inspect fizzbuzz.js

$ python       --inspect fizzbuzz.py

$ Rscript      --inspect fizzbuzz.r

$ ruby         --inspect fizzbuzz.rb
```

GraalVM DEMO JS + R

```
cd examples/03-functionalGraphDemo

./build.sh

./run.sh   &

open http://localhost:8084
```

GraalVM DEMO Simple Speed

```
cd examples/04-speed

./build.sh

./run-without.sh

./run-graalvm.sh
```

GraalVM DEMO LLVM

```
cd examples/05-llvm

./build.sh

file helloNcurses.bc

./run.sh
```

GraalVM DEMO Reflect

```
# run on docker container, setup env
cd examples/06-reflect

./build.sh

./run.sh

#use -H:ReflectionConfigurationFiles=./graalvm_config.json
```

GraalVM DEMO Kotlin

```
# run on docker container, setup env
cd examples/07-spring-kofu

./gradlew bootRun

./native-compile.sh
```

Adding some Ketchup

- Native can handle runtimes as long as the dependencies are identified for compile time
- Spring-FU: working to create a functional API that avoids annotations and reflection, more explicit
- 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 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://ktor.io
  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
- Enterprise Edition provides even more with commercial support

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-odf19.git

Thank You !

https://github.com/lseinc/seeking-graal-odf19.git

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

LSE