The Eclipse OpenJ9 JVM
A deep dive!

Tobi Ajila

Tobi_Ajila@ca.ibm.com

Disclaimer

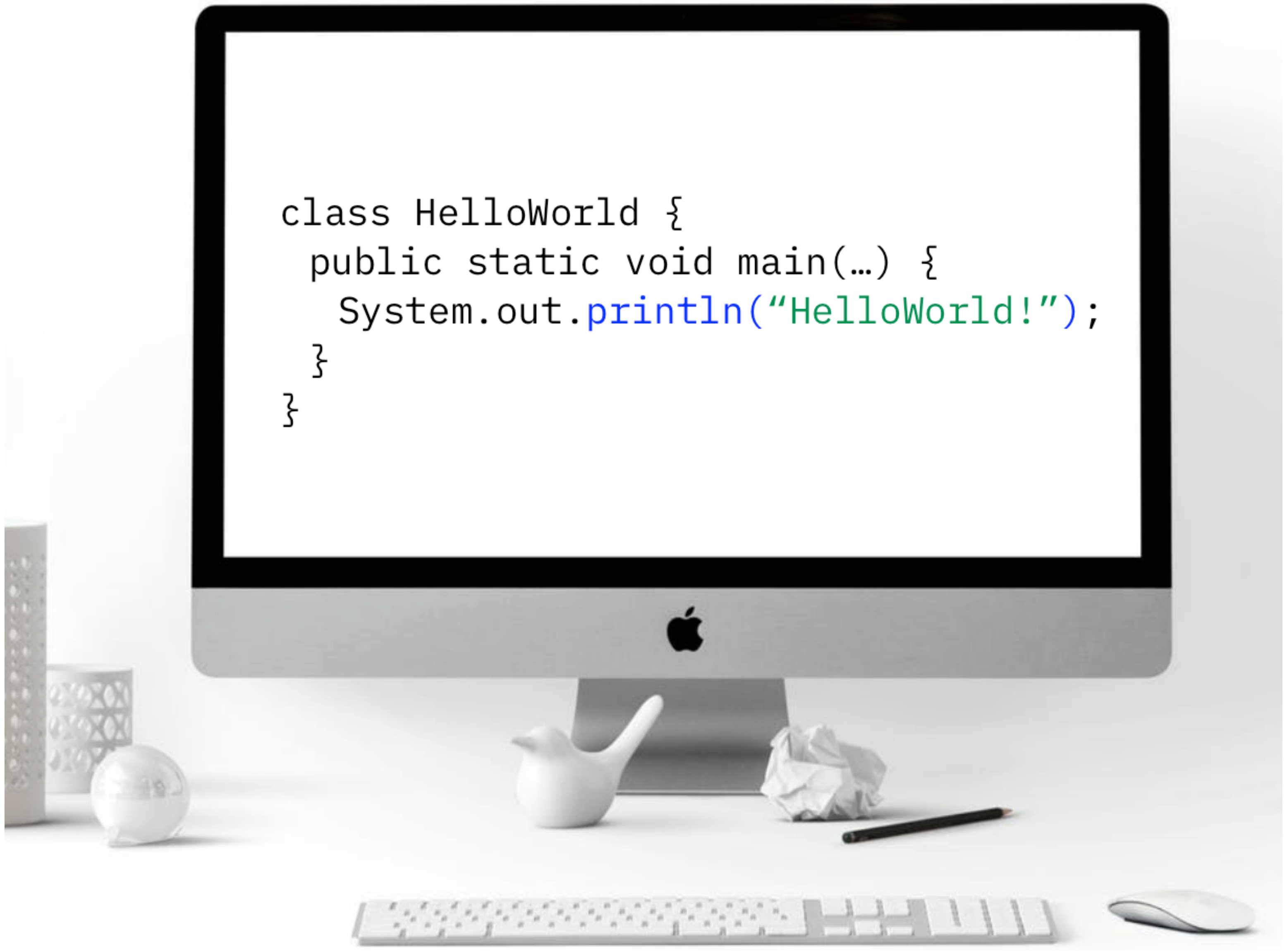
About the speaker

Tobi Ajila
Tobi_Ajila@ca.ibm.com

- Developer on the OpenJ9 VM team
- Worked on:
    - VM support for Lambda expressions
    - Java multitenancy incubator
    - Project Panama experts group

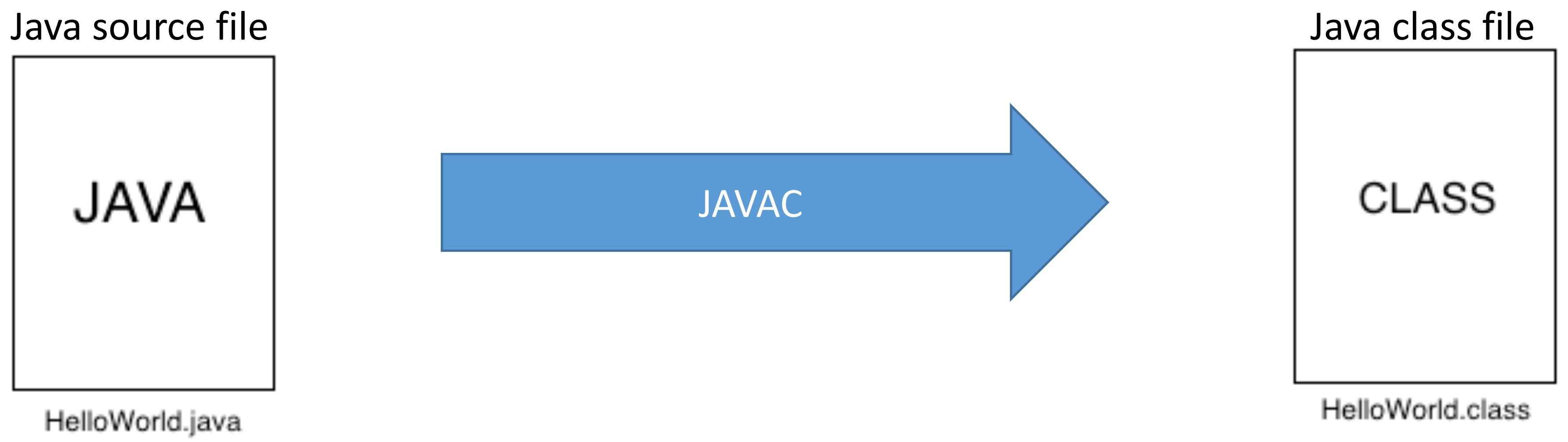
Agenda

- What is a JVM?
- What is OpenJ9?
- How does OpenJ9 work?
- What are some interesting features of OpenJ9?



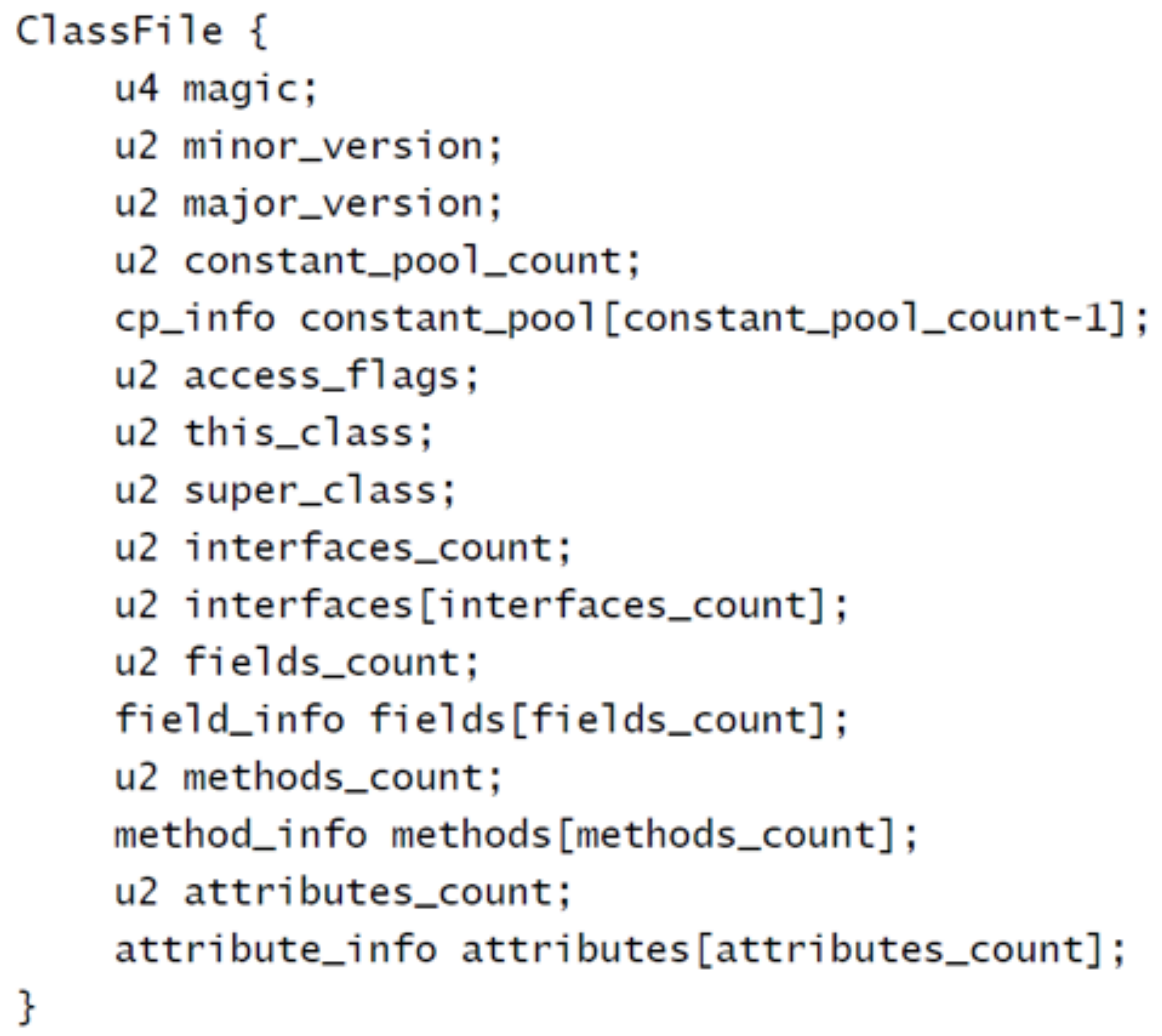
```
class HelloWorld {
 public static void main(…) {
  System.out.println("HelloWorld!");
 }
}
```

Compiling a Java application



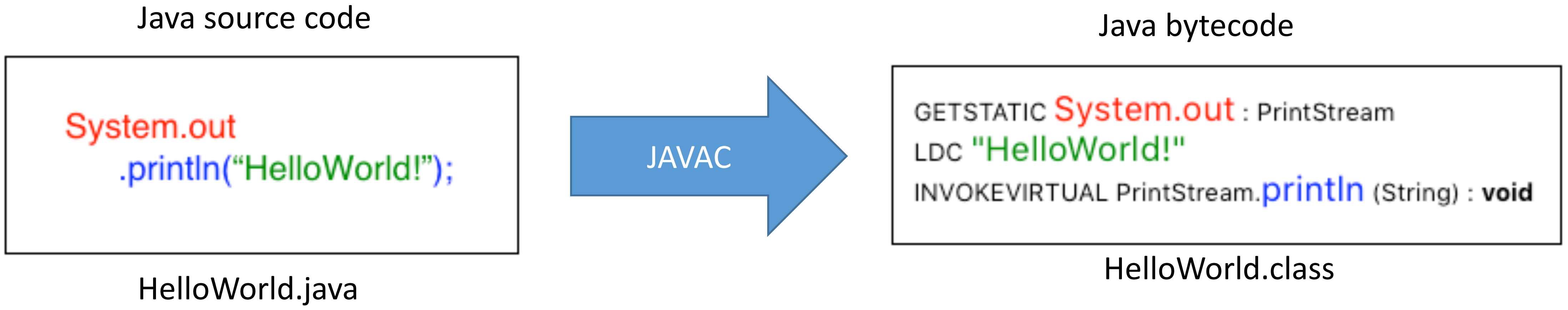
```
javac  HelloWorld.java
```

Java Class File



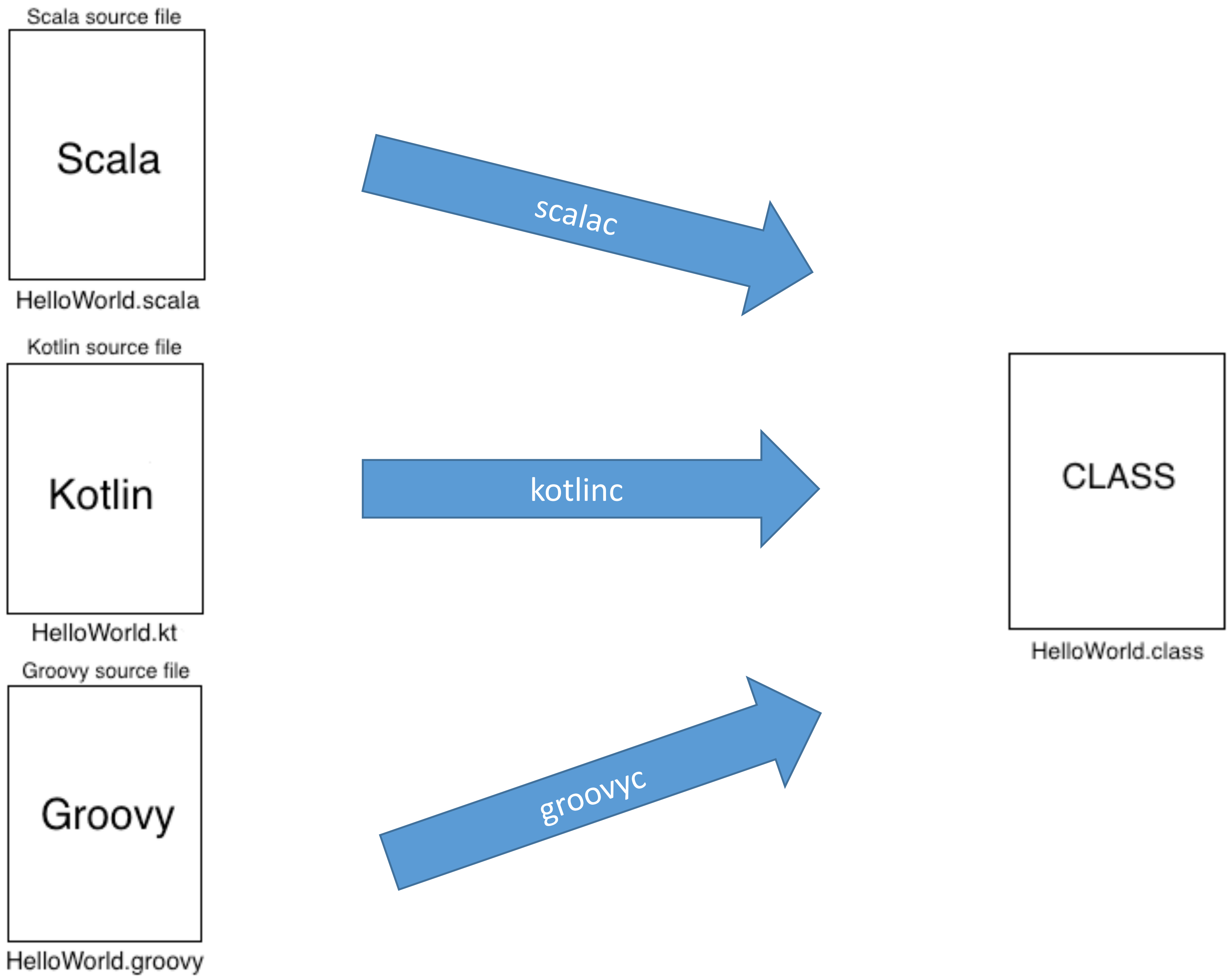
```
ClassFile {
    u4 magic;
    u2 minor_version;
    u2 major_version;
    u2 constant_pool_count;
    cp_info constant_pool[constant_pool_count-1];
    u2 access_flags;
    u2 this_class;
    u2 super_class;
    u2 interfaces_count;
    u2 interfaces[interfaces_count];
    u2 fields_count;
    field_info fields[fields_count];
    u2 methods_count;
    method_info methods[methods_count];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

Java bytecode

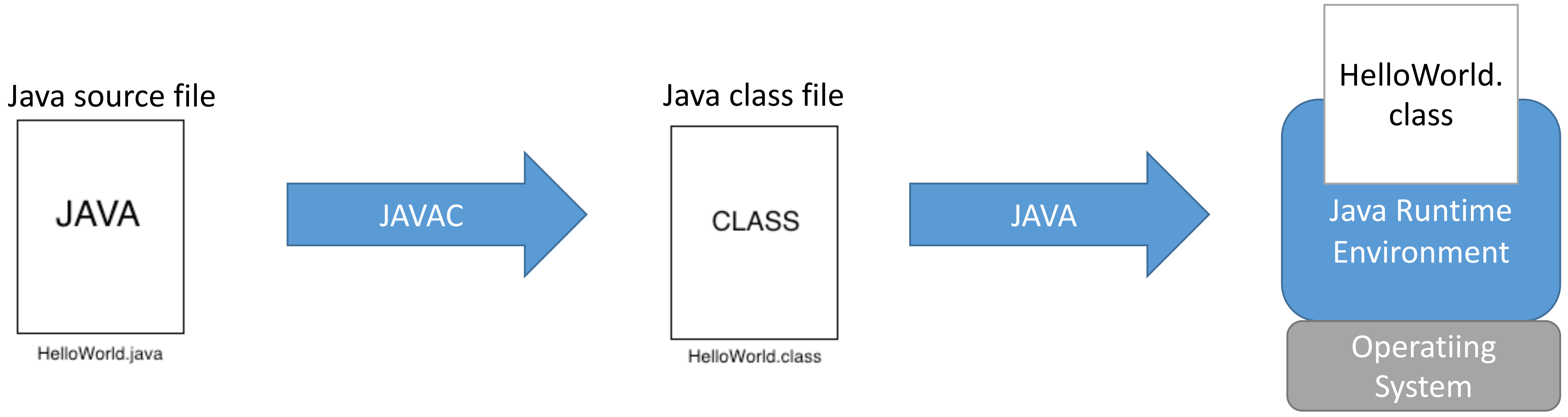


```
System.out
    .println("HelloWorld!");
```

```
GETSTATIC System.out : PrintStream
LDC "HelloWorld!"
INVOKEVIRTUAL PrintStream.println (String) : void
```


Scala source file
Scala
HelloWorld.scala
Kotlin source file
Kotlin
HelloWorld.kt
Groovy source file
Groovy
HelloWorld.groovy
scalac
kotlinc
groovyc
CLASS
HelloWorld.class

Running a Java application



```
javac  HelloWorld.java
```

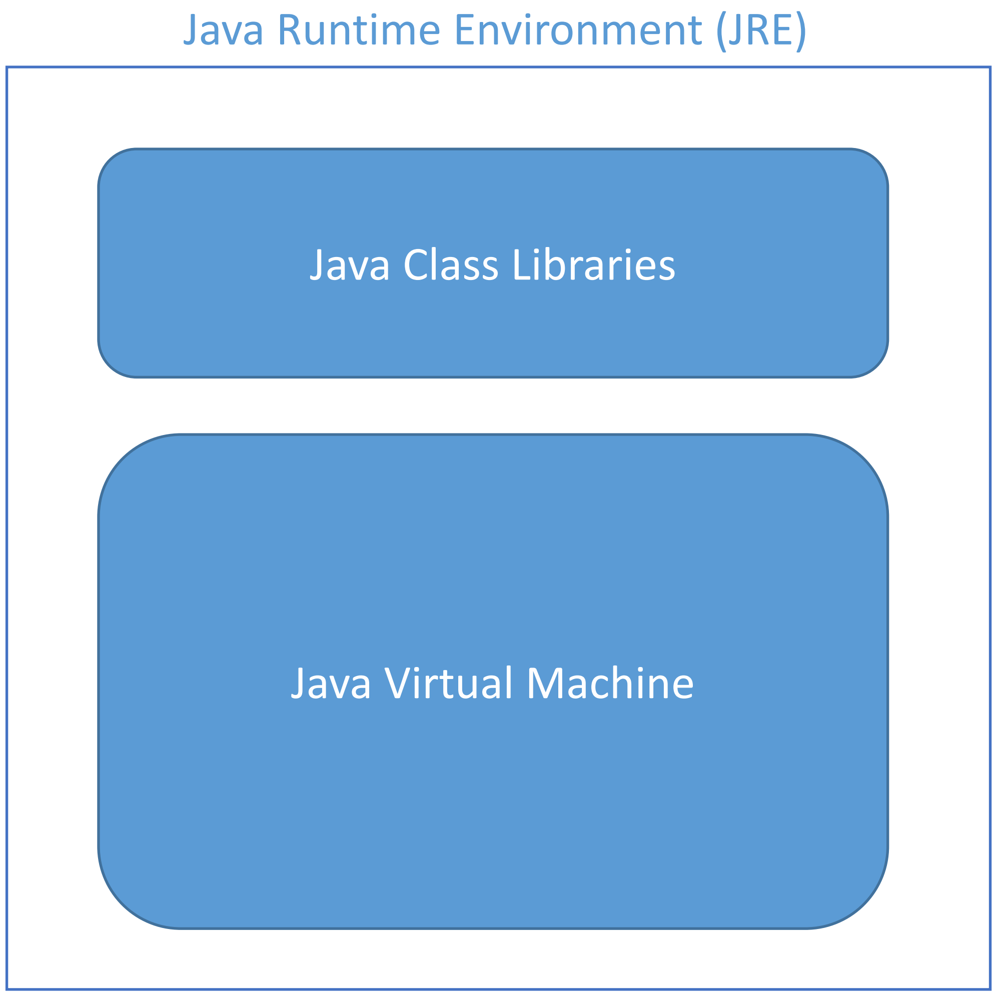
```
java  HelloWorld
```

Launching the jvm

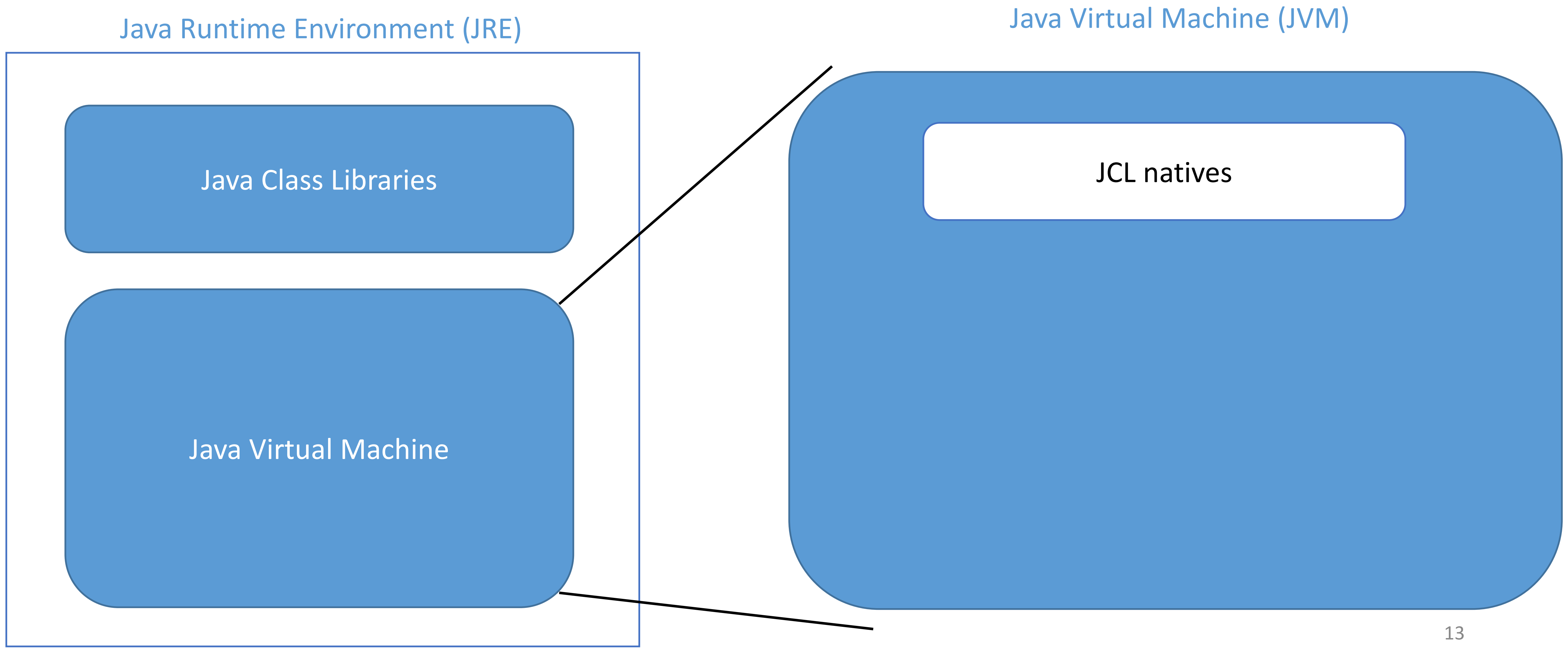
java HelloWorld

```
// create the VM
JNI_CreateJavaVM(JavaVM **p_vm, JNIEnv **env, . . .)

// find main class, ‘HelloWorld’
cls = (*env)->FindClass(env, . . .);

// get main(String[]…) method
mid = (*env)->GetStaticMethodID(env, cls, . . .);

// call it!
(*env)->CallStaticVoidMethod(env, cls, mid. . .);

//shutdown
p_vm->DestroyJavaVM();
```

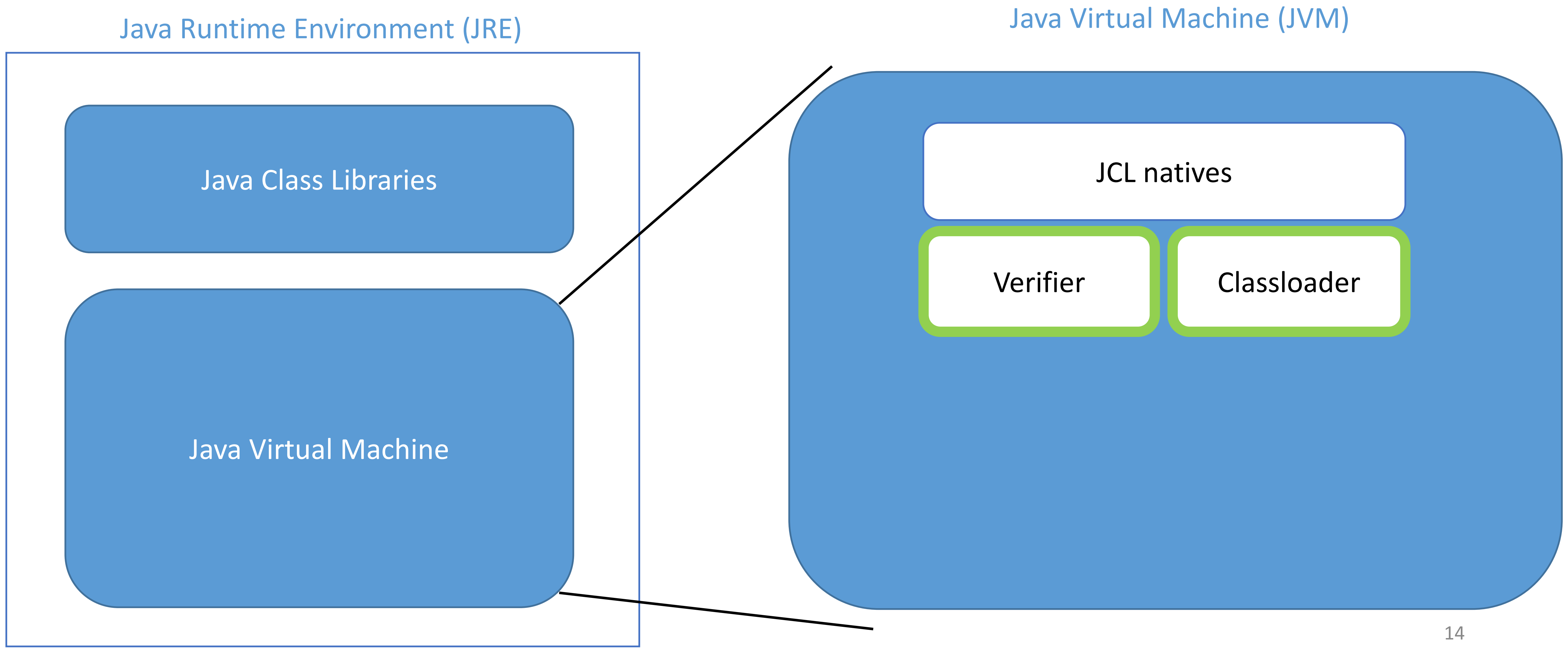
The JRE


Java Runtime Environment (JRE)
Java Class Libraries
Java Virtual Machine

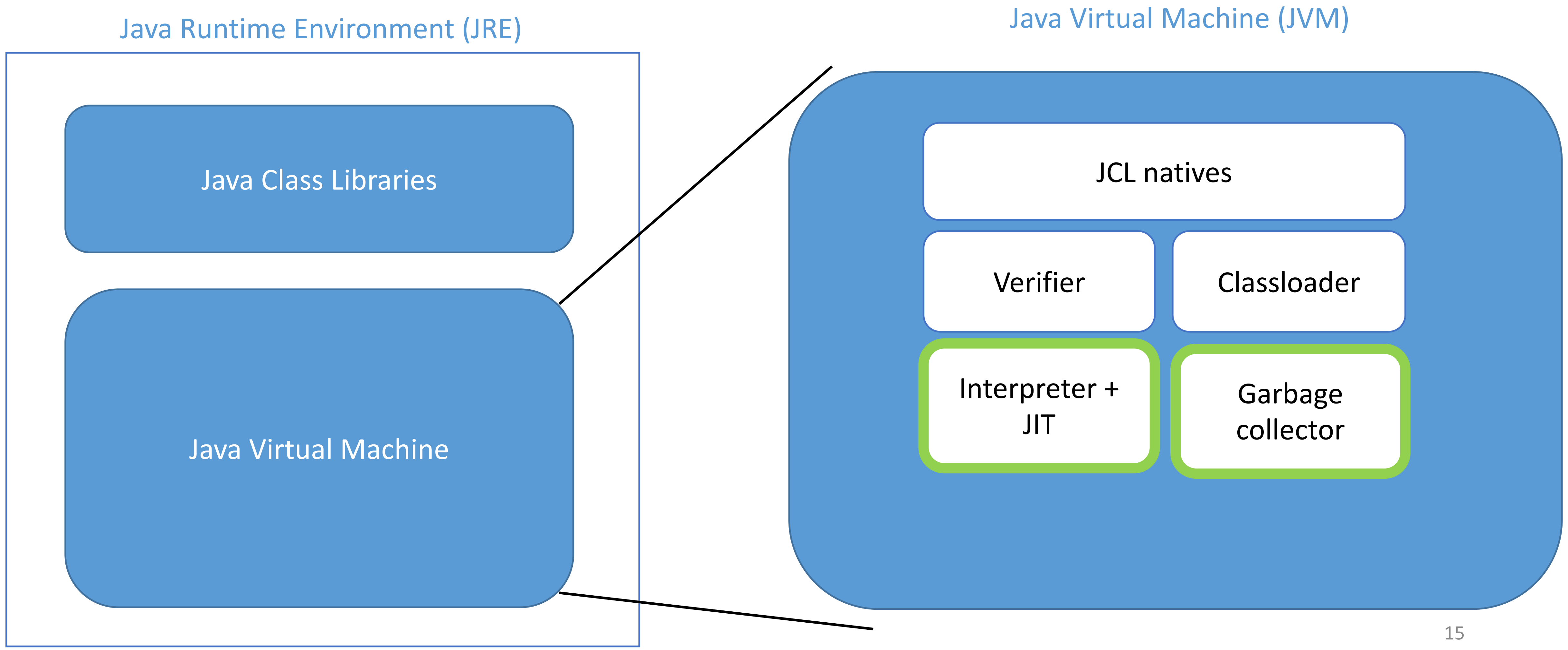
The JRE


Java Runtime Environment (JRE)
Java Class Libraries
Java Virtual Machine
Java Virtual Machine (JVM)
JCL natives

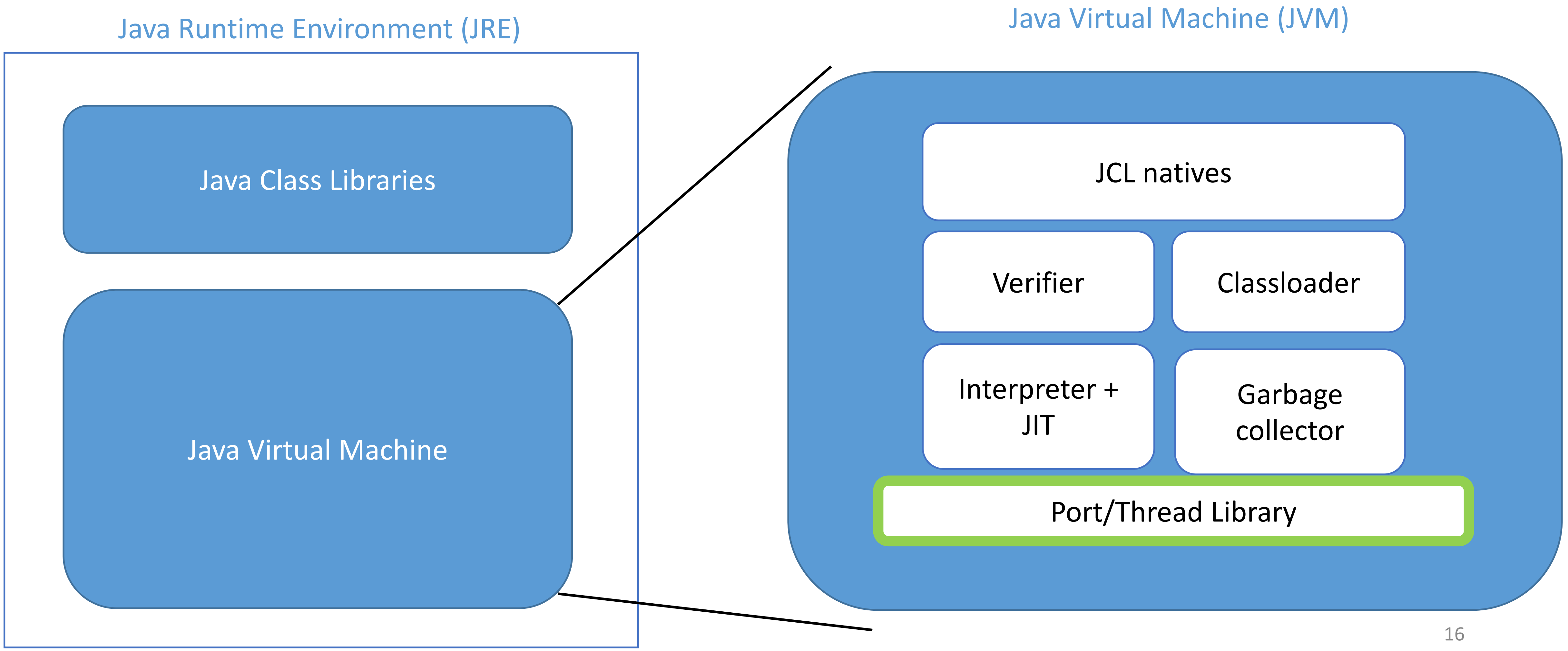
The JRE


Java Runtime Environment (JRE)
Java Class Libraries
Java Virtual Machine
Java Virtual Machine (JVM)
JCL natives
Verifier
Classloader

The JRE


Java Runtime Environment (JRE)
Java Class Libraries
Java Virtual Machine
Java Virtual Machine (JVM)
JCL natives
Verifier
Classloader
Interpreter + JIT
Garbage collector

The JRE


Java Runtime Environment (JRE)
Java Class Libraries
Java Virtual Machine
Java Virtual Machine (JVM)
JCL natives
Verifier
Classloader
Interpreter + JIT
Garbage collector
Port/Thread Library

OpenJ9

Eclipse OpenJ9
Created Sept 2017

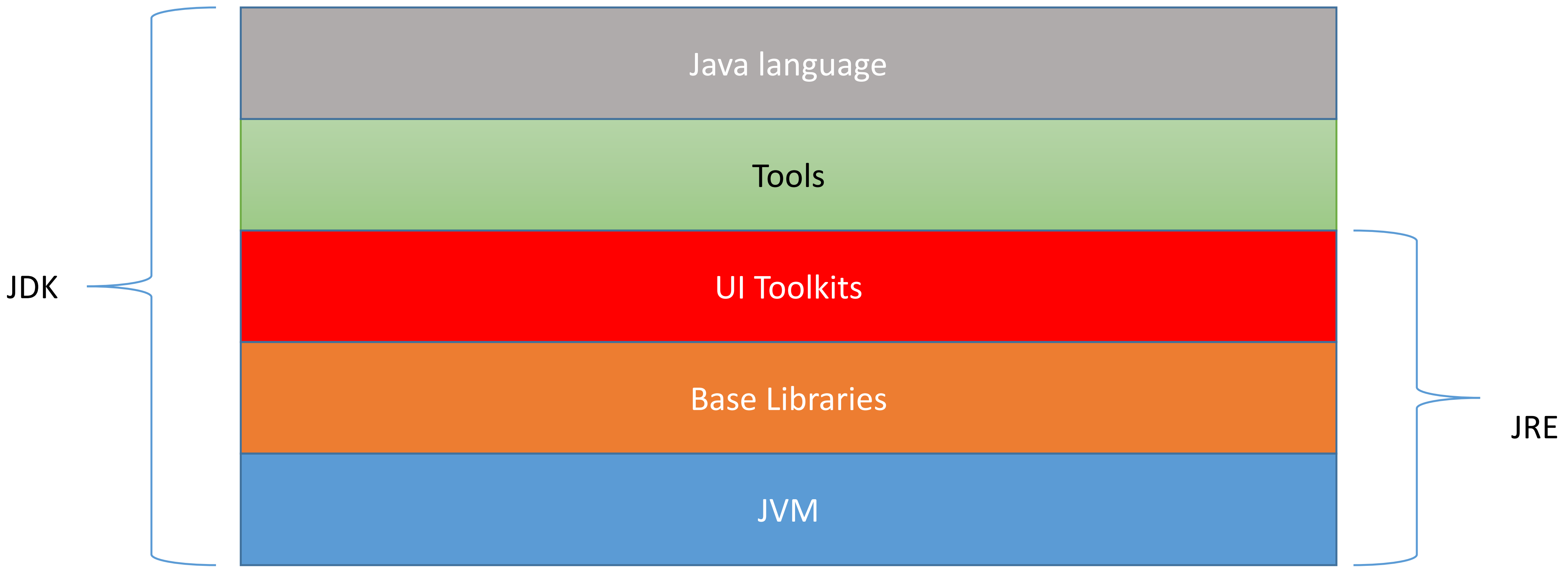
http://www.eclipse.org/openj9
https://github.com/eclipse/openj9

Dual License:
Eclipse Public License v2.0
Apache 2.0

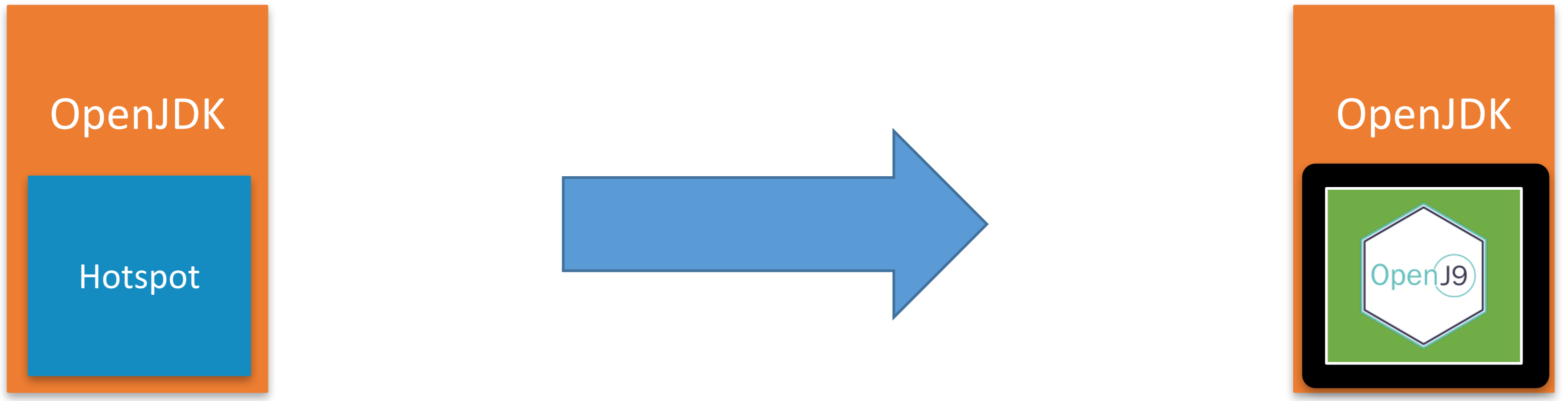
Users and contributors very welcome

https://github.com/eclipse/openj9/blob/master/CONTRIBUTING.md

OpenJDK



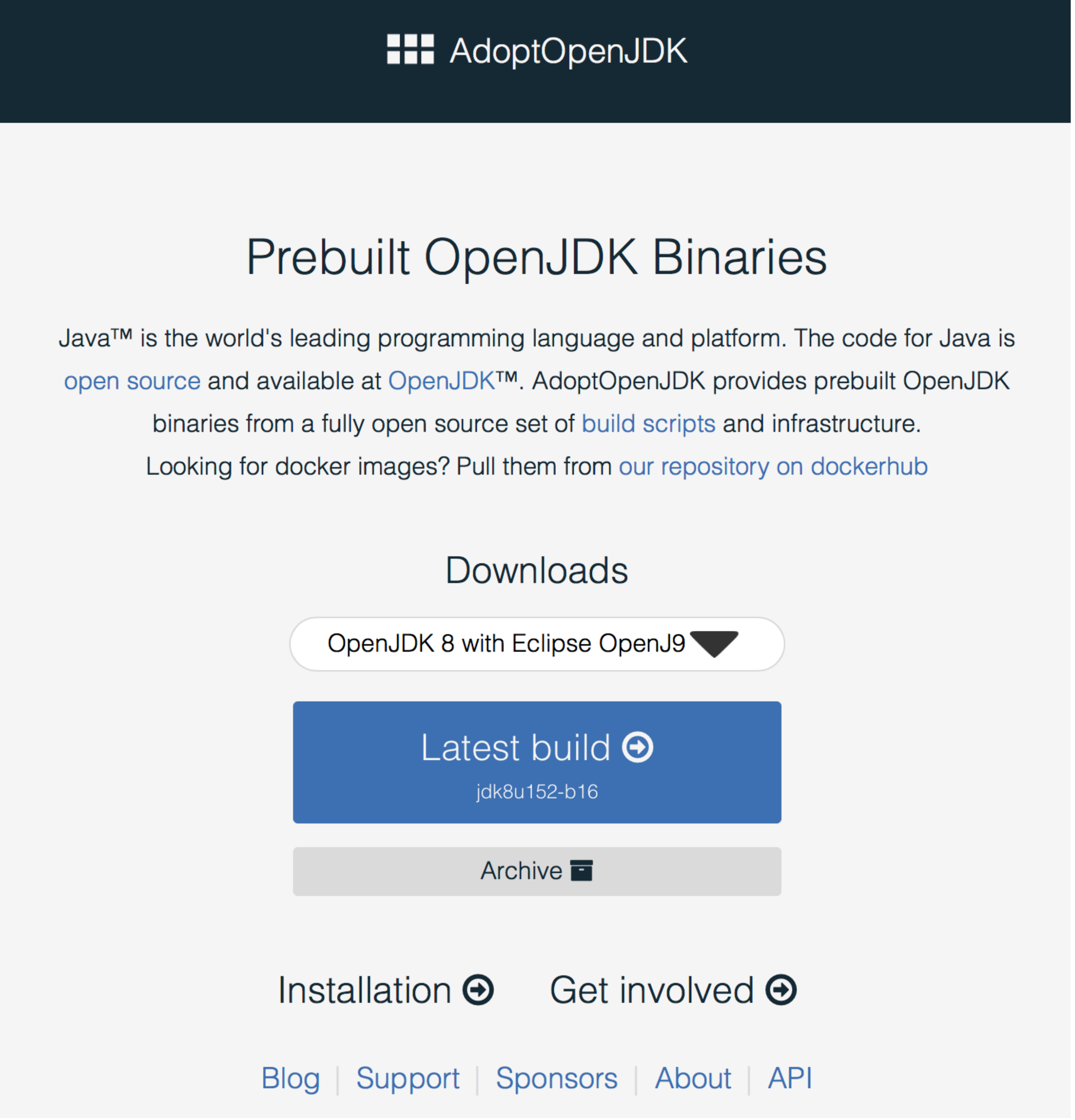
Building OpenJDK with OpenJ9



Building OpenJDK with OpenJ9

```
$ git clone https://github.com/ibmruntimes/openj9-openjdk-jdk9
$ cd openj9-openjdk-jdk9
$ bash ./get_source.sh
$ bash ./configure --with-freemarker-jar=freemarker.jar
$ make images
$ cd build/linux-x86_64-normal-server-release/images/
$ ./jdk/bin/java -version
```

➢**https://www.eclipse.org/openj9/oj9_build.html**


AdoptOpenJDK

Prebuilt OpenJDK Binaries

Java™ is the world's leading programming language and platform. The code for Java is open source and available at OpenJDK™. AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of build scripts and infrastructure.
Looking for docker images? Pull them from our repository on dockerhub

Downloads

OpenJDK 8 with Eclipse OpenJ9

Latest build
jdk8u152-b16

Archive

Installation   Get involved

Blog | Support | Sponsors | About | API

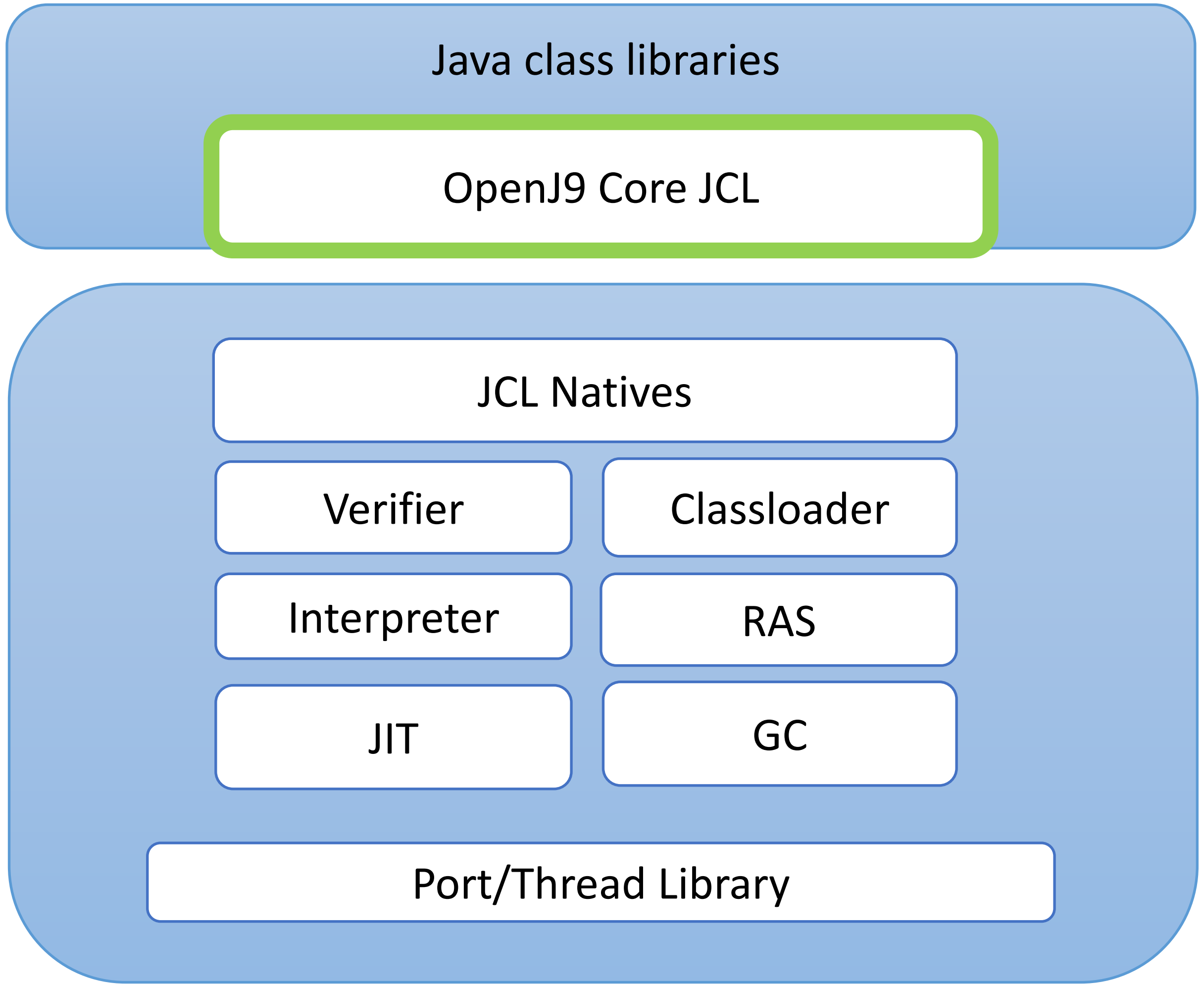
The place to get OpenJDK builds

For both:
- OpenJDK &
- OpenJDK with Eclipse OpenJ9

https://adoptopenjdk.net/releases.html?variant=openjdk9-openj9

How does OpenJ9 work?

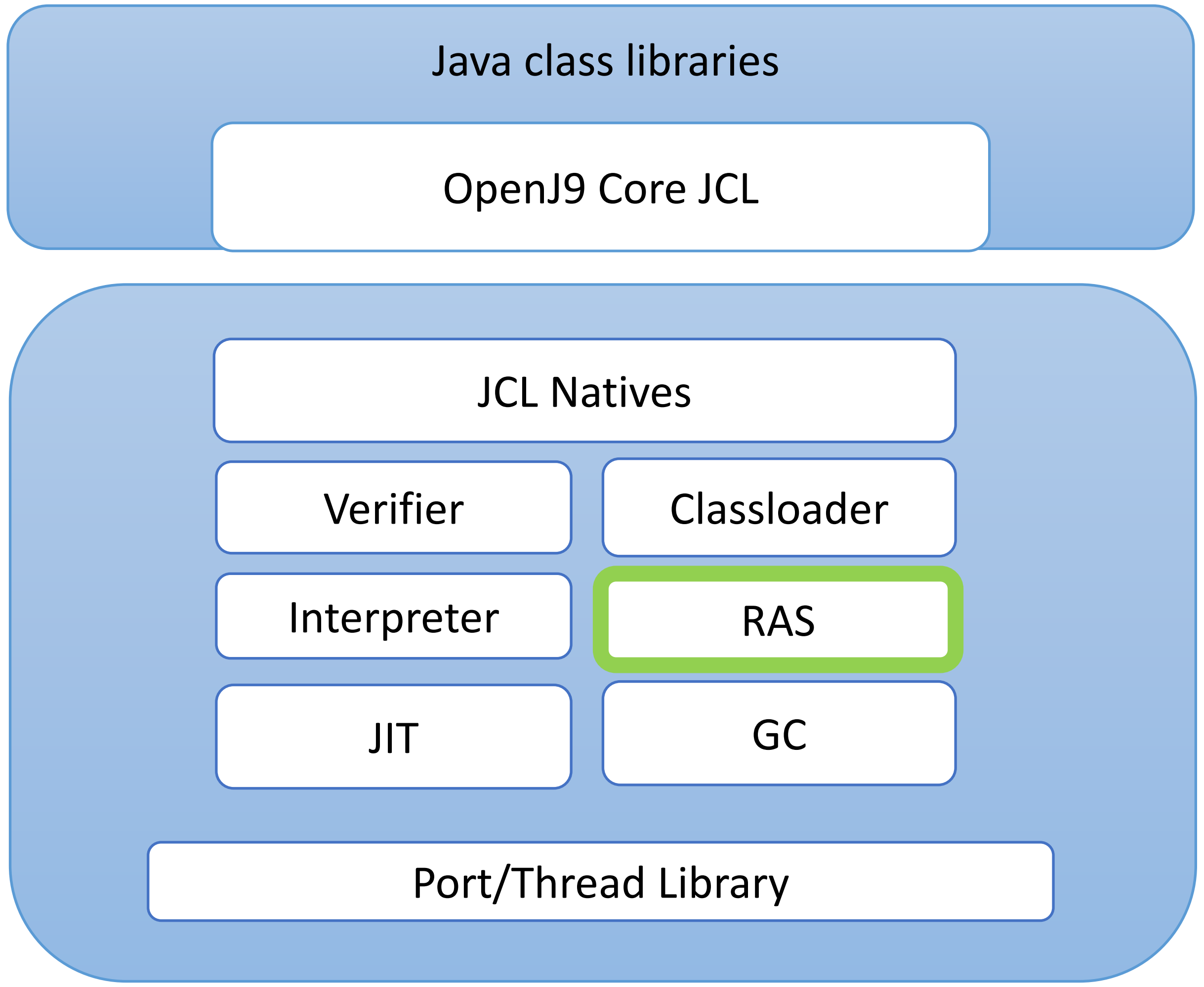
Core Java class libraries
- Includes classes that are closely tied to the JVM implementation
- These include j.l.Object, j.l.Class, j.l.Thread, …



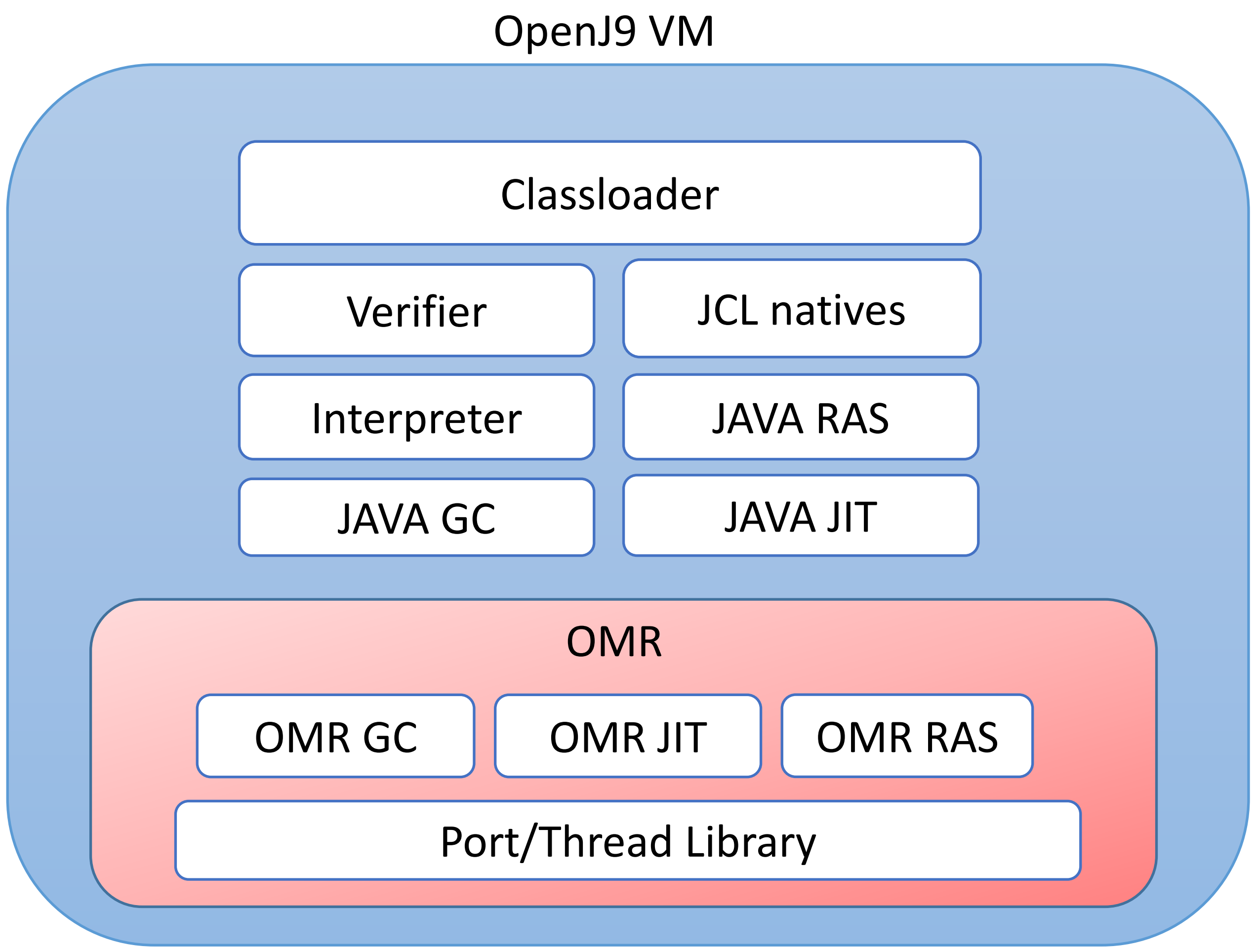
How does OpenJ9 work?

Reliability, Availability and Serviceability (RAS)

- Tools to simplify JVM and application debugging
- Trace engine
- Verification checking utilities



OpenJ9 and OMR



Eclipse OMR
Created March 2016

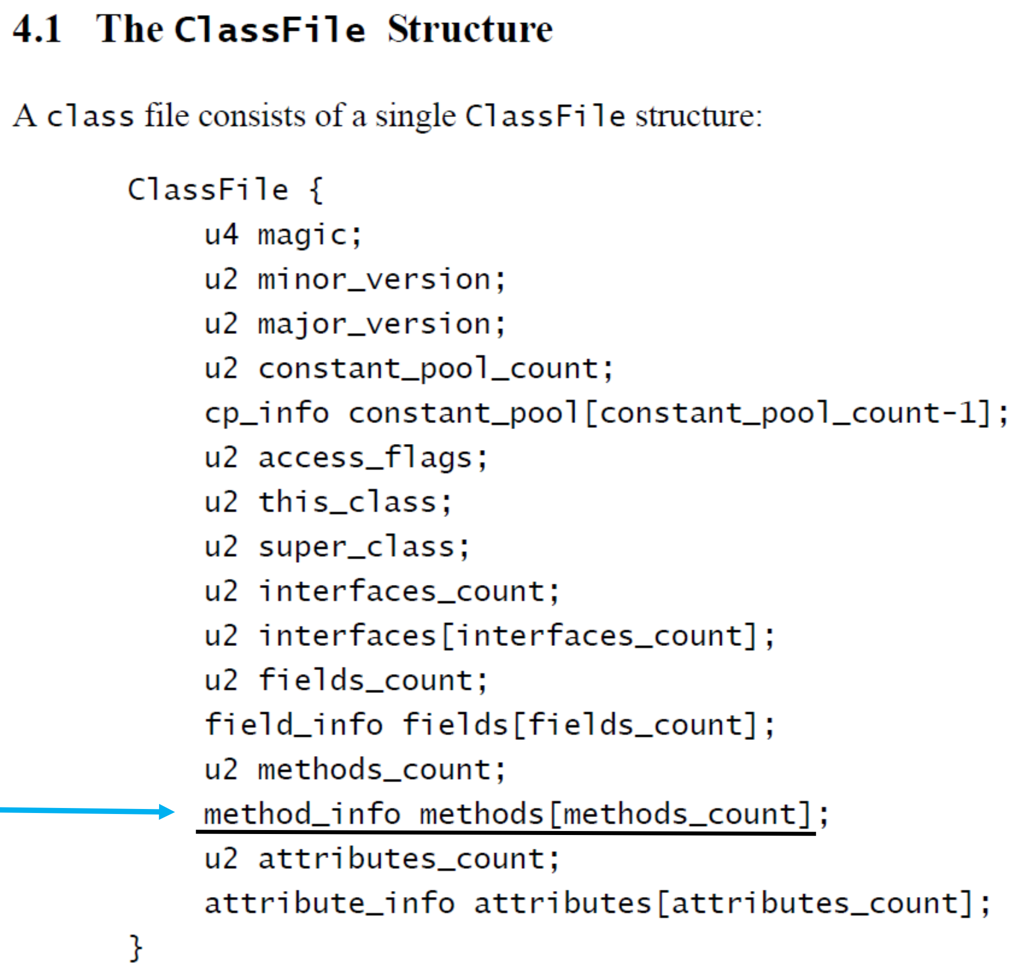
http://www.eclipse.org/omr
https://github.com/eclipse/omr
https://developer.ibm.com/open/omr/

Dual License:
Eclipse Public License v2.0
Apache 2.0

Users and contributors very welcome
https://github.com/eclipse/omr/blob/master/CONTRIBUTING.md

OpenJ9: Classloading

```
System.out
    .println("HelloWorld!");
```



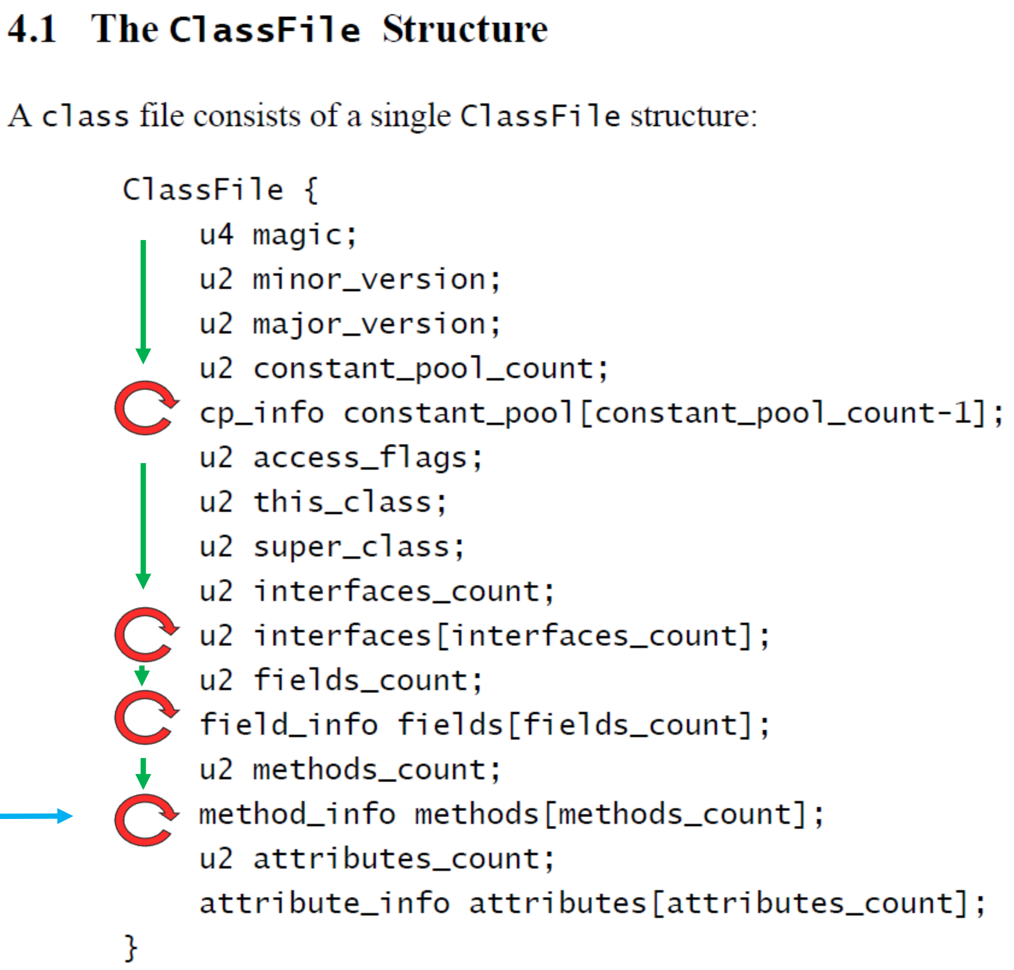
4.1 The ClassFile Structure

A class file consists of a single ClassFile structure:

```
ClassFile {
    u4 magic;
    u2 minor_version;
    u2 major_version;
    u2 constant_pool_count;
    cp_info constant_pool[constant_pool_count-1];
    u2 access_flags;
    u2 this_class;
    u2 super_class;
    u2 interfaces_count;
    u2 interfaces[interfaces_count];
    u2 fields_count;
    field_info fields[fields_count];
    u2 methods_count;
    method_info methods[methods_count];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

OpenJ9: Classloading

```
System.out
    .println("HelloWorld!");
```



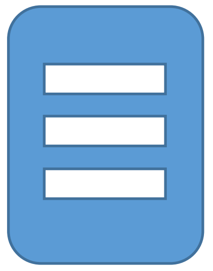
4.1 The `ClassFile` Structure

A `class` file consists of a single `ClassFile` structure:

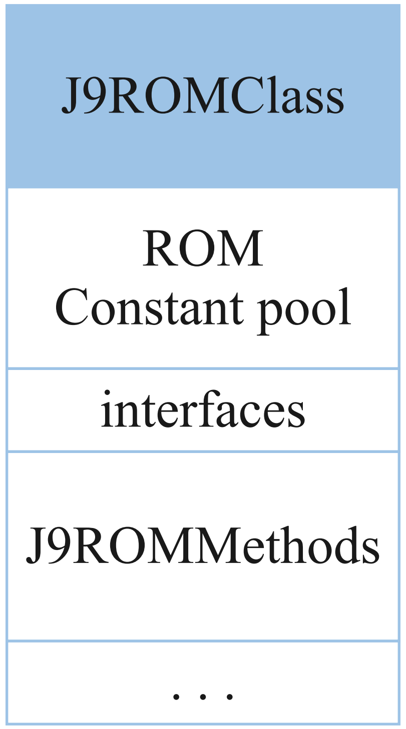
```
ClassFile {
    u4 magic;
    u2 minor_version;
    u2 major_version;
    u2 constant_pool_count;
    cp_info constant_pool[constant_pool_count-1];
    u2 access_flags;
    u2 this_class;
    u2 super_class;
    u2 interfaces_count;
    u2 interfaces[interfaces_count];
    u2 fields_count;
    field_info fields[fields_count];
    u2 methods_count;
    method_info methods[methods_count];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

OpenJ9: ROMClass

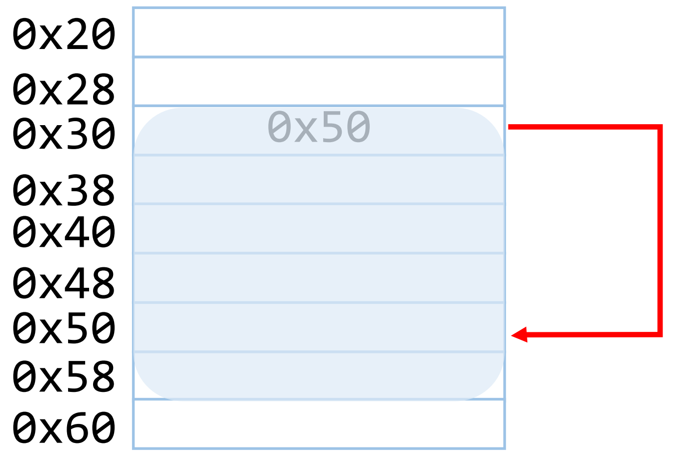
- Keep all the symbolic info from a classfile
- Remove variability (where possible)
- Position independent: map anywhere in the address space
- ROM: written once, only read after
    - Learn from the Smalltalk/Embedded past



```
typedef struct J9ROMClass {
   U_32 romSize;
   U_32 singleScalarStaticCount;
   J9SRP className;
   J9SRP superclassName;
   U_32 modifiers;
   U_32 extraModifiers;
   U_32 interfaceCount;
   J9SRP interfaces;
   U_32 romMethodCount;
   J9SRP romMethods;
   U_32 romFieldCount;
   J9SRP romFields;
   U_32 objectStaticCount;
   U_32 doubleScalarStaticCount;
   U_32 ramConstantPoolCount;
   U_32 romConstantPoolCount;
   ...
} J9ROMClass;
```

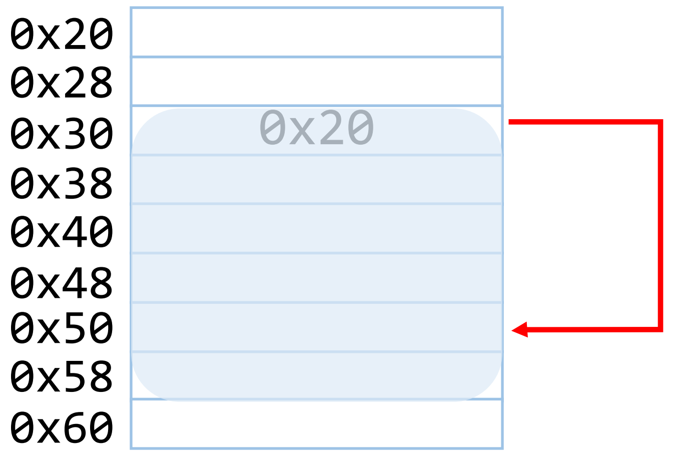


OpenJ9: Self relative pointers



Regular pointer

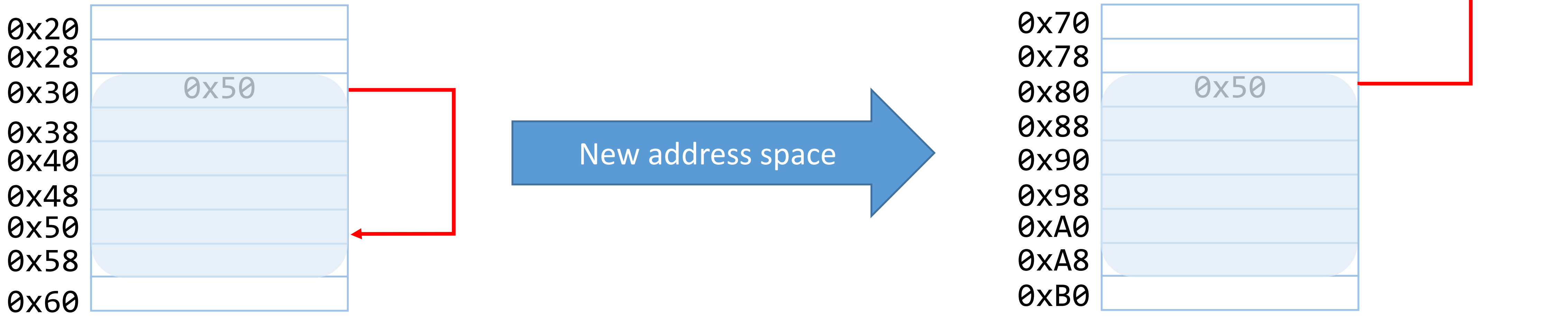
`Address of slot + value = target address`



`0x30 + *(0x30) = 0x30 + 0x20 = 0x50`

Self relative pointer

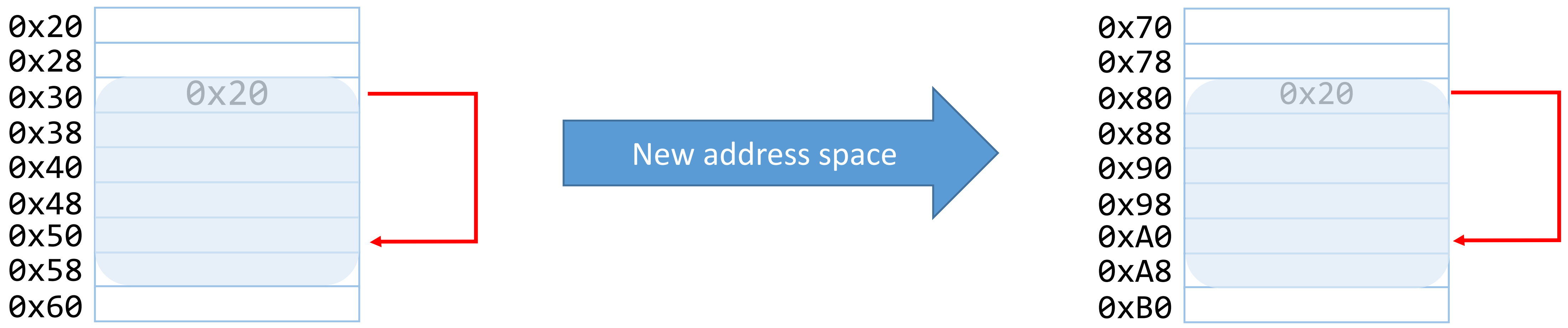
OpenJ9: Self relative pointers



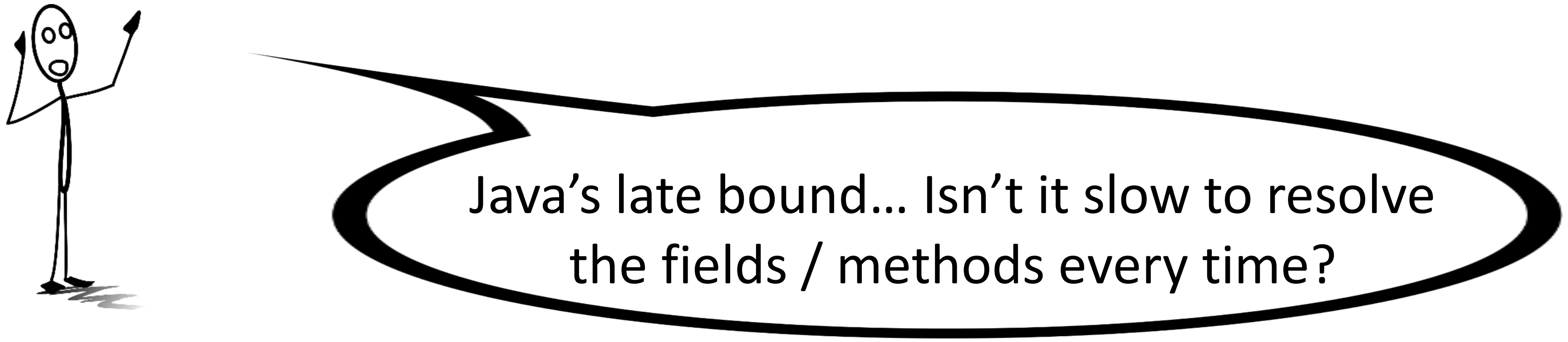
Need to update pointer

Regular pointer

OpenJ9: Self relative pointers

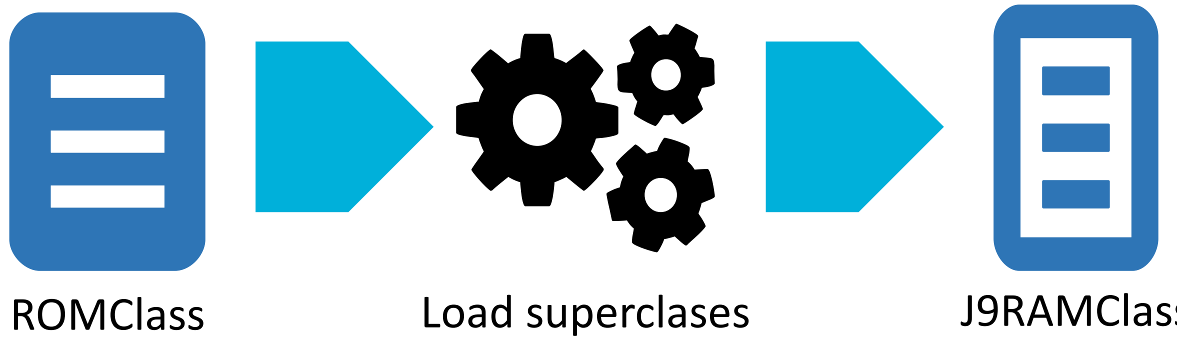


self relative pointer


Java’s late bound… Isn’t it slow to resolve the fields / methods every time?

```
public static void main(java.lang.String[])
  0 JBgetstatic 2 System.out LPrintStream;
  3 JBldc 3 (java.lang.String) "HelloWorld”
→ 5 JBinvokevirtual 4 PrintStream.println(LString;)V
  8 JBreturn
```

OpenJ9: ROM to RAM

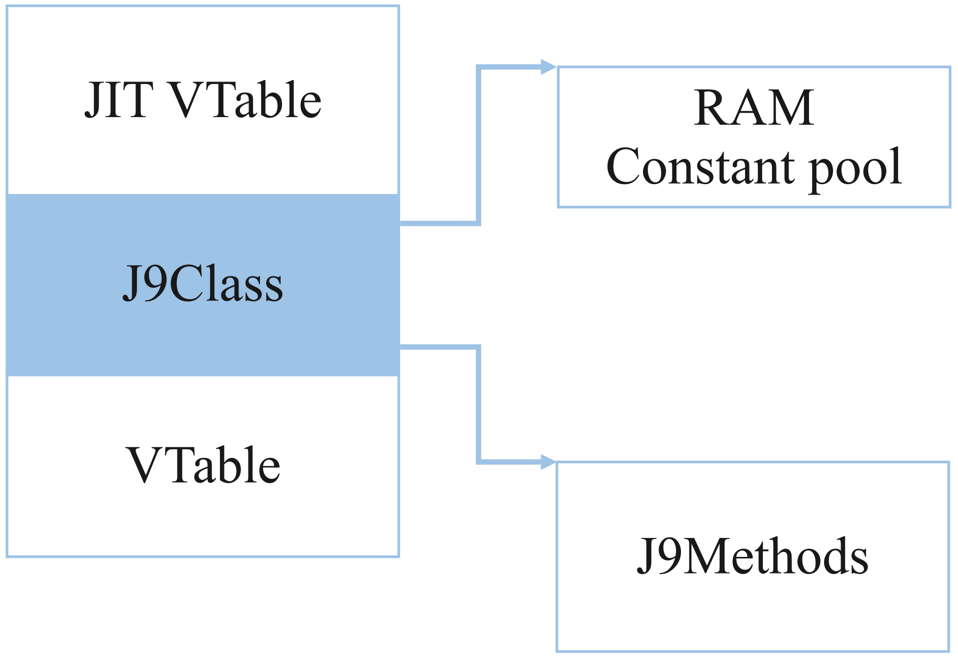


Load superclases
Load superinterfaces
Calculate vtable
Calculate field offsets

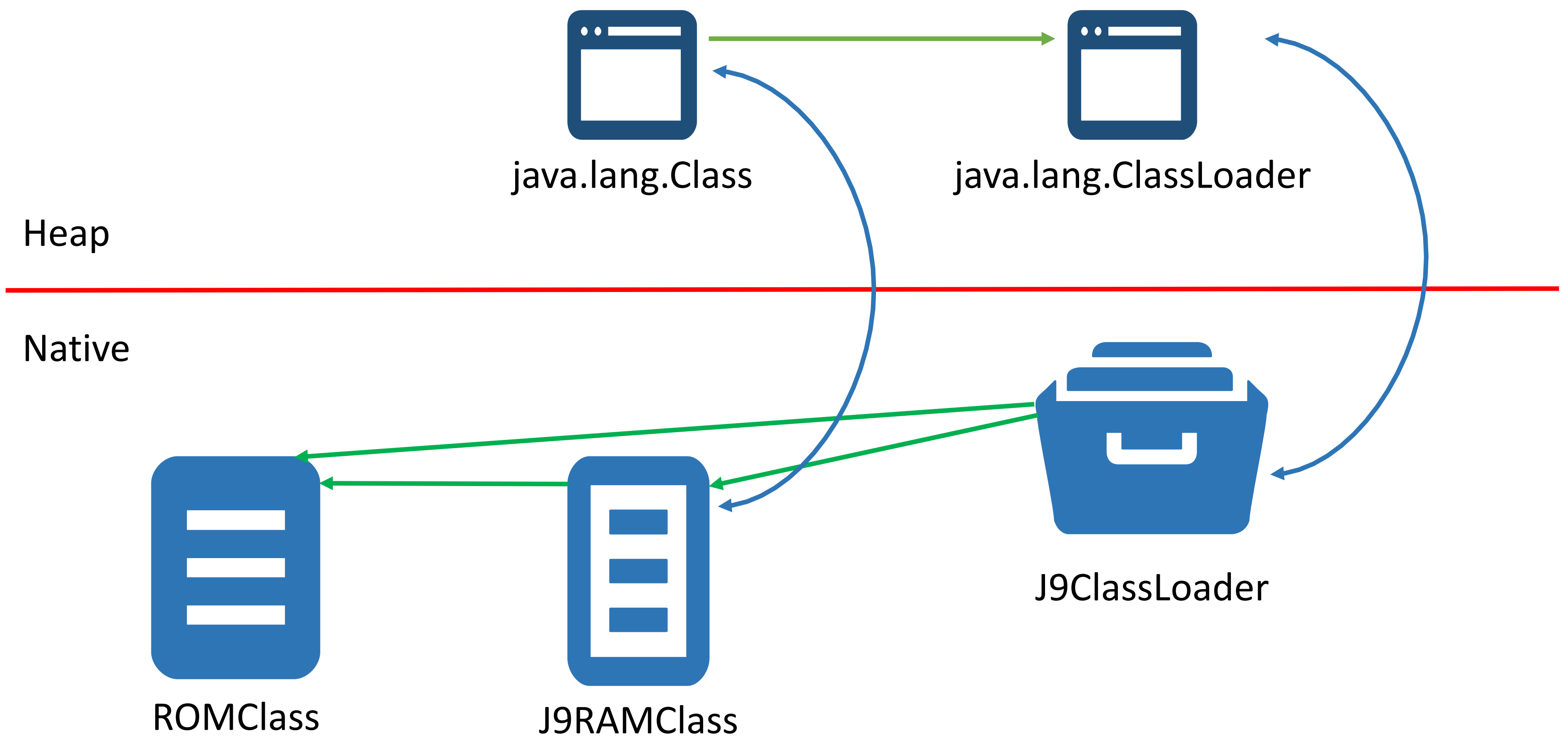
https://github.com/eclipse/openj9/blob/master/runtime/vm/createramclass.cpp

OpenJ9: RamClass

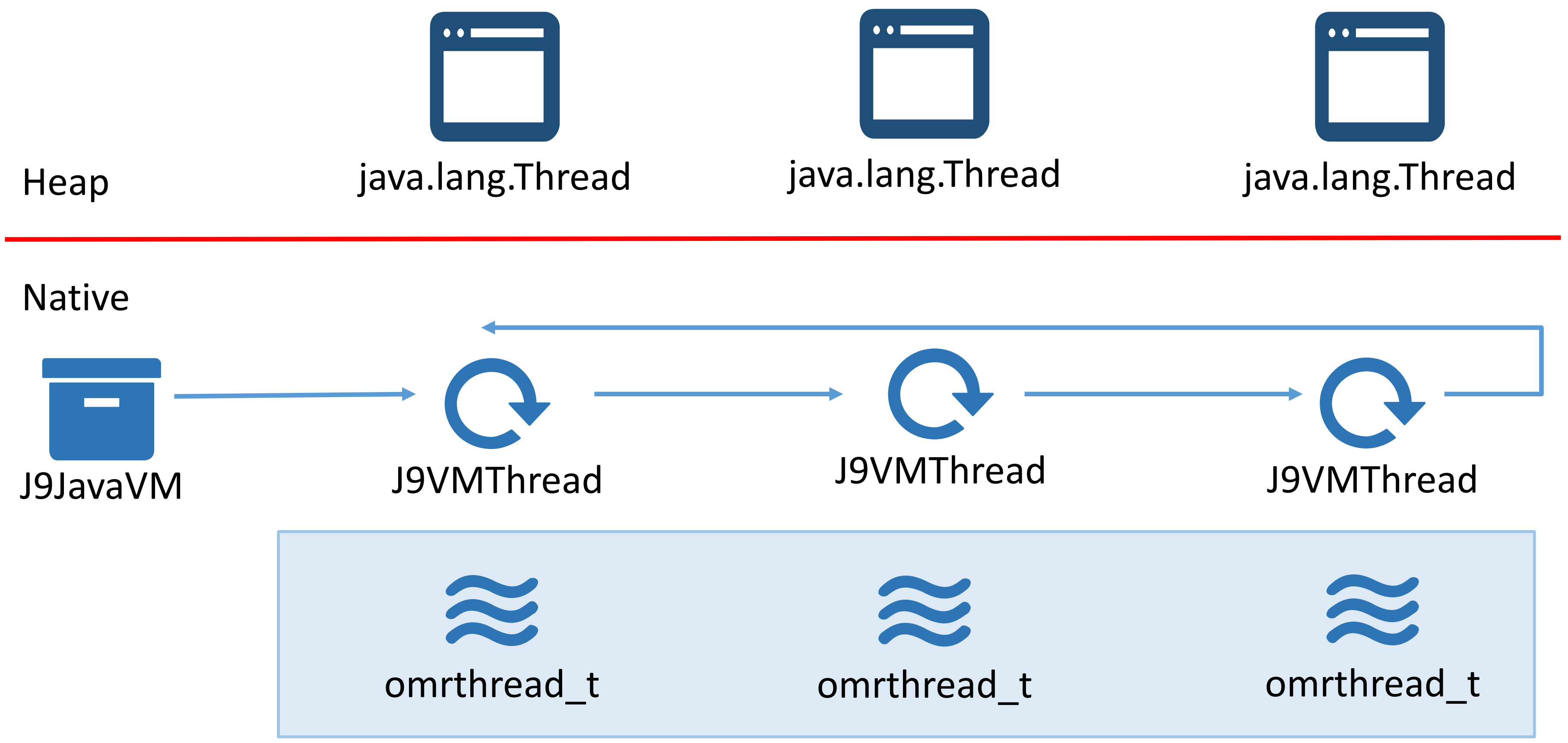
```
typedef struct J9Class {
   UDATA eyecatcher;
   struct J9ROMClass* romClass;
   struct J9Class** superclasses;
   UDATA classDepthAndFlags;
   struct J9ClassLoader* classLoader;
   j9object_t classObject;
   UDATA volatile initializeStatus;
   struct J9Method* ramMethods;
   UDATA* ramStatics;
   struct J9Class* arrayClass;
   UDATA totalInstanceSize;
   UDATA* instanceDescription;
   UDATA packageID;
   void** iTable;
   void** jniIDs;
   struct J9Class* replacedClass;
   UDATA* ramConstantPool;
   . . .
} J9Class;
```



OpenJ9: Classloading

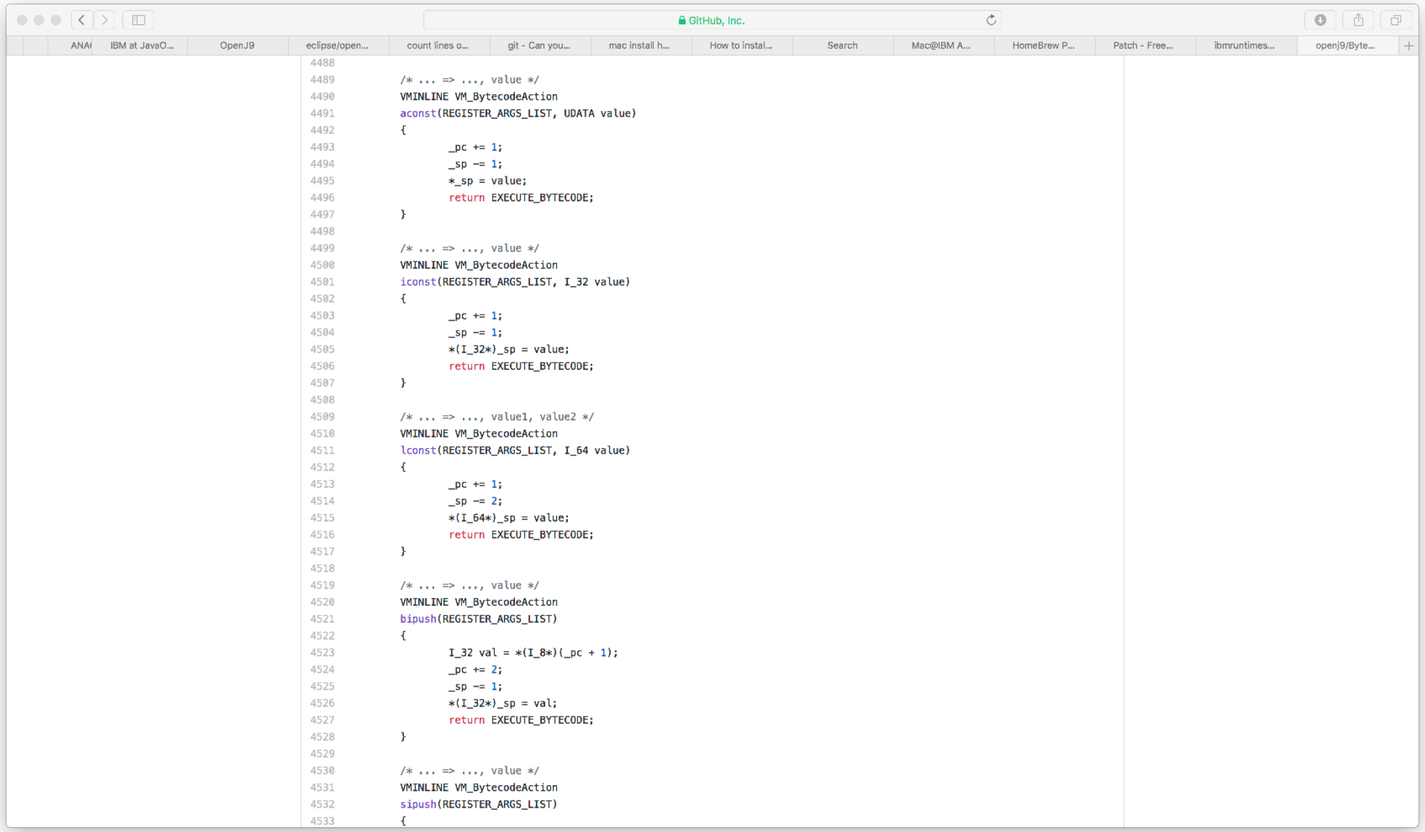


OpenJ9: Threading



OpenJ9: Threading

- Cooperative threading model
- Thread states
  - VM access
  - Exclusive access
  - Wait, Sleep, Blocked
  - Halted (exclusive)

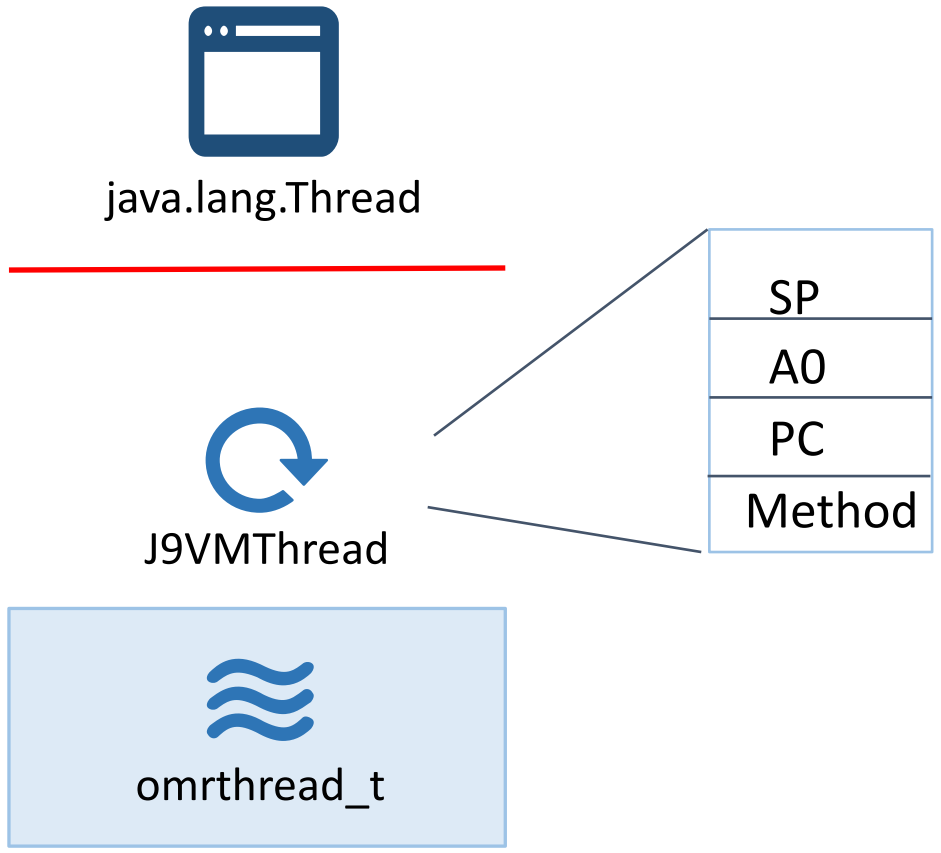


```
	/* ... => ..., value */
	VMINLINE VM_BytecodeAction
	aconst(REGISTER_ARGS_LIST, UDATA value)
	{
		_pc += 1;
		_sp -= 1;
		*_sp = value;
		return EXECUTE_BYTECODE;
	}

	/* ... => ..., value */
	VMINLINE VM_BytecodeAction
	iconst(REGISTER_ARGS_LIST, I_32 value)
	{
		_pc += 1;
		_sp -= 1;
		*(I_32*)_sp = value;
		return EXECUTE_BYTECODE;
	}

	/* ... => ..., value1, value2 */
	VMINLINE VM_BytecodeAction
	lconst(REGISTER_ARGS_LIST, I_64 value)
	{
		_pc += 1;
		_sp -= 2;
		*(I_64*)_sp = value;
		return EXECUTE_BYTECODE;
	}

	/* ... => ..., value */
	VMINLINE VM_BytecodeAction
	bipush(REGISTER_ARGS_LIST)
	{
		I_32 val = *(I_8*)(_pc + 1);
		_pc += 2;
		_sp -= 1;
		*(I_32*)_sp = val;
		return EXECUTE_BYTECODE;
	}

	/* ... => ..., value */
	VMINLINE VM_BytecodeAction
	sipush(REGISTER_ARGS_LIST)
	{
```

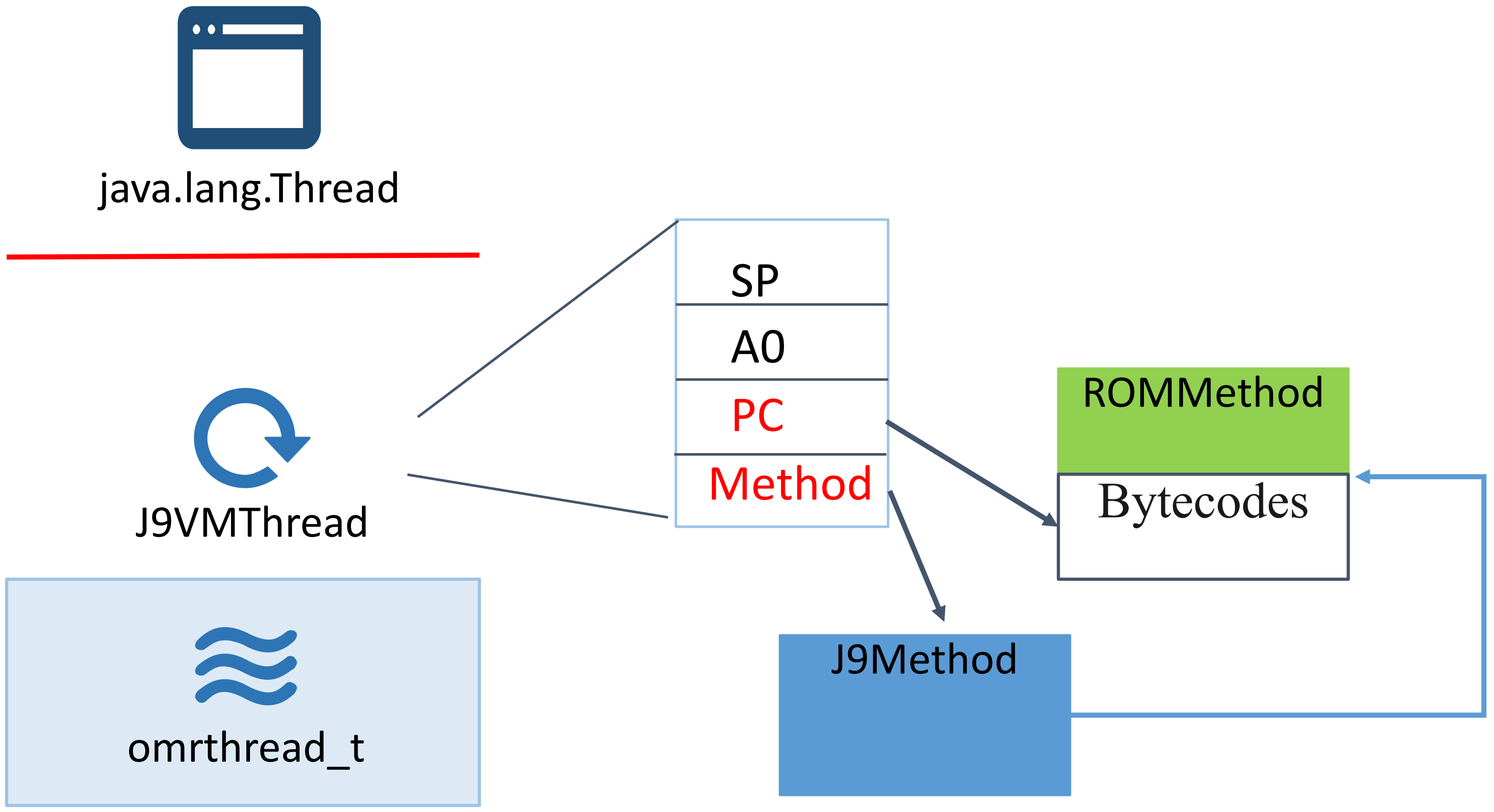
OpenJ9: Interpreter

- Written in C++
- Switch statement / computed goto
- Executes:
  - bytecodes
  - INLs
  - builds stack frames
- Transition to the JIT

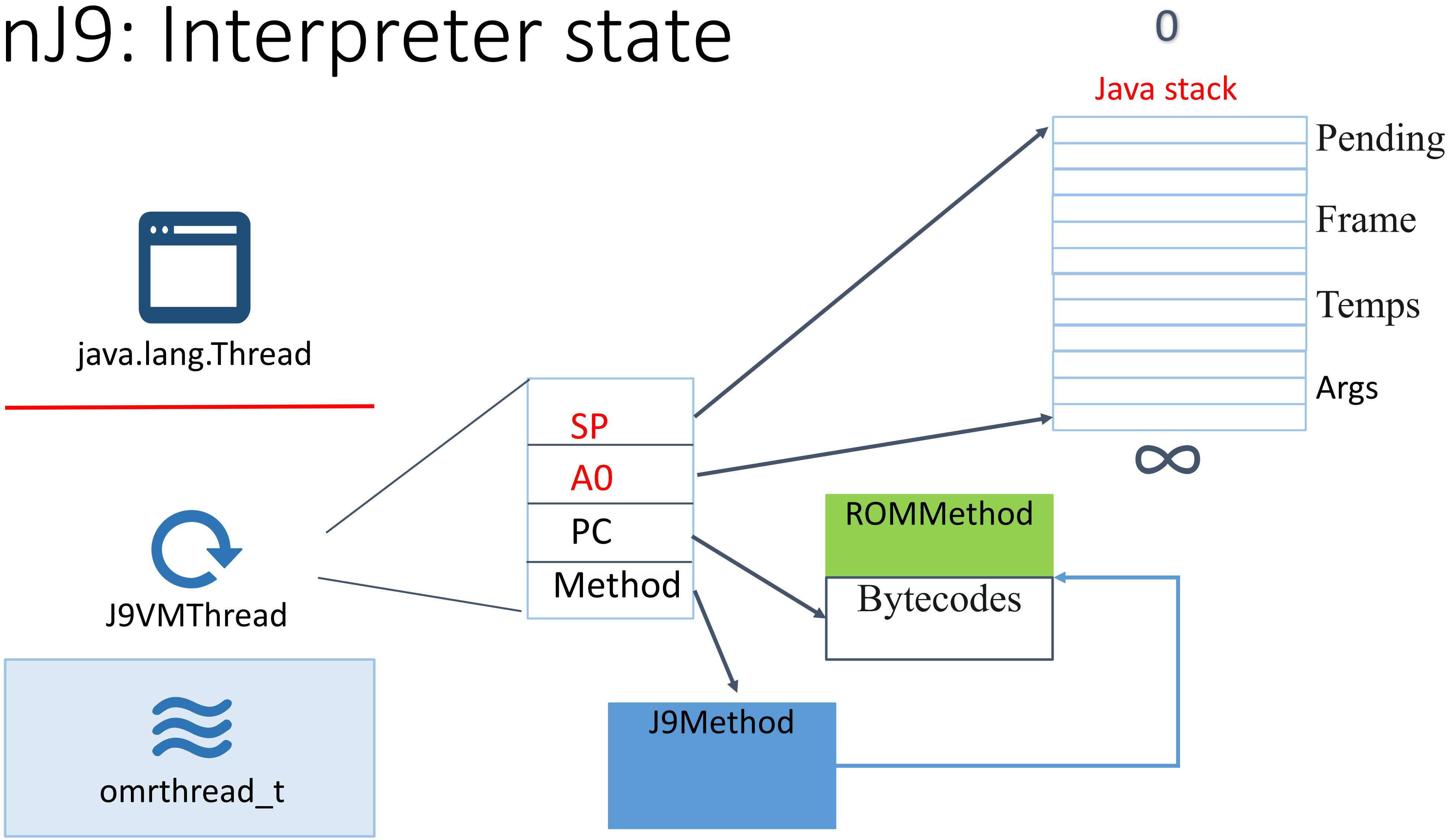
OpenJ9: Interpreter state



OpenJ9: Interpreter state

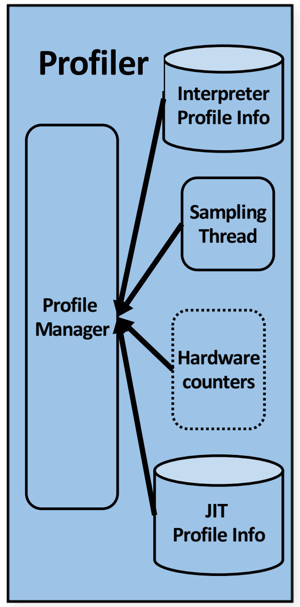


OpenJ9: Interpreter state

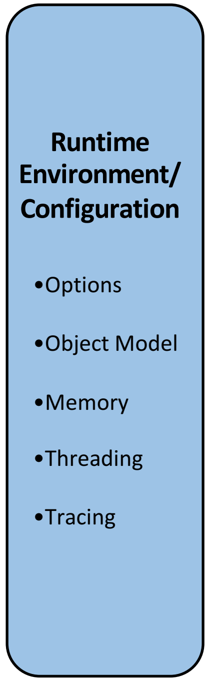


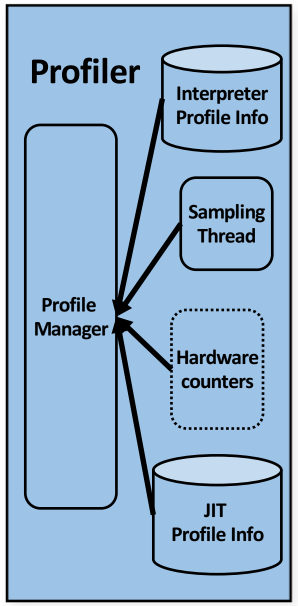
OpenJ9: Testarossa JIT

OpenJ9: Testarossa JIT compiler

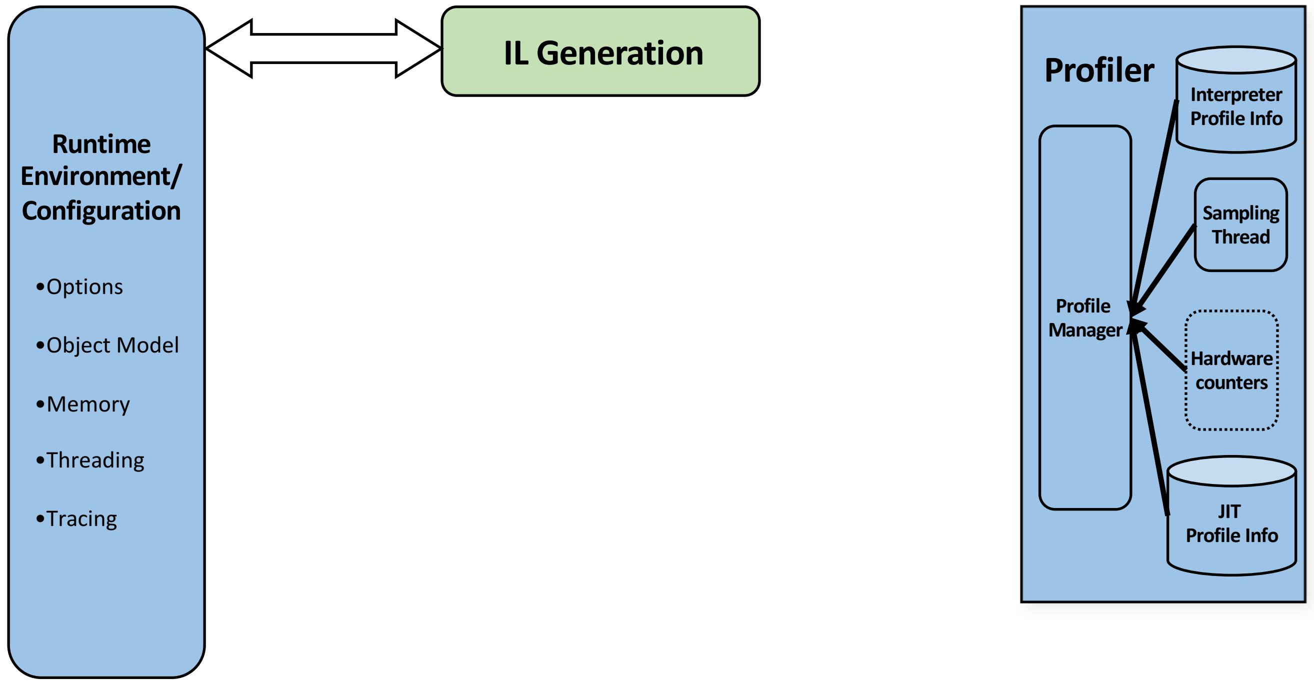


OpenJ9: Testarossa JIT compiler

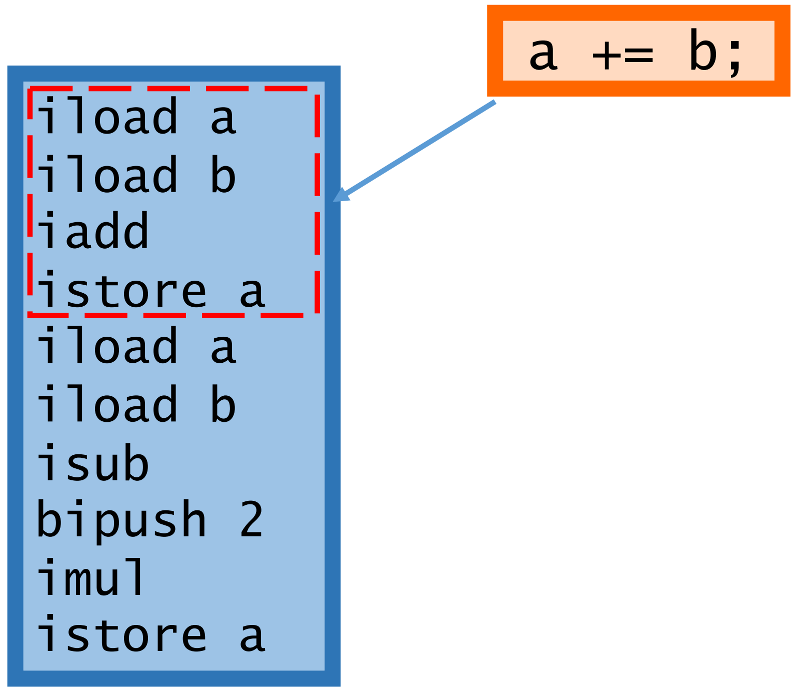




OpenJ9: Testarossa JIT compiler



ILGen



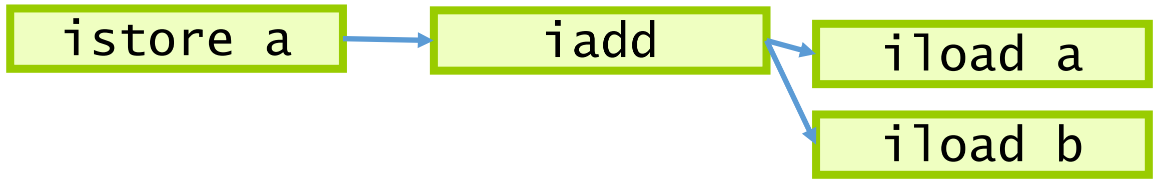
```
a += b;
```

```
iload a
iload b
iadd
istore a
iload a
iload b
isub
bipush 2
imul
istore a
```

ILGen

```
iload a
iload b
iadd
istore a
iload a
iload b
isub
bipush 2
imul
istore a
```

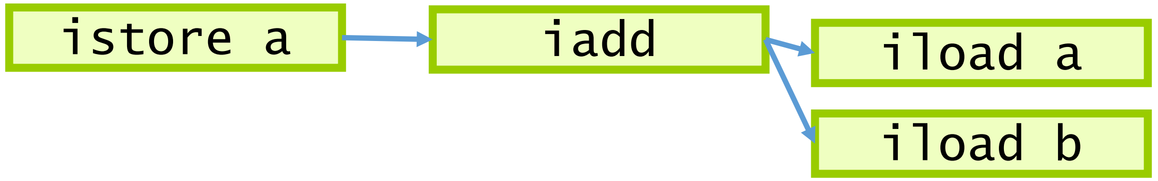
```
a += b;
```



ILGen

```
iload a
iload b
iadd
istore a
iload a
iload b
isub
bipush 2
imul
istore a
```

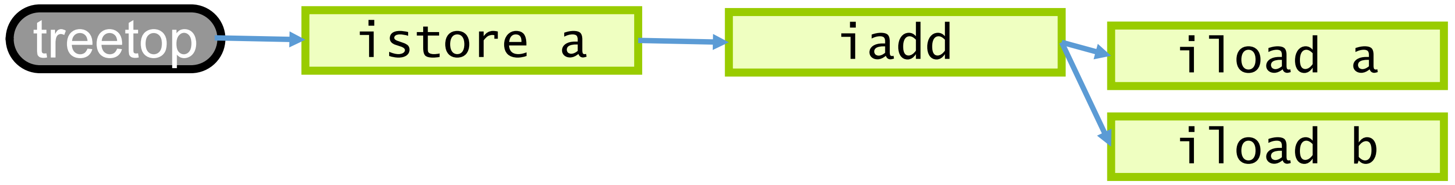
```
a += b;
```



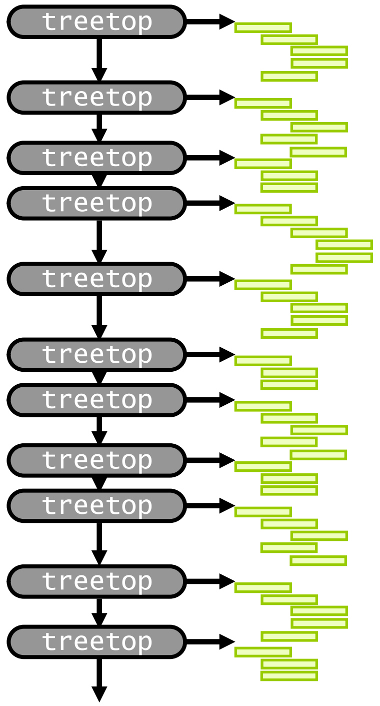
IL Generator

```
iload a
iload b
iadd
istore a
iload a
iload b
isub
bipush 2
imul
istore a
```

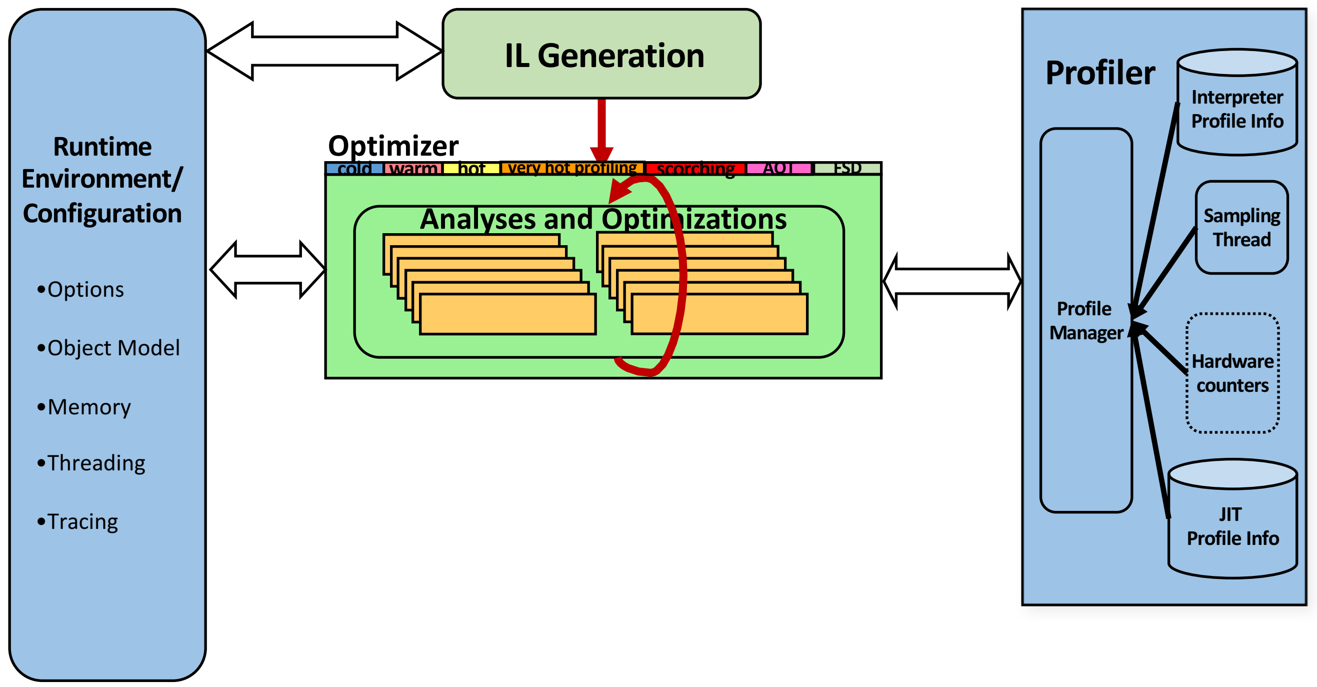
```
a += b;
```



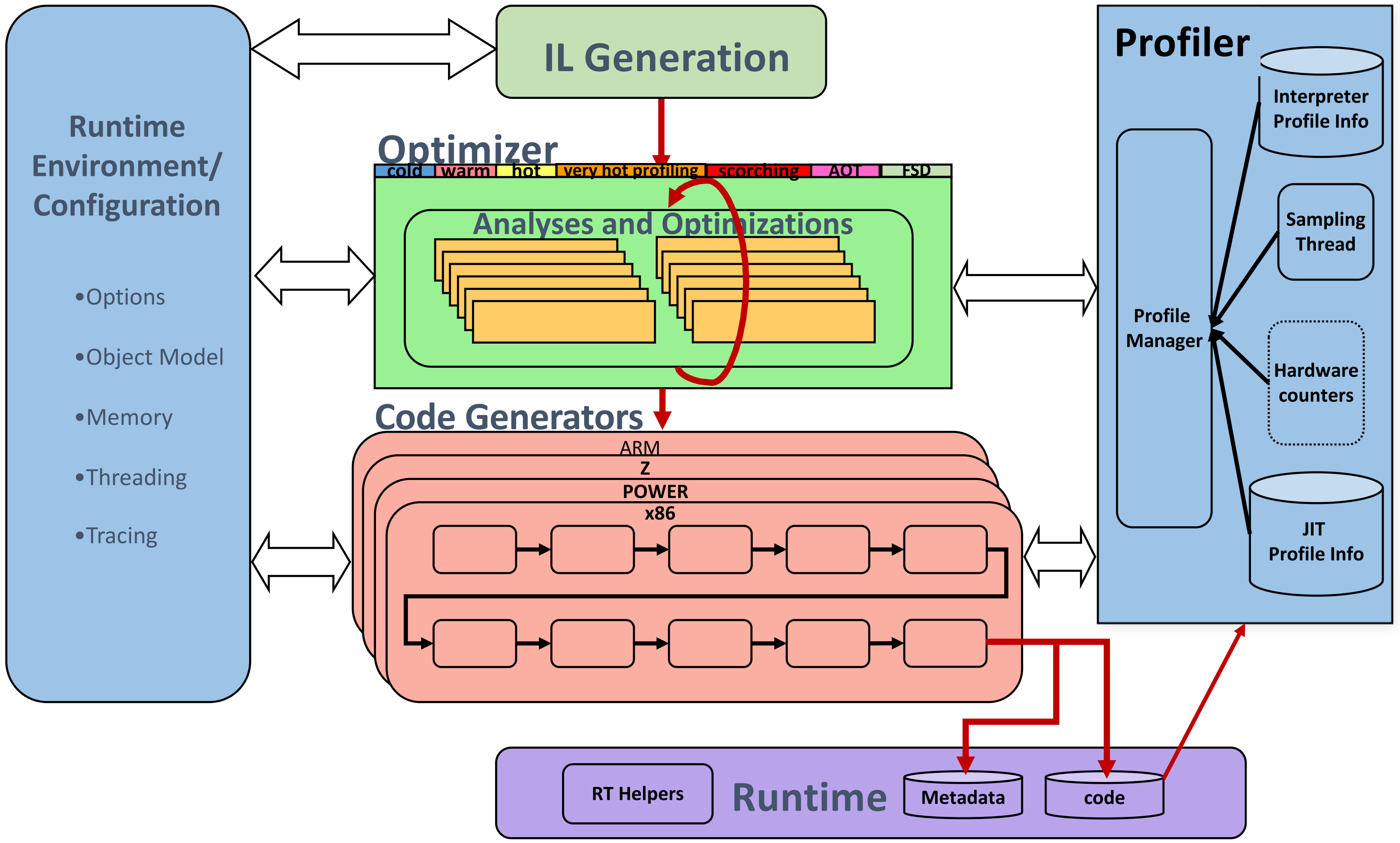
IL Generator



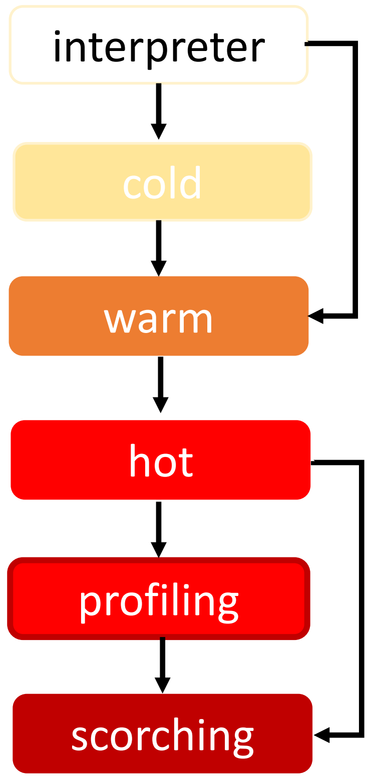
OpenJ9: Testarossa JIT compiler



JIT Compilation



Adaptive JIT Compilation



- Methods start out running bytecode form directly
- After many invocations (or via sampling) code get compiled at 'cold' or 'warm' level
- Low overhead sampling thread is used to identify hot methods
- Methods may get recompiled at 'hot' or 'scorching' levels (for more optimizations)
- Transition to 'scorching' goes through a temporary profiling step

OpenJ9: Object model

Field and array element sizes

Object and field alignment

compressed vs non-compressed references

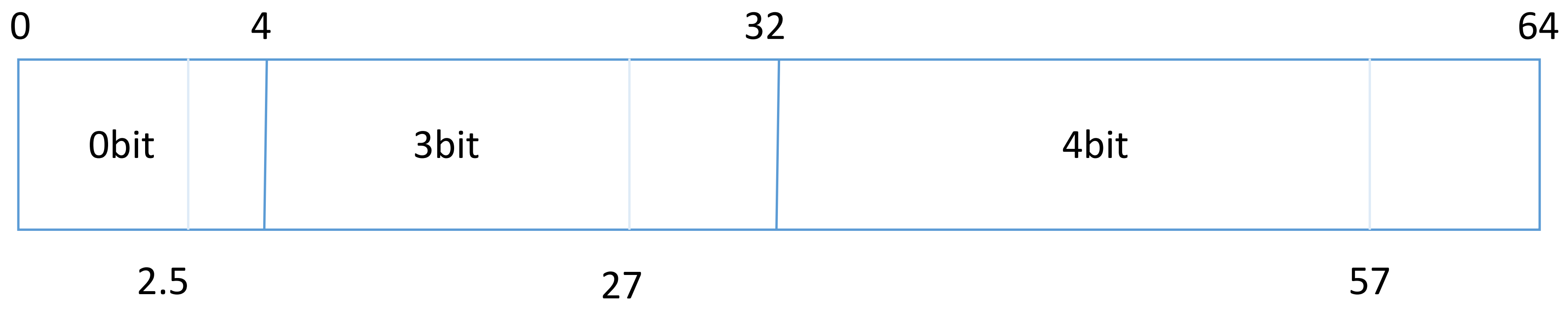
OpenJ9: Field and array element sizes

	boolean	byte	short	char	int	float	long	double
Size in field	32	32	32	32	32	32	64	64
Size in array	8	8	16	16	32	32	64	64

OpenJ9: Alignment and compressed references

- 8byte alignment – 0x0, 0x8, 0x10, 0x18, 0x20, …
  - XXXXX000 (3 least significant bits are always zero)
- 16byte alignment – 0x0, 0x10, 0x20, 0x30, 0x40 …
  - XXXX0000 (4 least significant bits are always zero)
- Using 3 bit shift (>>)
  - 35bit 8byte aligned pointer can fit in 32bit value (has 32gb range)
- Using 4 bit shift (>>)
  - 36bit 16byte aligned pointer can fit in 32bit value (has 64gb range)

OpenJ9: Compressed refs



OpenJ9: Garbage collection

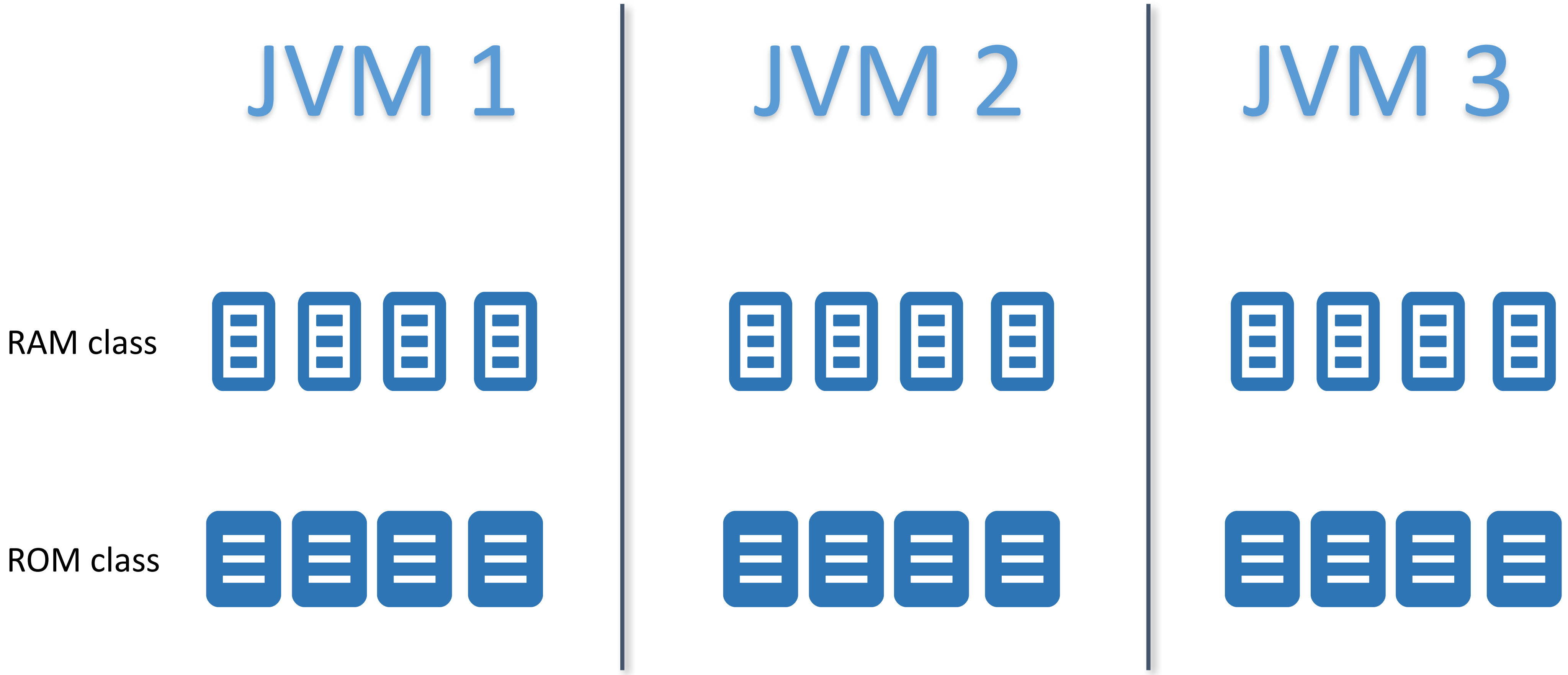
- Goals
  - Allocate space for new objects
  - Identify live objects
  - Reclaim space occupied by dead objects
- Techniques
  - Mark-sweep
  - Mark-sweep-compact
  - Copying collector

OpenJ9: GC collectors

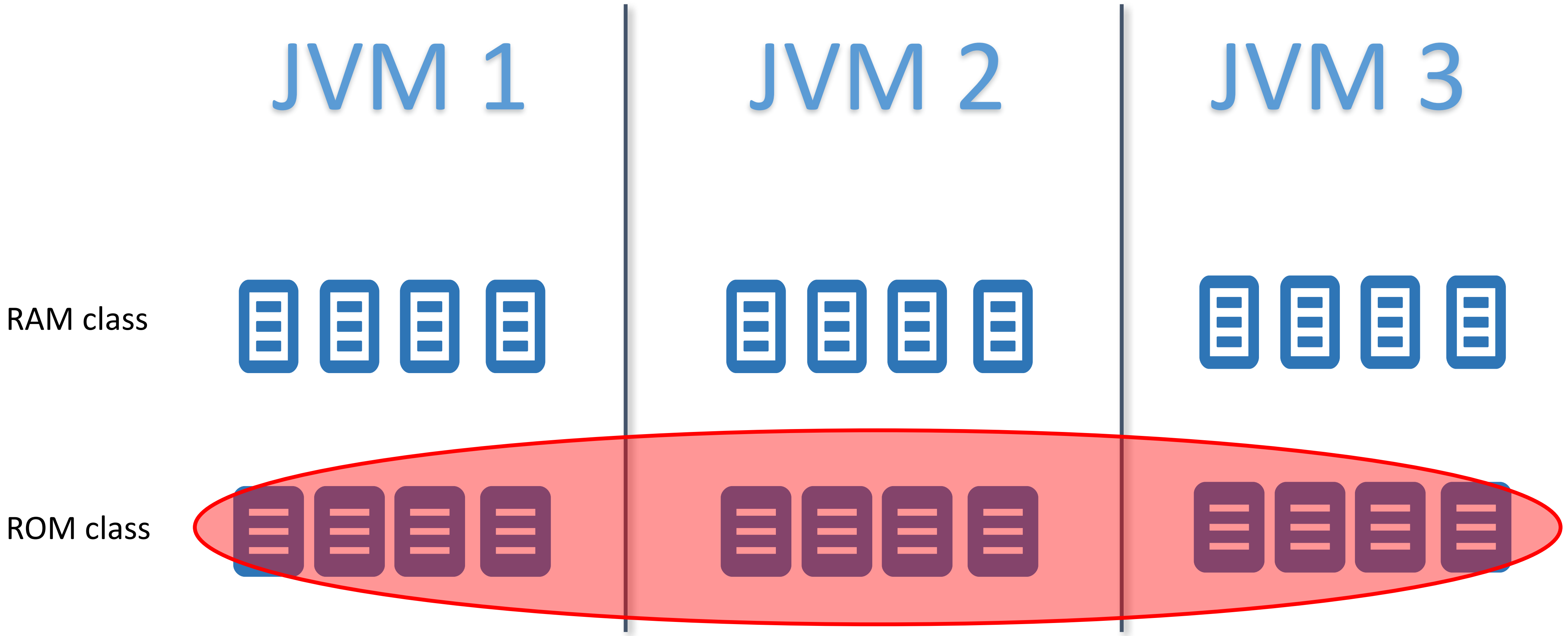
Collector	Type	Moving Objects	Concurrent	Parallel	Description
Opthruput	Tracing	Yes	No	Mark, Sweep, Compact	Classic Stop The World
Optavgpause	Tracing	Yes	Yes (Mark, Sweep)	Mark, Sweep, Compact	Concurrent
Gencon	Generational	Yes	Yes (Mark)	Mark, Sweep, Compact, Copy	Best for small and medium heaps
Balanced	Region	Yes	Yes (Mark)	Mark, Sweep, Compact, Copy	Best for large heaps
Metronome	Incremental	No	No	Mark, Sweep	Soft real-time

What are some interesting features of OpenJ9?

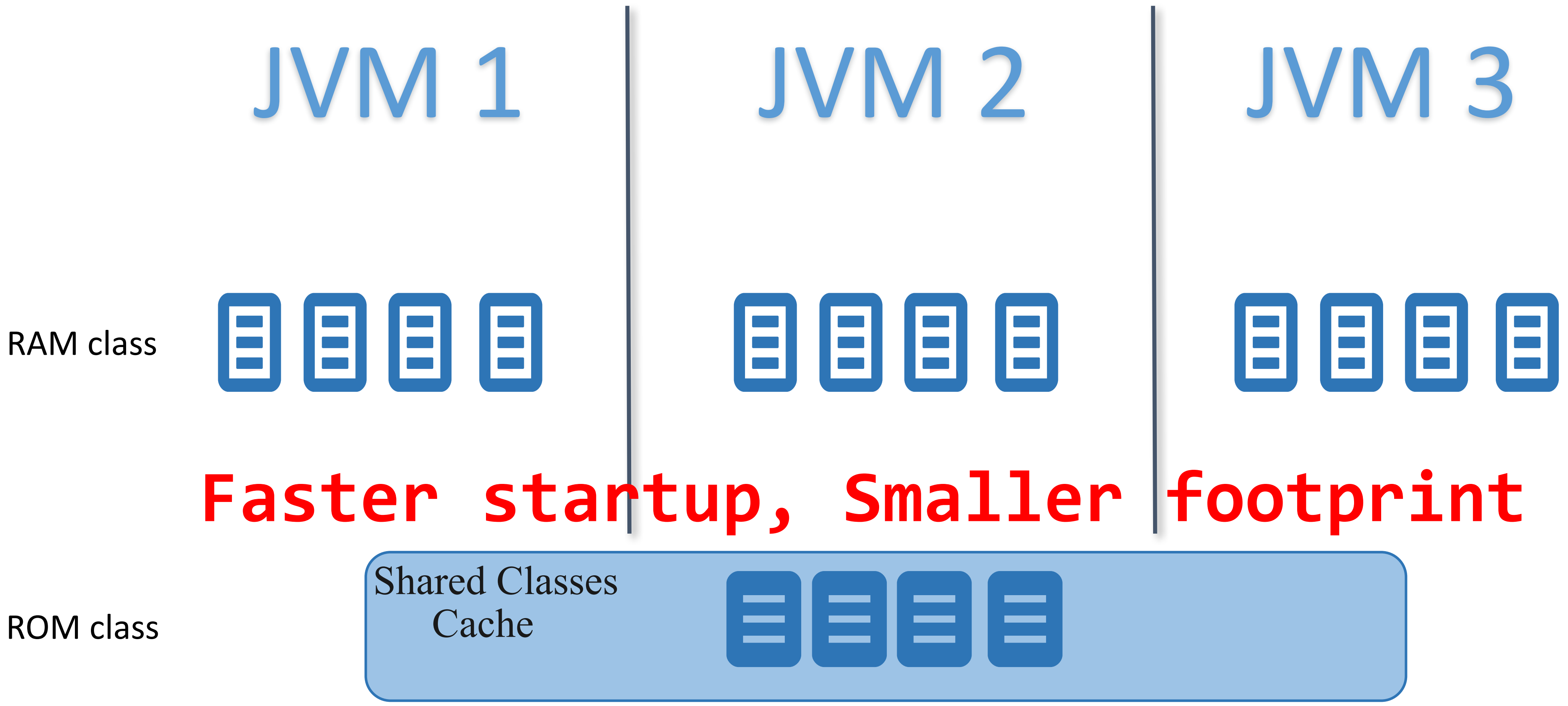
OpenJ9: Shared classes



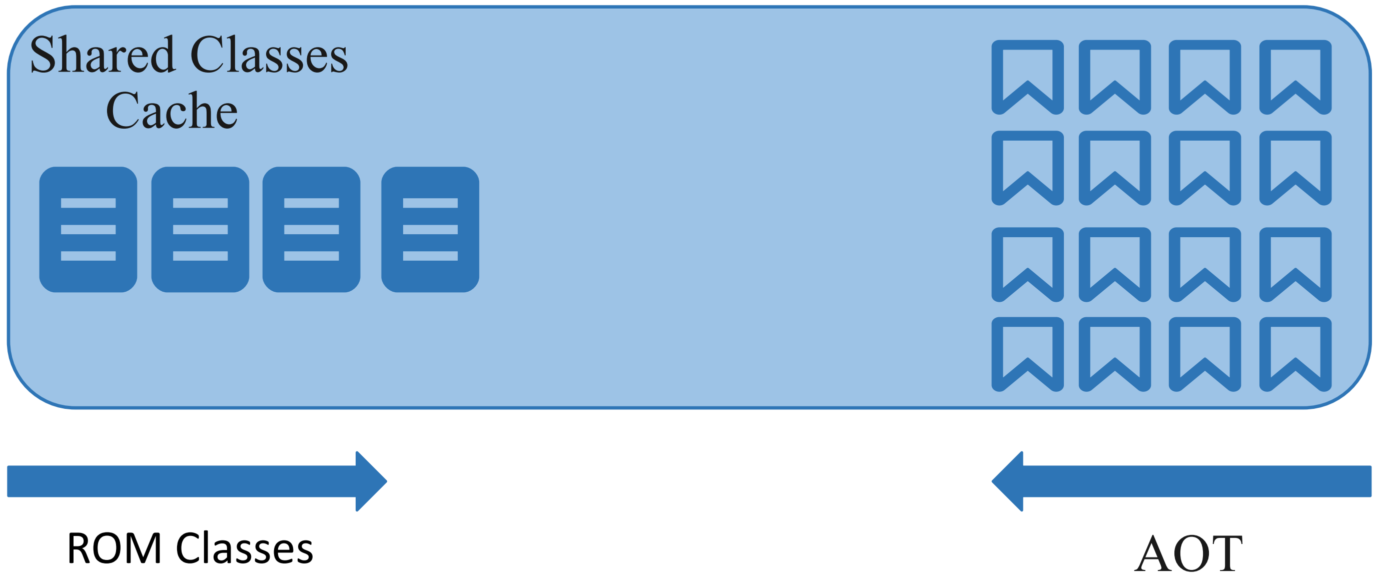
OpenJ9: Shared classes



OpenJ9: Shared classes



OpenJ9: Dynamic AOT

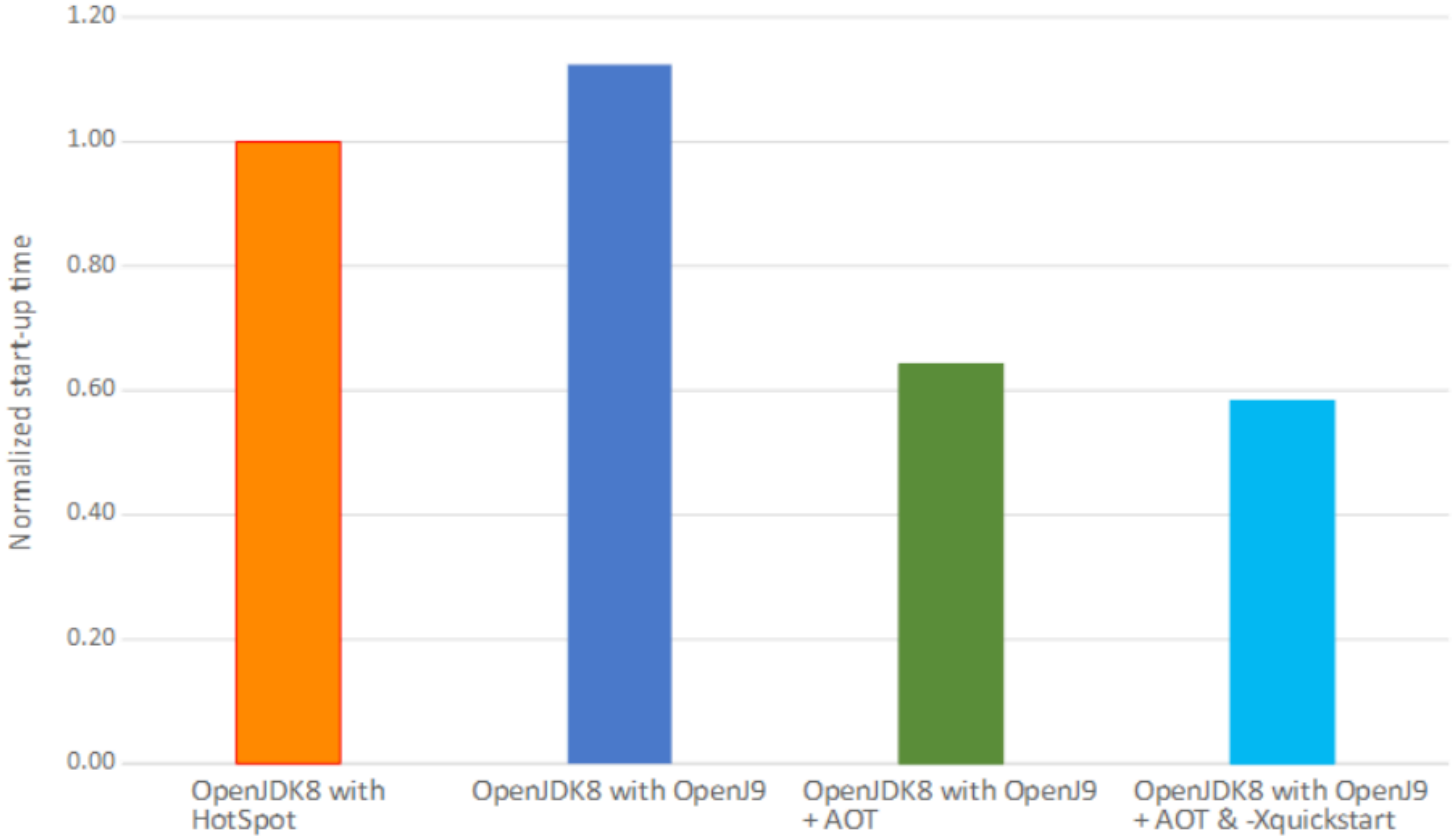


OpenJ9: using shared class cache

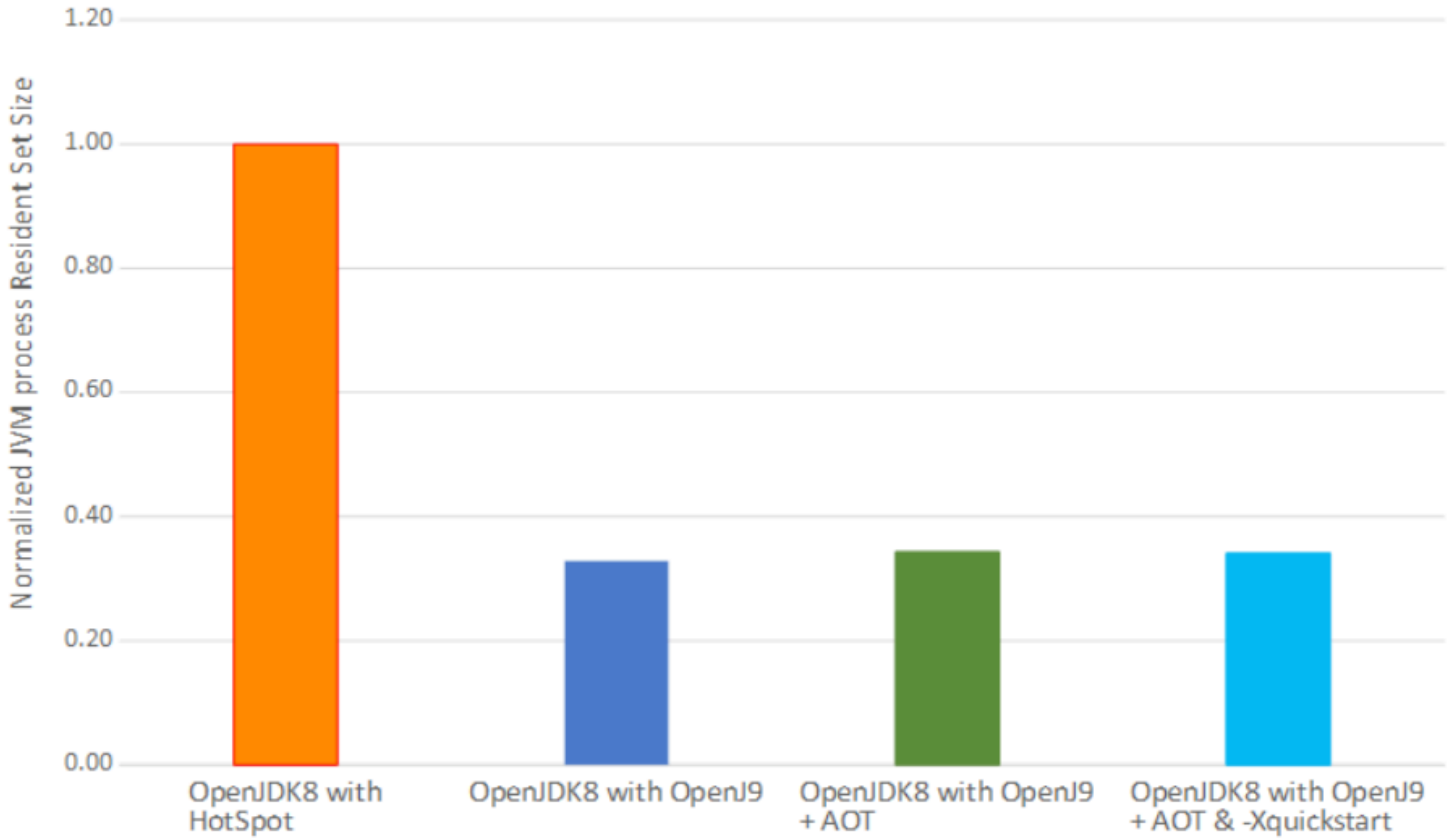
```
java –Xshareclasses …
```

```
java –Xshareclasses –Xscmx32m …
```

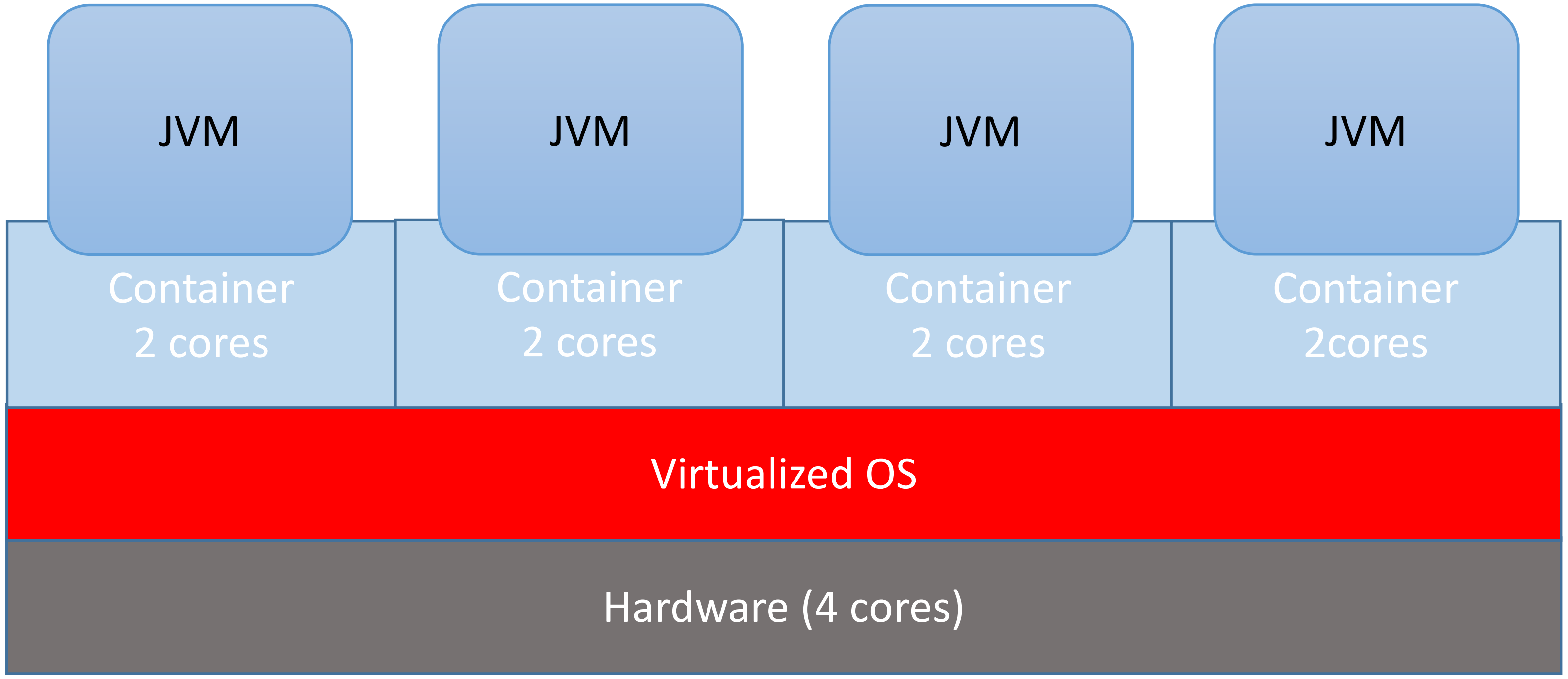
Startup time (with -Xmx1g)



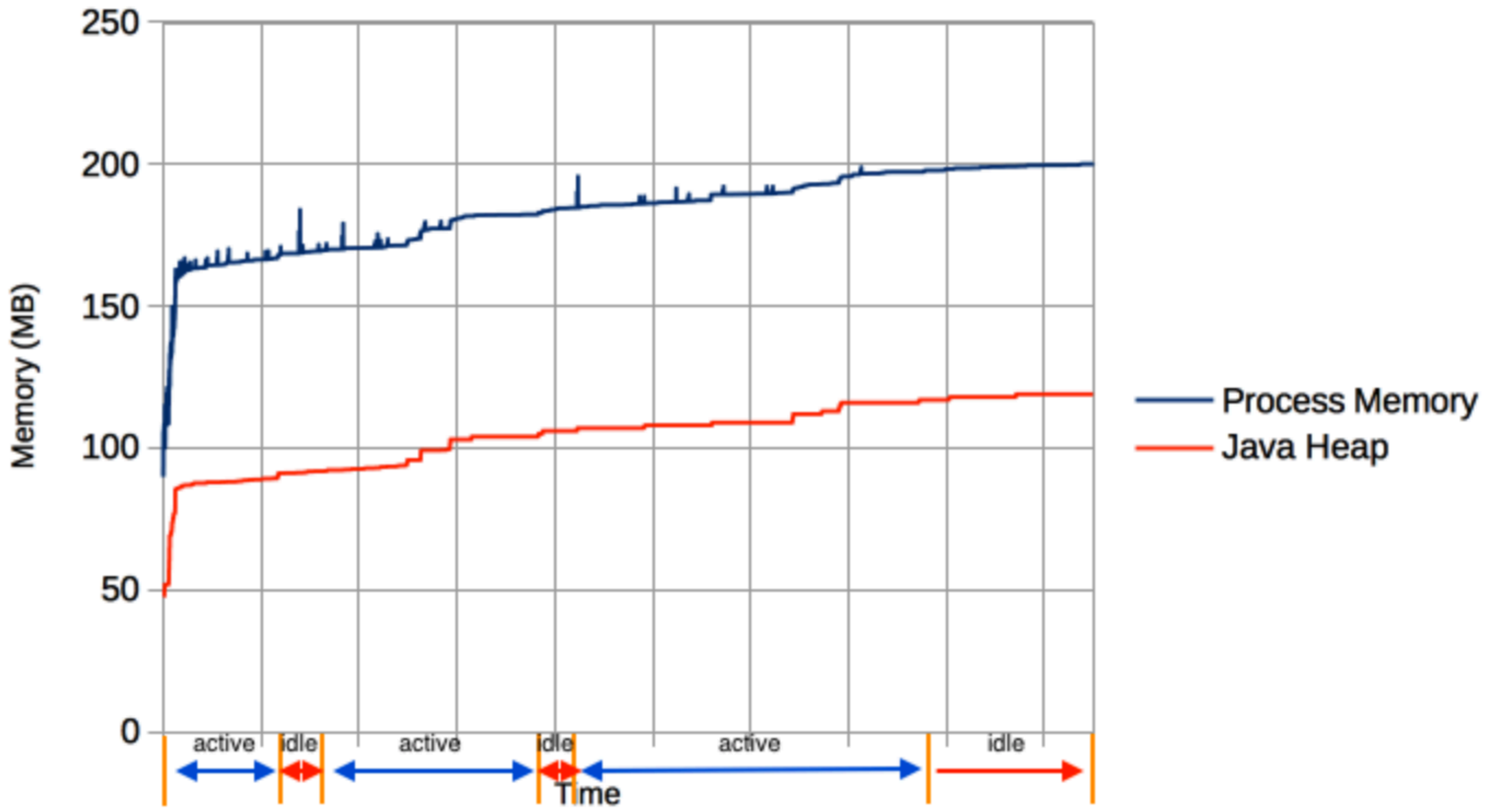
Footprint size after start-up (with -Xmx1g)



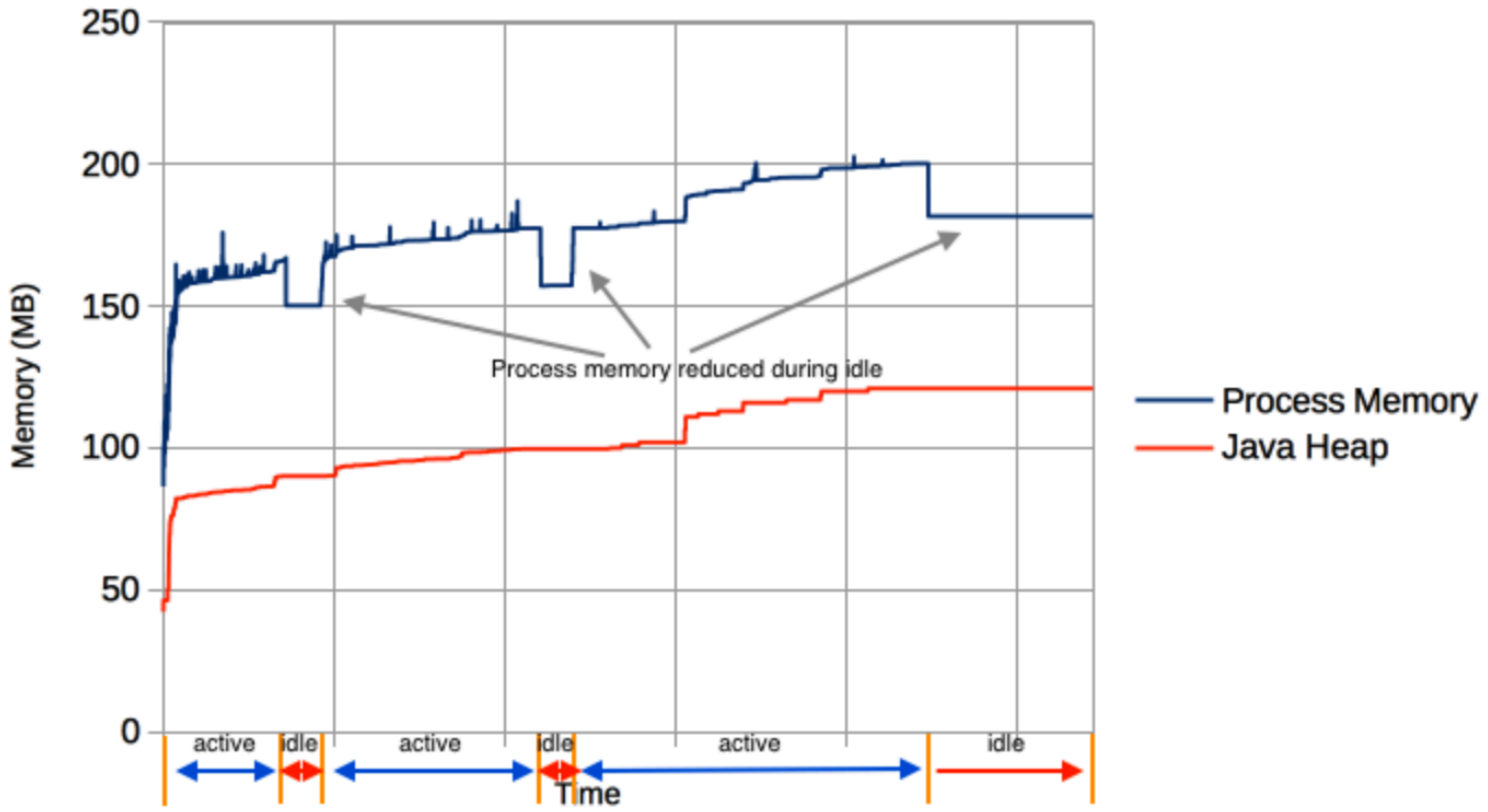
OpenJ9: -Xtune:virtualized



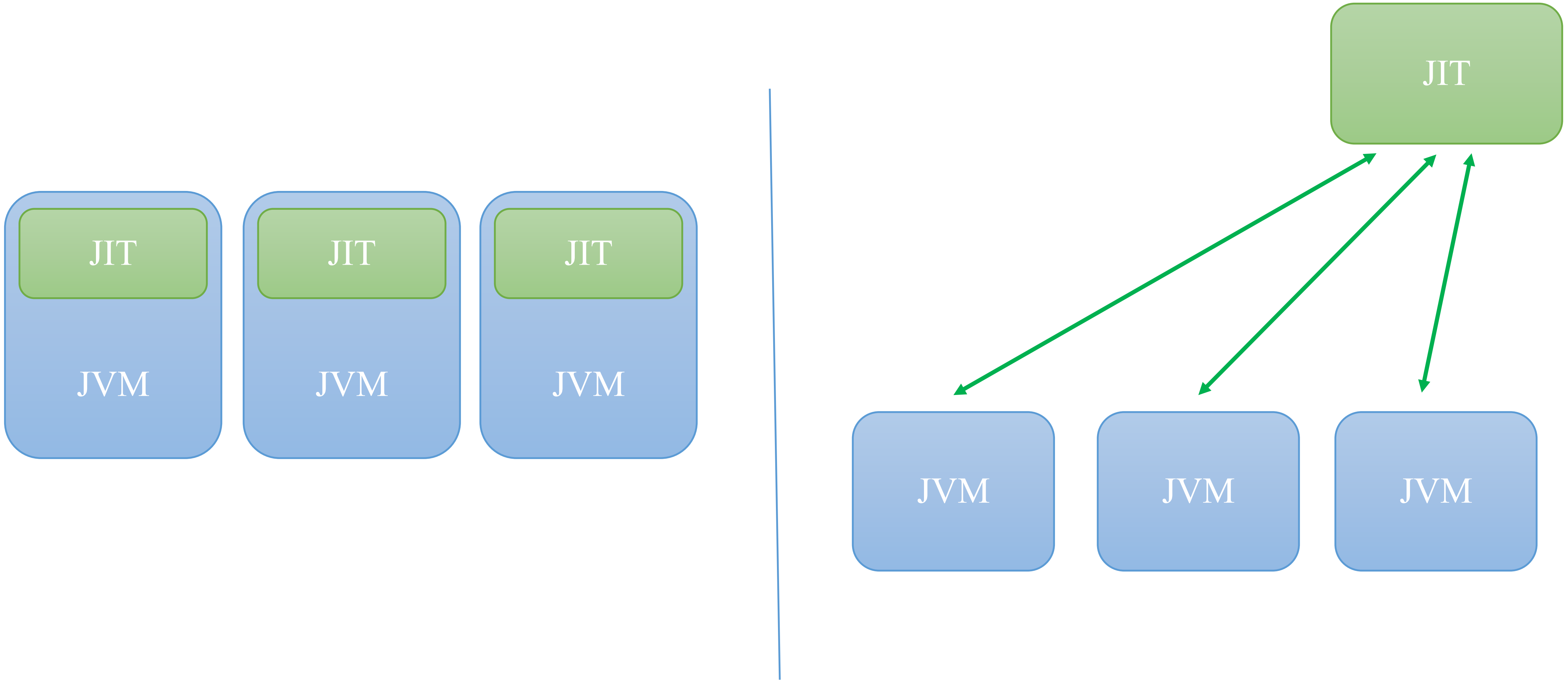
OpenJ9: Idle detection



OpenJ9: -XX:+IdleTuningGcOnIdleoption



OpenJ9: JIT as a service



Summary

- OpenJ9 is new JVM that runs with OpenJDK
- Many features that make it suitable for cloud environments
- Join us at https://github.com/eclipse/openj9

Questions???