Assisted warmup with the Zing JVM

Iván Krȳlov
@JohnWings

AZUL SYSTEMS®

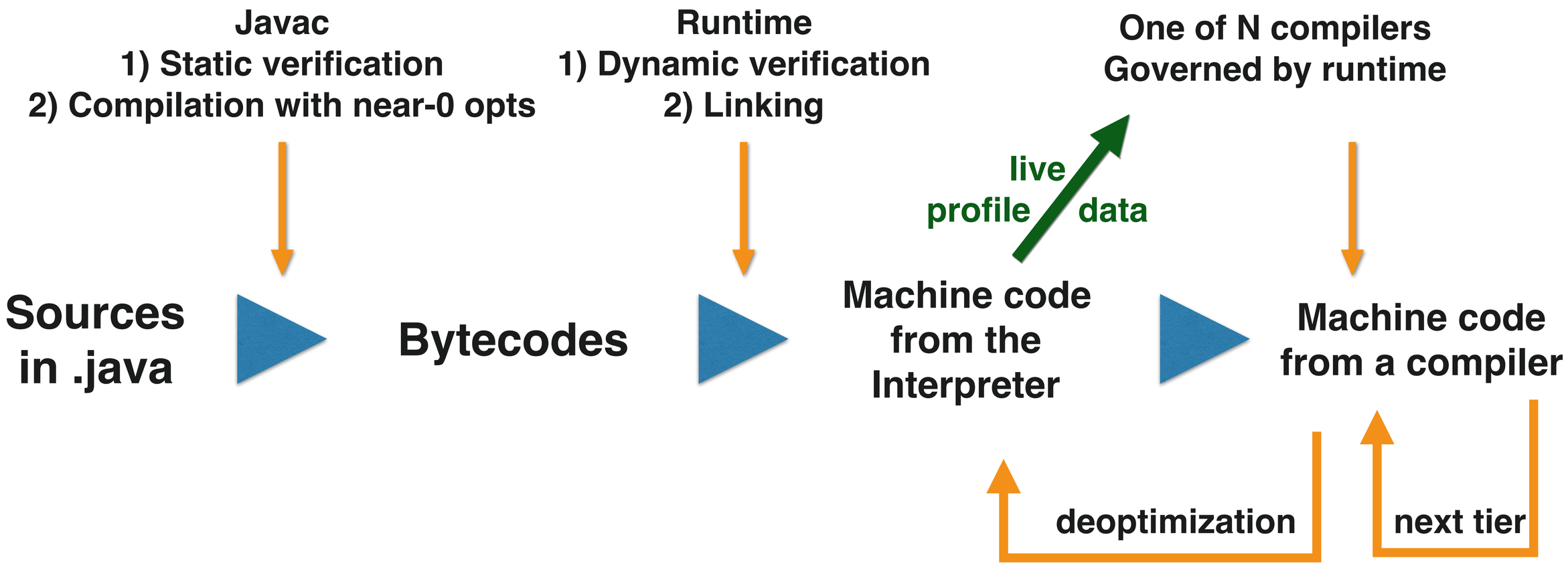
Assisted warmup with the Zing JVM

- Overview of 3 technologies
  - Falcon compiler
  - ReadyNow & Compile Stashing
- Challenges *(largely universal to all AOTs)*
  - Identification of classloaders & classes
  - Timing of class' initializers
  - Consistency of class generators

Zing

- **Zing:** A better JVM for the your servers
  - Consistent performance - not just fast, *always* fast
  - Eliminate GC as a concern for large apps
  - Very wide operating range
    - From human-sensitive app responsiveness to low-latency trading
    - From microservices to huge in-memory apps
  - Eliminates an entire class of engineering workarounds common in Java
  - Home for Falcon, ReadyNow, Compile Stashing and other technologies

Code pipeline in JVM



AZUL SYSTEMS®

Falcon

A new LLVM-based JIT for JVM-languages in Zing VM

AZUL SYSTEMS®

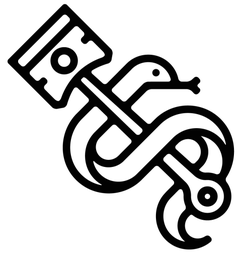
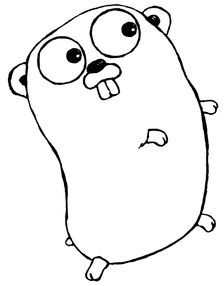
“The LLVM Project is a collection of modular and reusable compiler and toolchain technologies”

– *llvm.org*

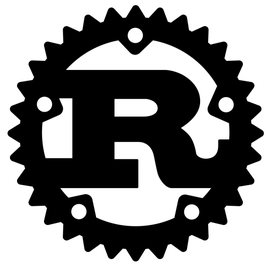
AZUL SYSTEMS®

Where LLVM is used?

- C/C++/Objective C
- Swift
- Haskell
- Rust
- …




Who makes LLVM?

Google

AMD

SONY

Microsoft

IBM

COMPUTER ENTERTAINMENT ®

SAMSUNG

AZUL SYSTEMS®

ARM

Linaro

Imagination

NVIDIA®

intel®

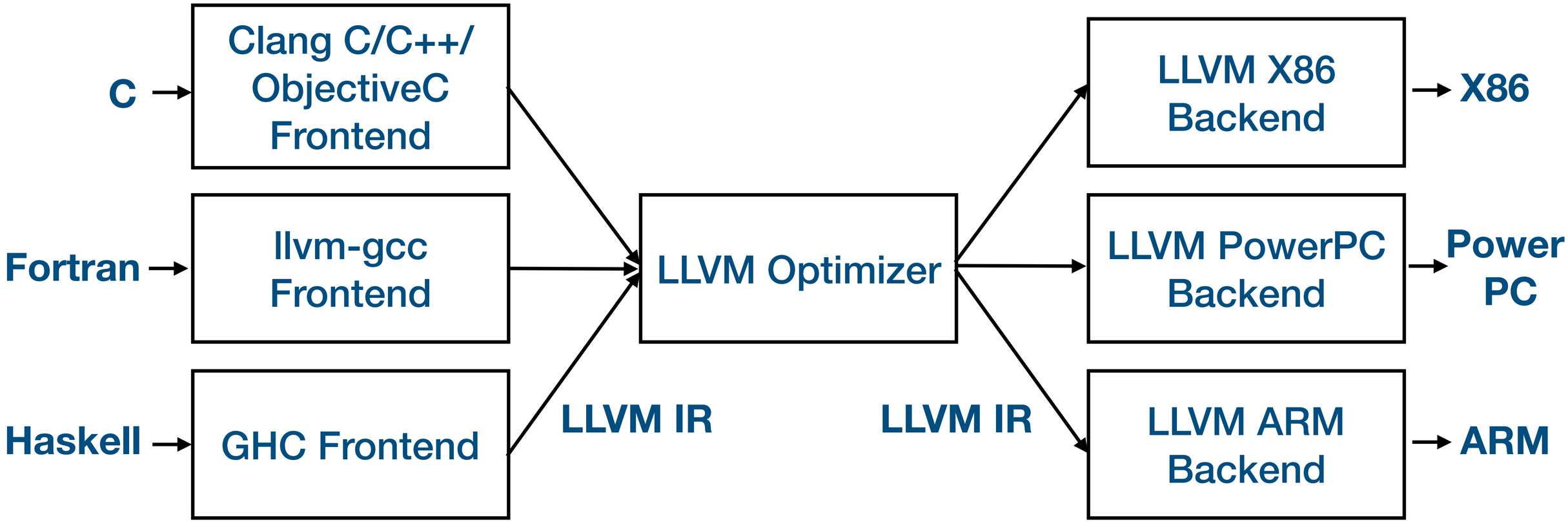
QUALCOMM®

CRAY

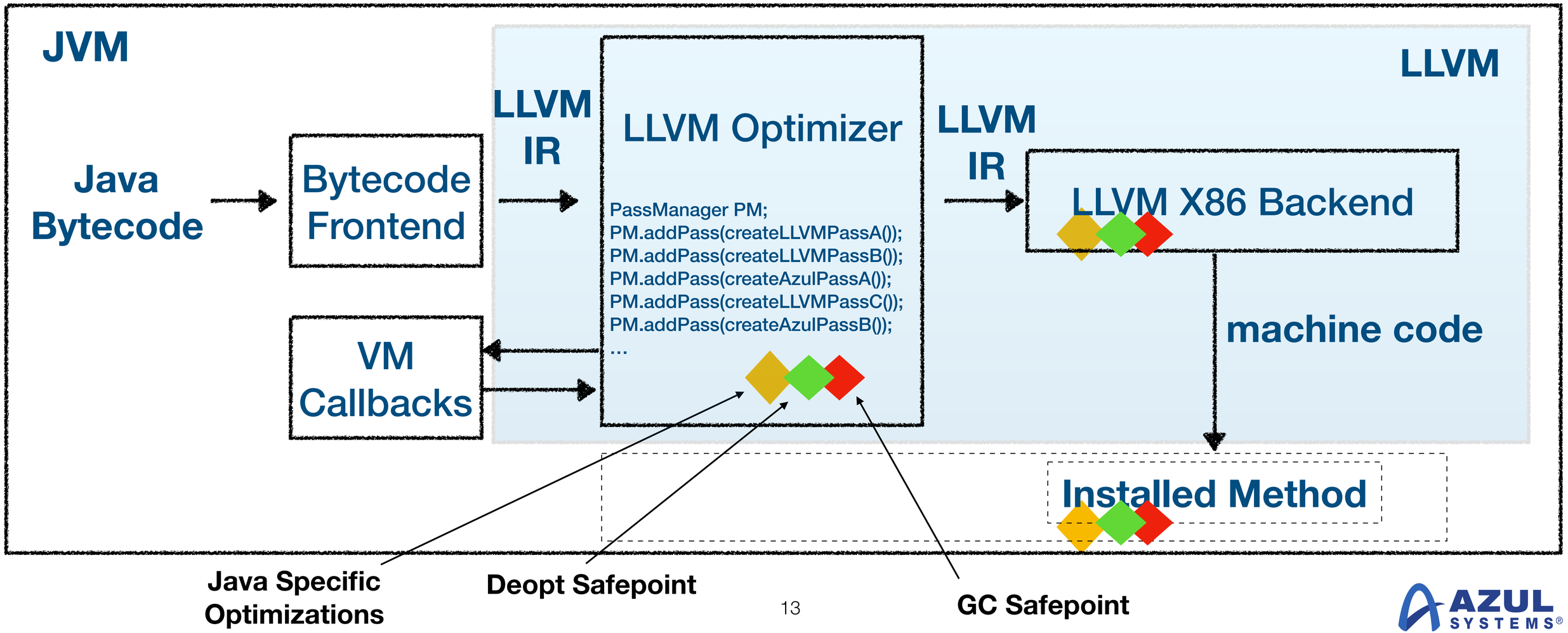
More than 500 developers

AZUL SYSTEMS®

A typical llvm-based compiler

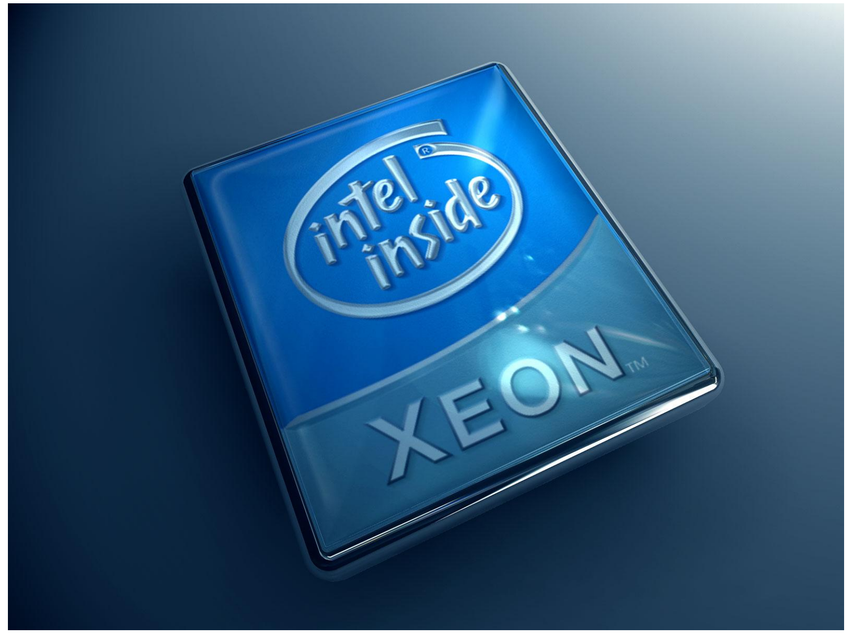


New concepts for LLVM



AZUL SYSTEMS®

Support for new CPUs

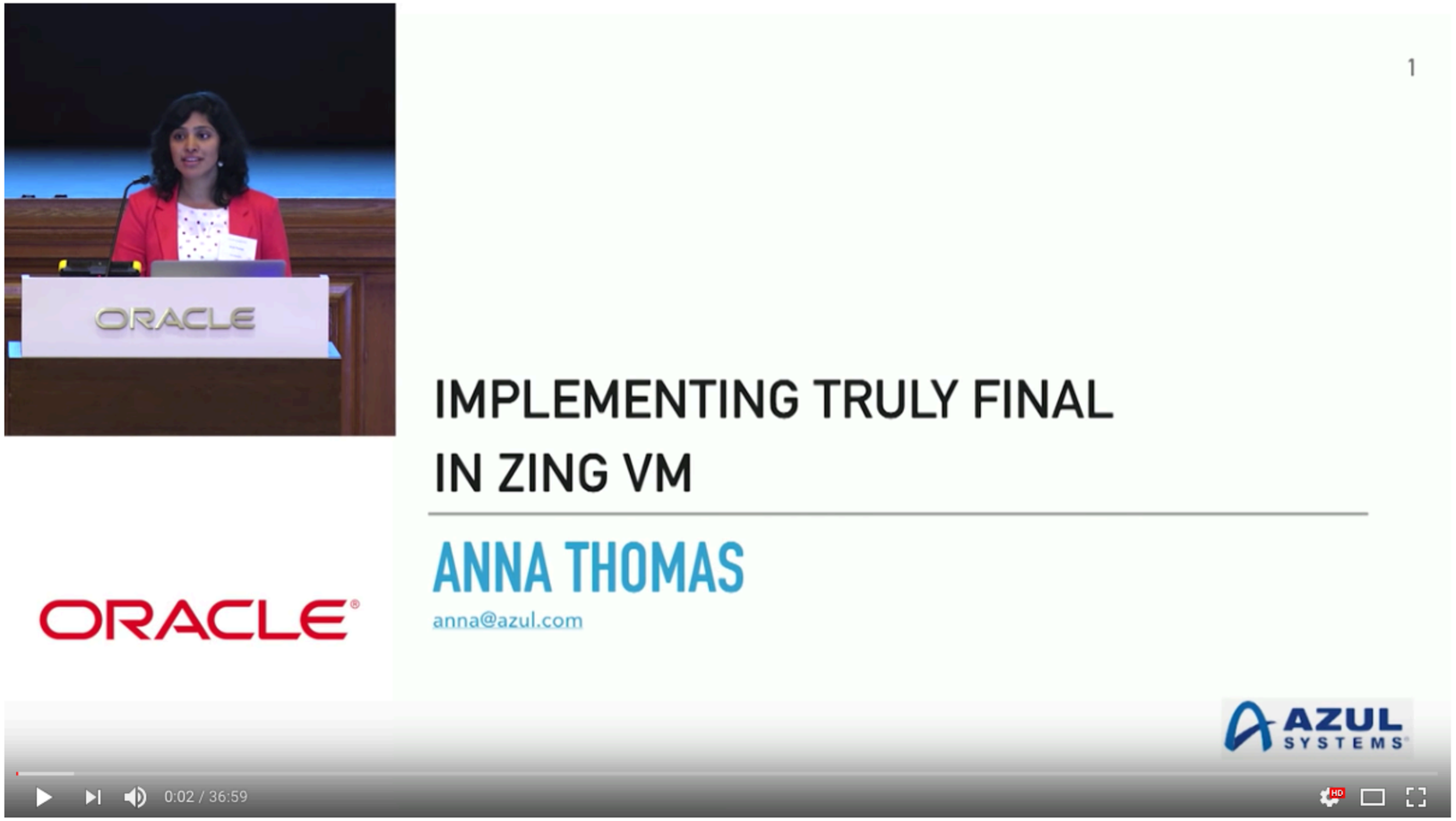


Thanks to Intel’s hard work on LLVMs backed Falcon emits AVX-512 instructions from the day those CPUs are on the market

AZUL SYSTEMS®

Faster feature development

A. Thomas, JVMLS 2018, https://www.youtube.com/watch?v=2HfnaXND7-M



AZUL SYSTEMS®

Simple things done simpler

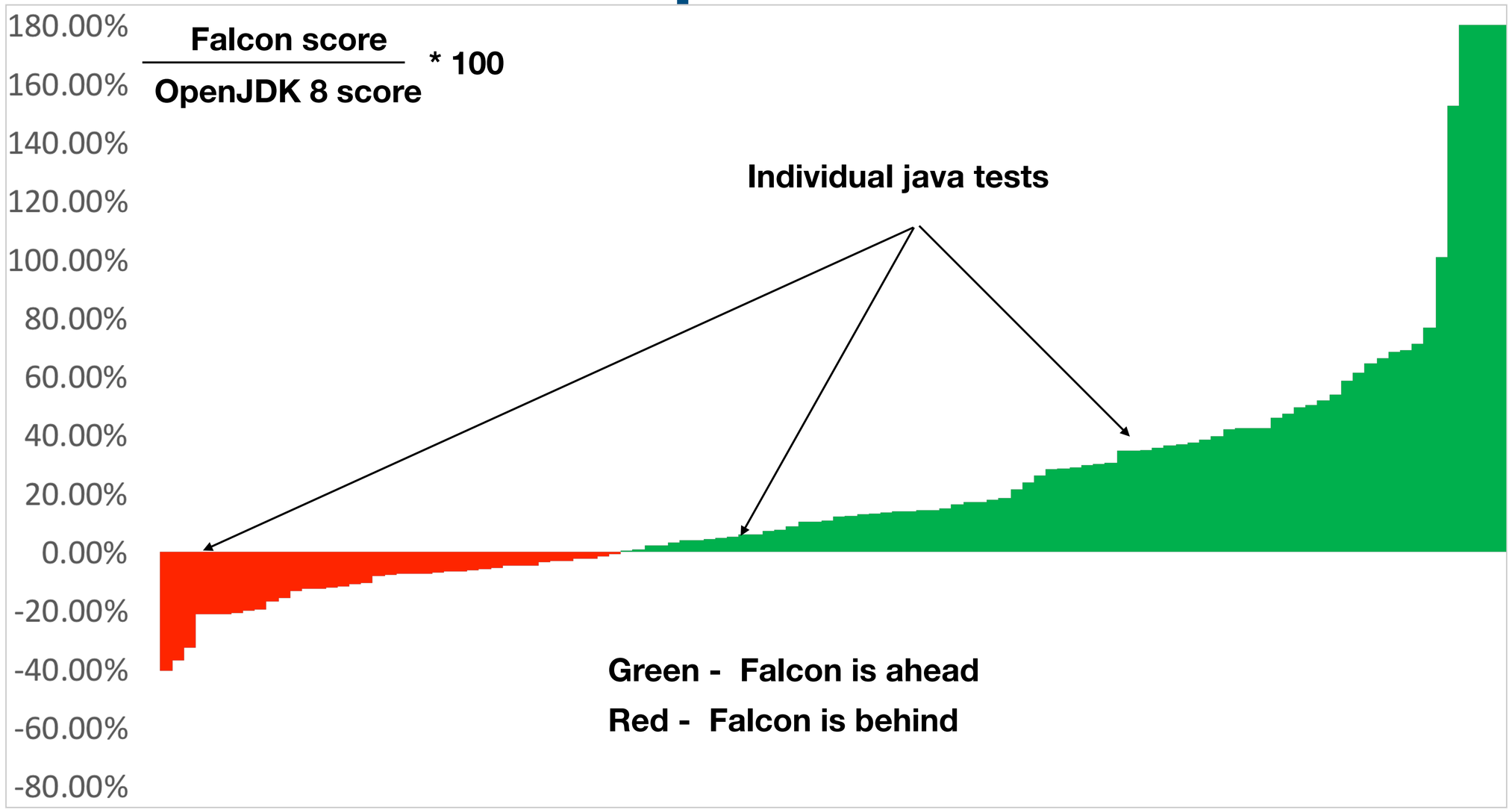
- Developing new intrinsics is super-easy
- Example for onSpinWait intrinsic written in LLVM IR

```
declare void @llvm.x86.sse2.pause() nounwind

;; intrinsic for java.lang.Thread.onSpinWait()
define zing void @_onSpinWait_performance_Hints() nounwind alwaysinline "azul-inlining-candidate" {
entry:
call void @llvm.x86.sse2.pause() nounwind
ret void
}
```

On Falcon performance

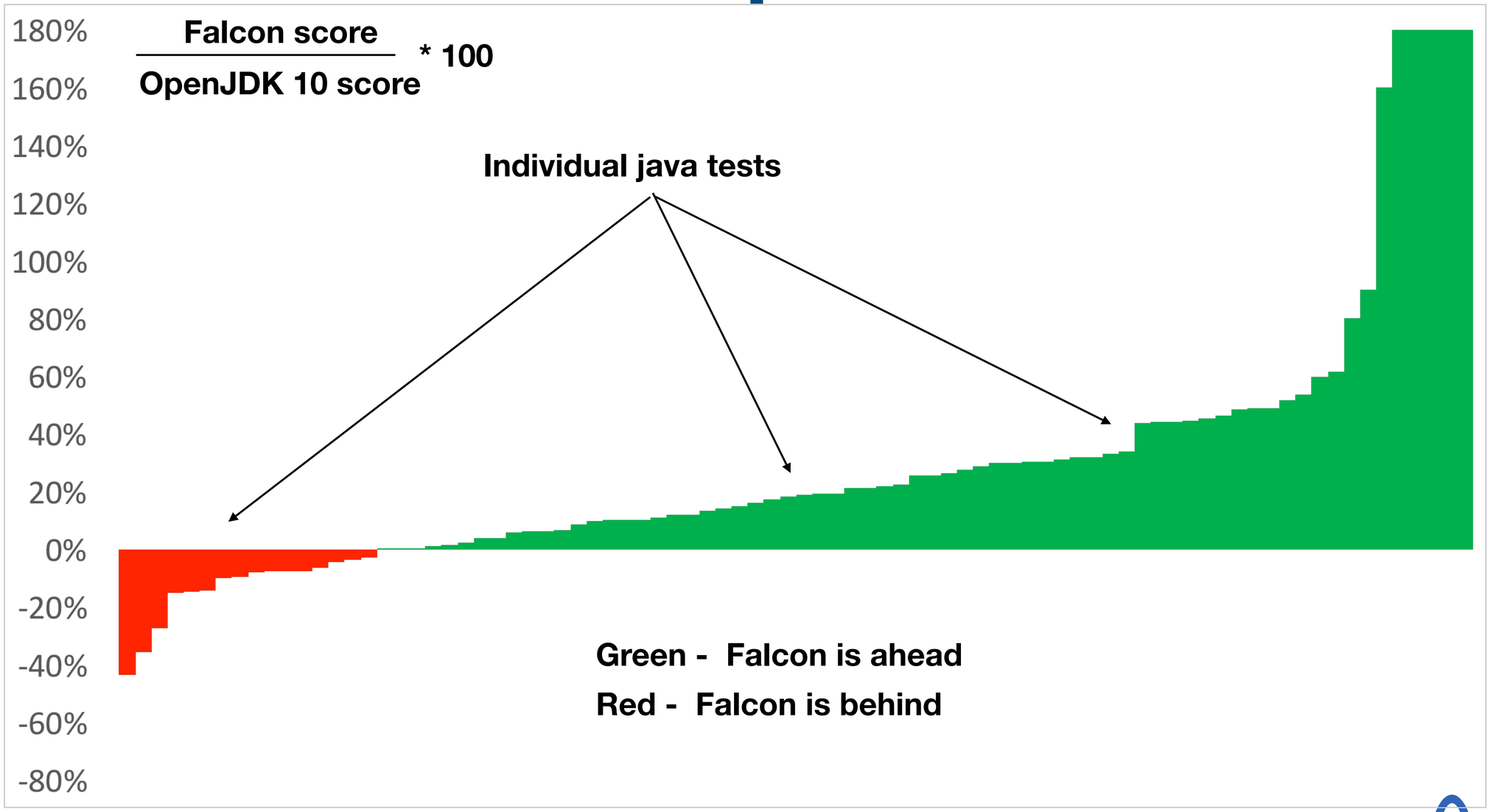
- Beats our own C2 across the board
- Looks good against other VMs

Falcon vs OpenJDK 8u171



AZUL SYSTEMS®

Falcon vs OpenJDK 10



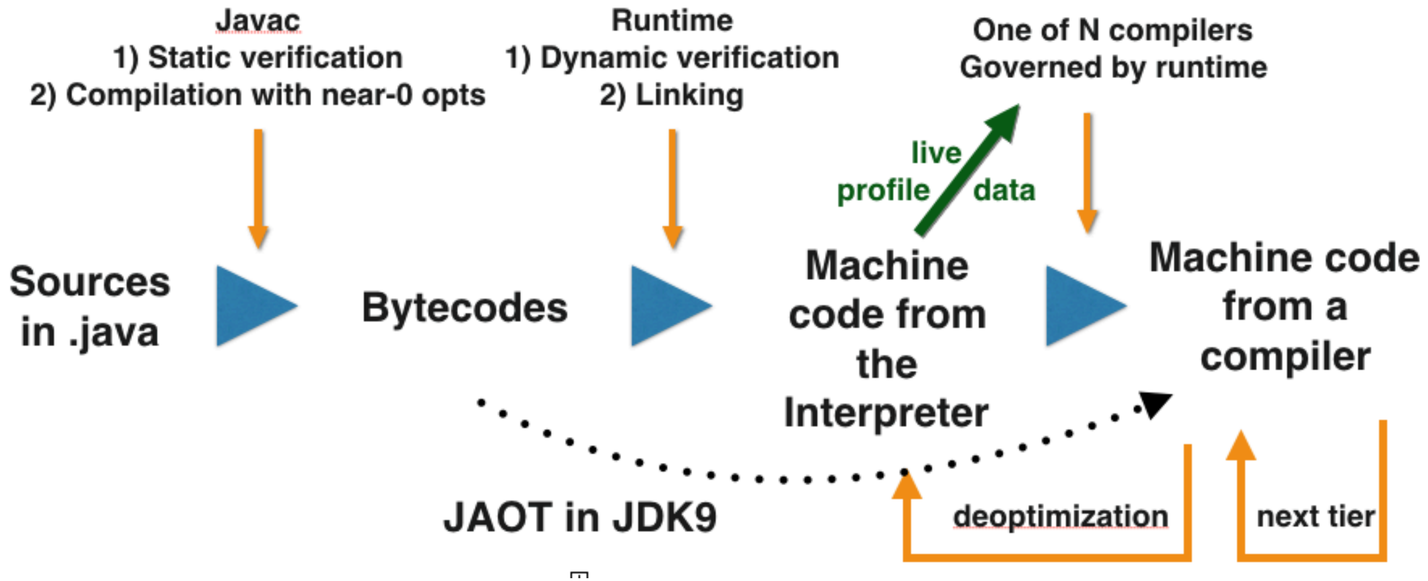
AZUL SYSTEMS®

ReadyNow

- To reduces warmup times
- To avoid mistakes of JITs speculative optimizations
- Achieved by feeding information from pervious run
- Very simple to use: -XX:ProfileLogOut=yourapp.log and -XX:ProfileLogIn=yourapp.log

Terminology: profile

- Method's Live profile - memory structure inside of JVM filled during warm up and updated as needed



AZUL SYSTEMS®

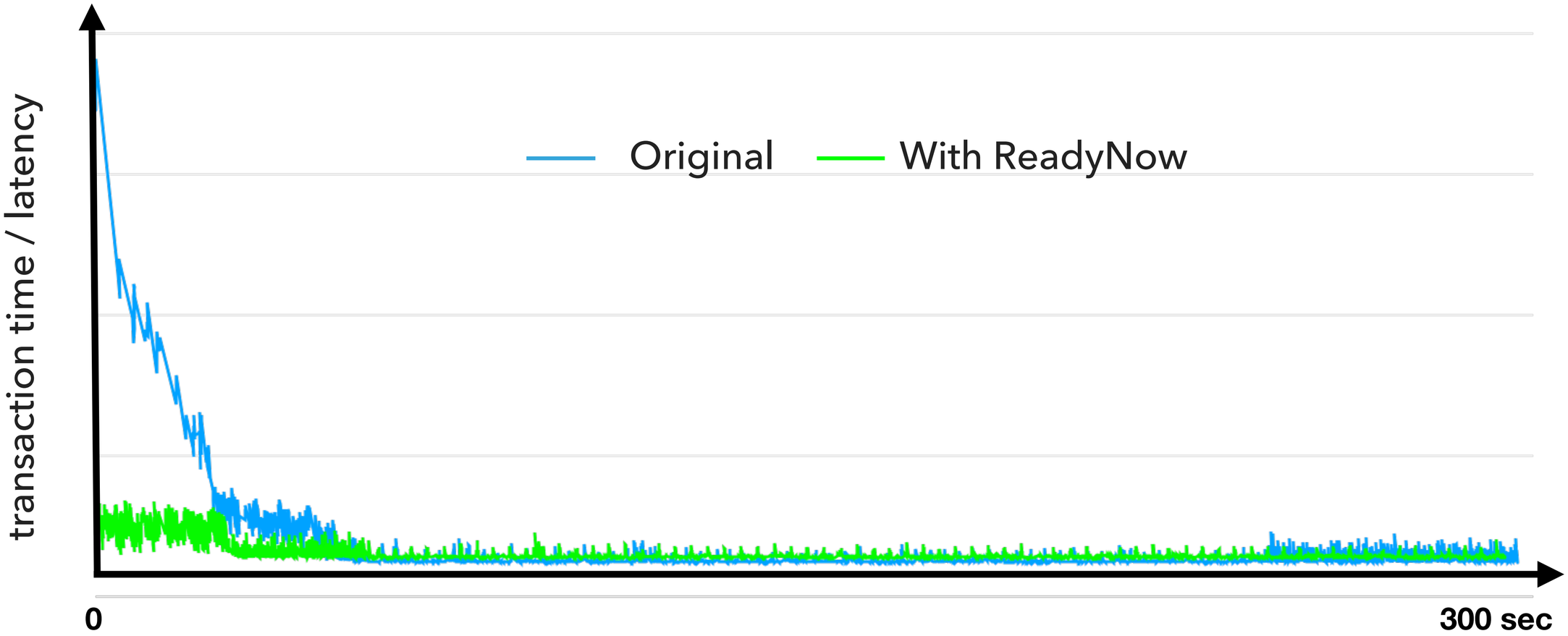
Terminology: profile

- Method's Live profile - memory structure inside of JVM filled during warm up and updated as needed
- Method's Persisted profile - the one externally saved to a "file" during one run and used during the other
  - Can be in per-method context or full-application context

On ReadyNow performance

AZUL SYSTEMS®

ReadyNow

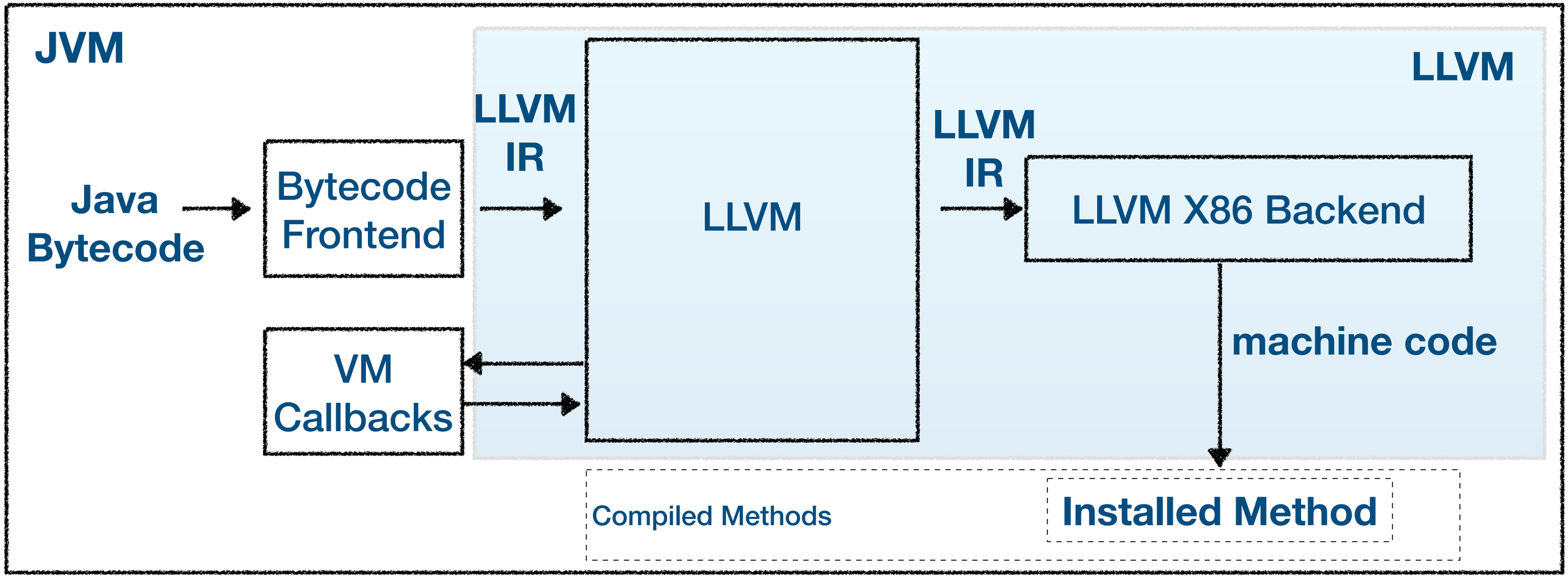


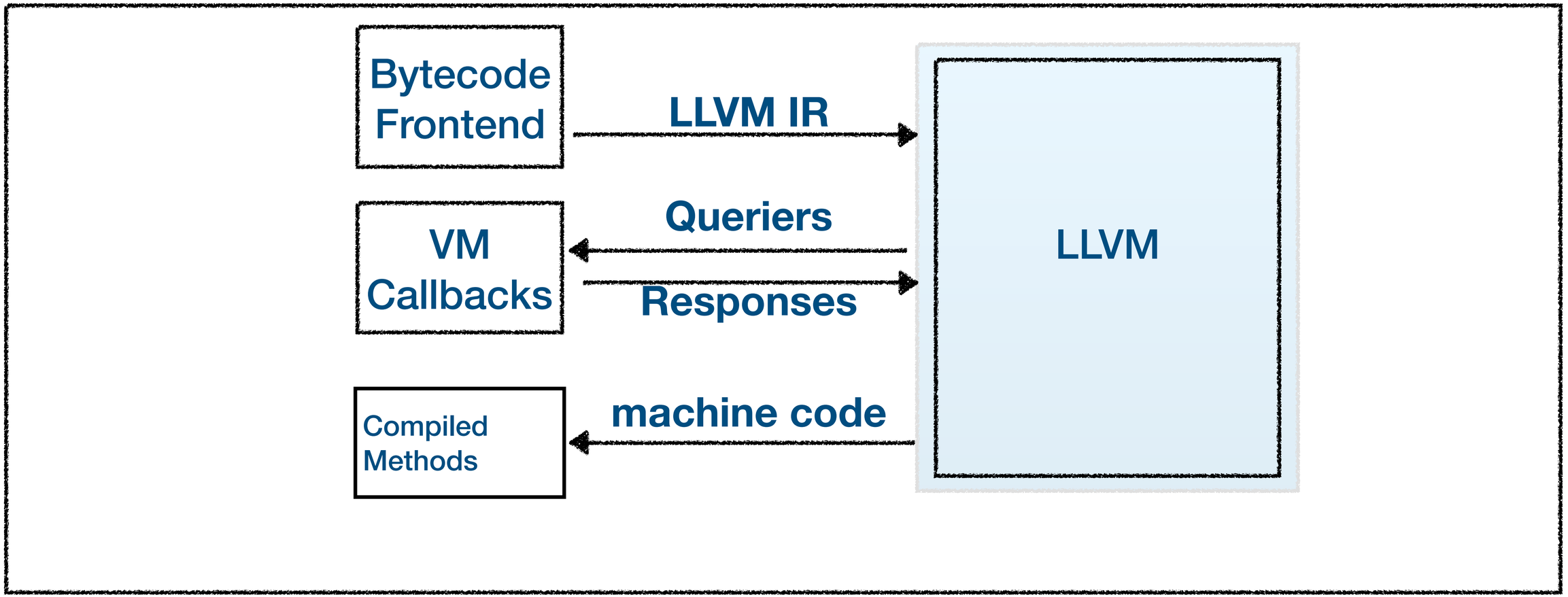
AZUL SYSTEMS®

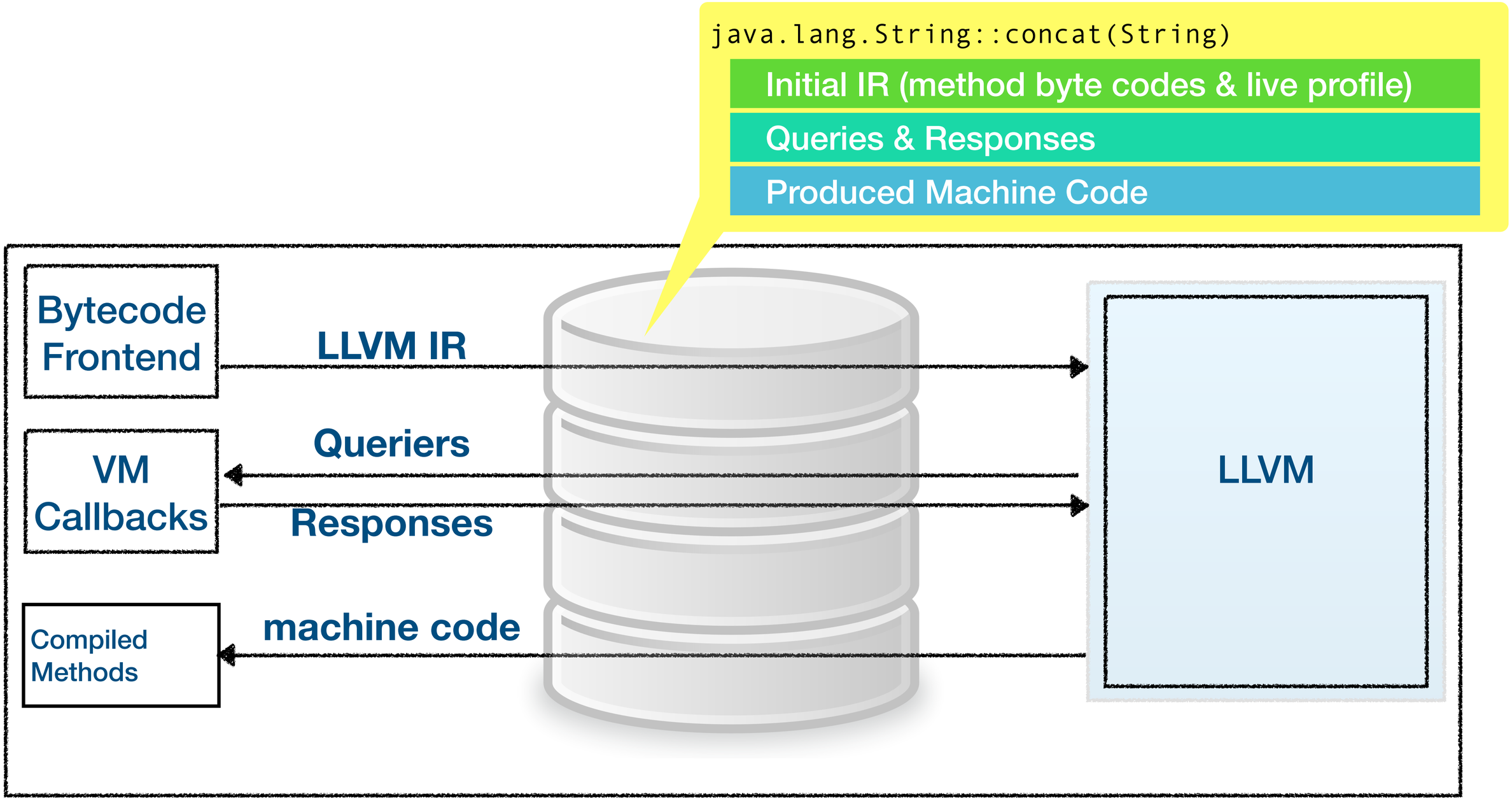
Compile Stashing

- Reuse **top-tier** compilation
- Applied to JDK and user code
- Challenges are the same as for JAOT

VM Callbacks API

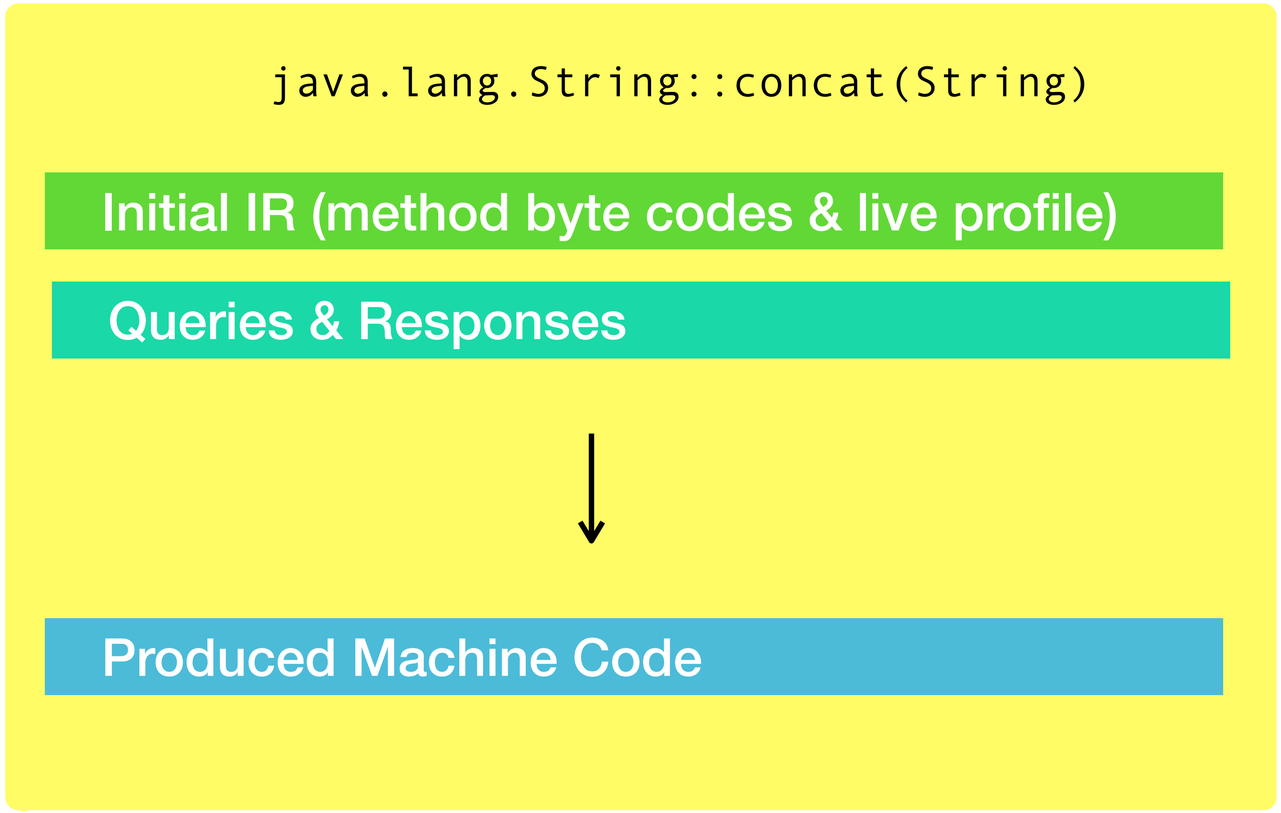


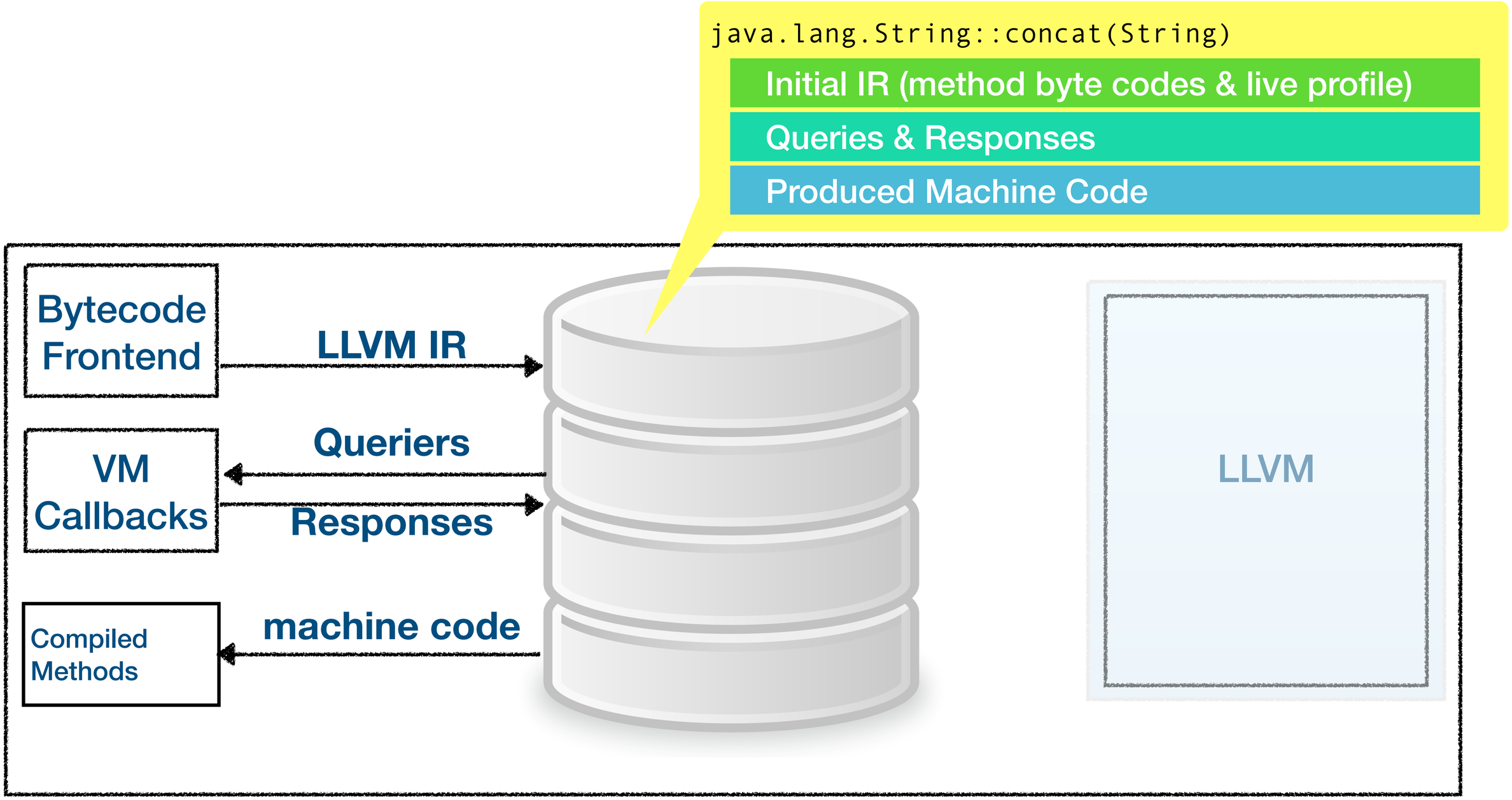




AZUL SYSTEMS®

Determinism in compilation





AZUL SYSTEMS®

More on Falcon

A. Pilipenko, I. Krylov, JFocus 2018, https://www.youtube.com/watch?v=XovFnMw_eGk

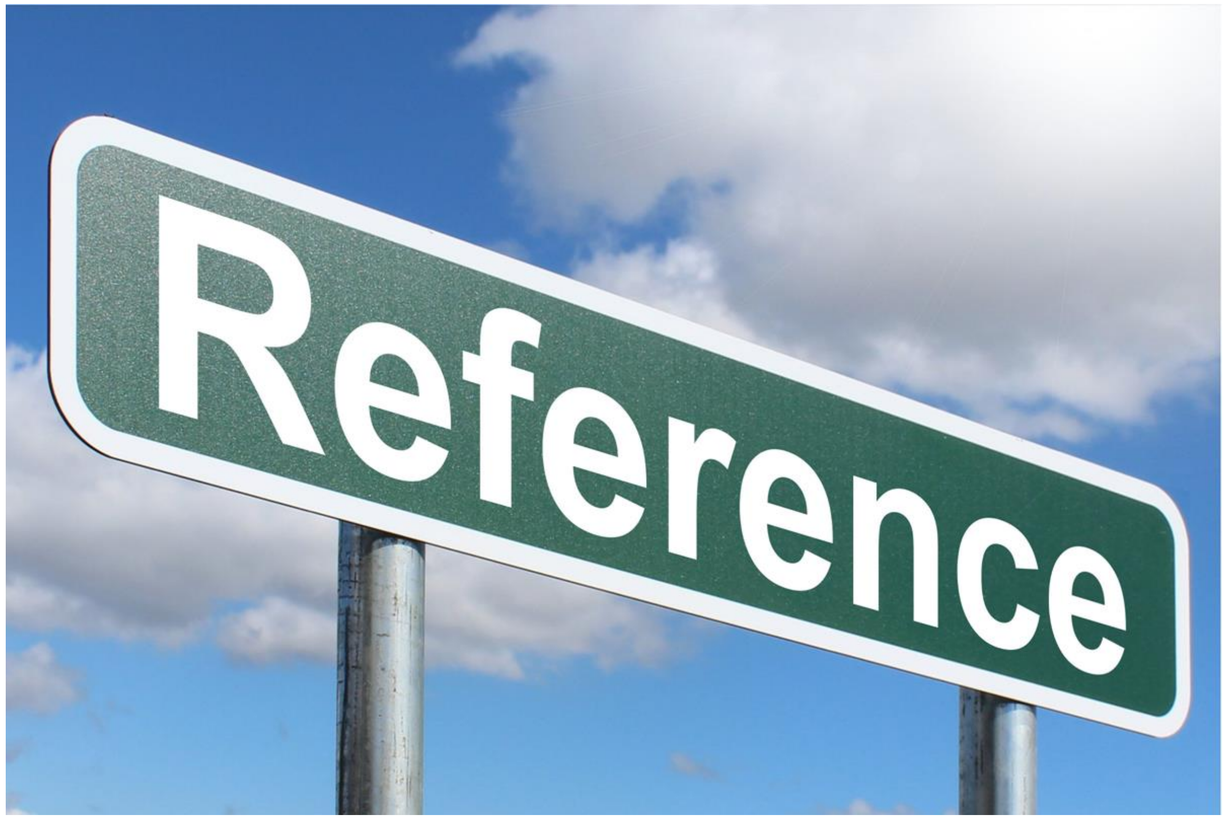
AZUL SYSTEMS®

Challenges

Challenges

- Class Referencing
- Profile Normalization
- Class' and method' instrumentation
- Classloader identity
- Class generators
- Static Initializers

Class referencing



AZUL SYSTEMS®

Referencing a class in IR

```
class Merchandise {
    Map<Integer, String> merchandise;

    int getLength() {
        return merchandise.size();
    }
}
```

Devirtualization

```
// IR written as pseudo C++ code

int getLength() {
 if (merchandise instanceof HashMap) {
    return HashMap::size();
 } else {
   // Continue in interpreter & perhaps recompile
 }
}
```

Classes as pointers

```
// IR written as pseudo C++ code

int getLength() {
 if (merchandise instanceof 0x4d3v7ef0) {
    return 0x4d3v7ef0::size();
 } else {
   // Deoptimize and continue in interpreter
 }
}
```

Zing uses persistent klass ids

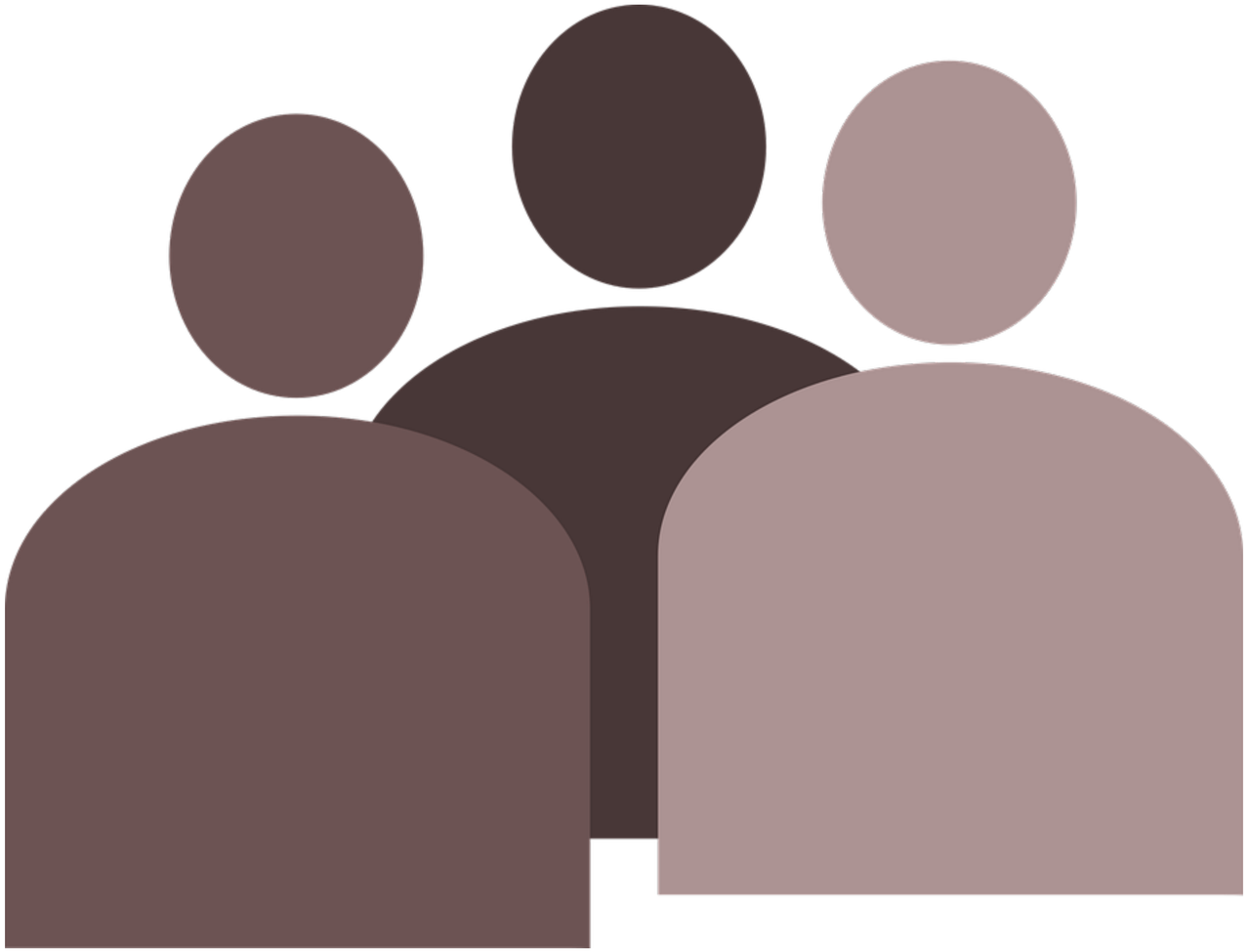
```
// IR written as pseudo C++ code

int getLength() {
 if (merchandise instanceof 336) {
    return 336::size();
 } else {
   // Deoptimize and continue in interpreter
 }
}
```

Klass id	Klass address
335	0x4d3e4d20
336	0x4d3e4df0
337	0x4d3ea03c
338	0x447c21d0

Challenges

- ✓ Class Referencing
- Profile Normalization
- Class’ and method’ instrumentation
- Classloader identity
- Class generators
- Static Initializers

Profile Normalization

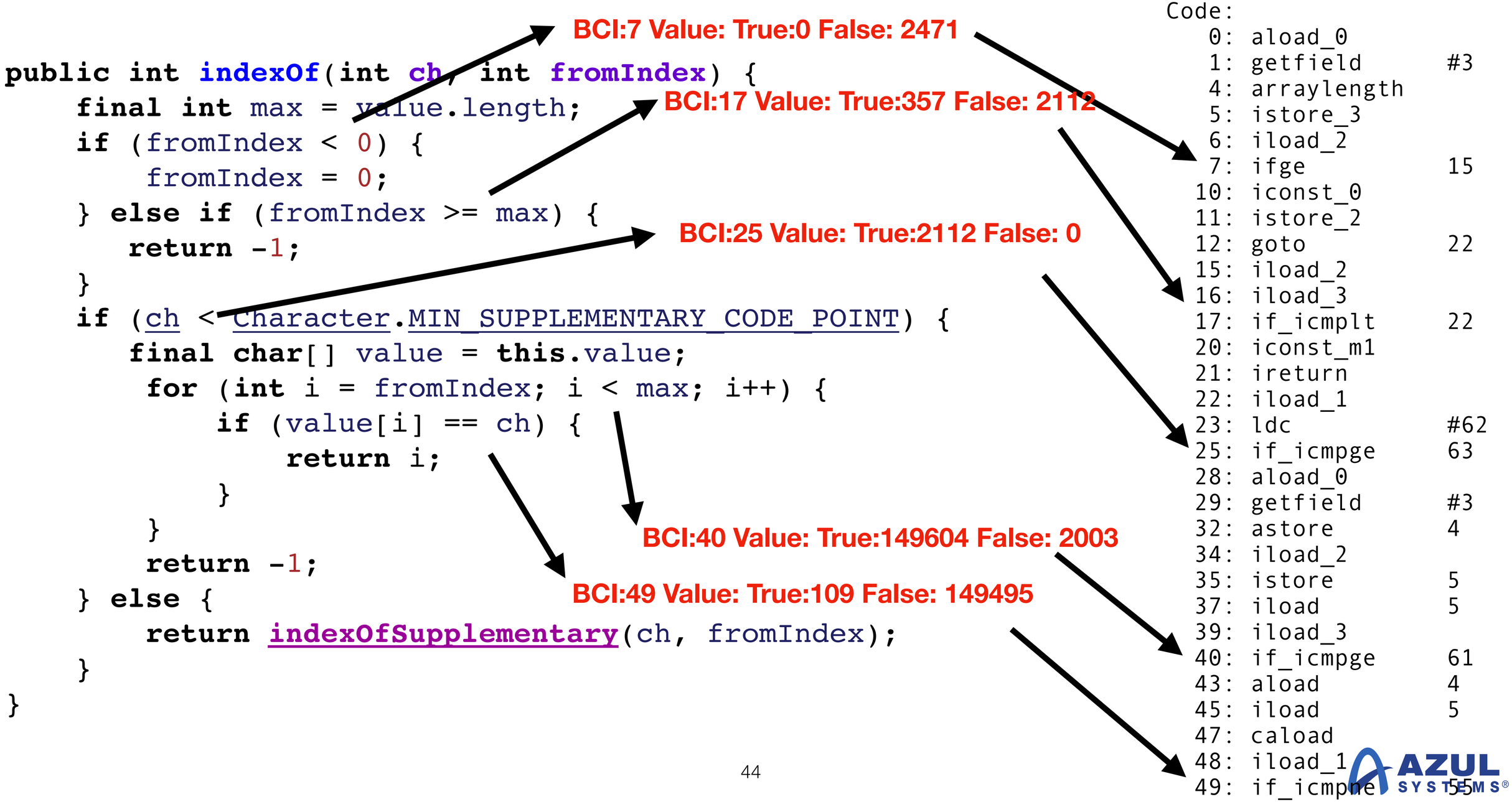


AZUL SYSTEMS®

j.l.String::indexOf

```
public int indexOf(int ch, int fromIndex) {
    final int max = value.length;
    if (fromIndex < 0) {
        fromIndex = 0;
    } else if (fromIndex >= max) {
       return -1;
    }
    if (ch < Character.MIN_SUPPLEMENTARY_CODE_POINT) {
       final char[] value = this.value;
        for (int i = fromIndex; i < max; i++) {
            if (value[i] == ch) {
                return i;
            }
        }
        return -1;
    } else {
        return indexOfSupplementary(ch, fromIndex);
    }
}
```

Live profile



AZUL SYSTEMS®

After normalization

BCI:7 Value: True:0 False: 2471

BCI:17 Value: True:357 False: 2112

BCI:25 Value: True:2112 False: 0

BCI:40 Value: True:149604 False: 2003

BCI:49 Value: True:109 False: 149495

BCI:7 Value: True:0 False: 1000

BCI:17 Value: True:100 False: 1000

BCI:25 Value: True:1000 False: 0

BCI:40 Value: True:100000 False: 1000

BCI:49 Value: True:100 False: 100000

Challenges

- ✓ Class Referencing
- ✓ Profile Normalization
- Class' and method' instrumentation
- Classloader identity
- Class generators
- Static Initializers

Class' and method' instrumentation

AZUL SYSTEMS®

modified j.l.String::indexOf

```
public int indexOf(int ch, int fromIndex) {
    final int max = value.length;
    if (fromIndex < 0) {
        fromIndex = 0;
    } else if (fromIndex >= max) {
       return -1;
    }
    if (ch < Character.MIN_SUPPLEMENTARY_CODE_POINT) {
       final char[] value = this.value;
        for (int i = fromIndex; i < max; i++) {
            if (value[i] == ch) {
                return i;
            }
        }
        return -1;
    } else {
        return indexOfSupplementary(ch, fromIndex);
    }
}
```

modified j.l.String::indexOf

```
public int indexOf(int ch, int fromIndex) {
    final int max = value.length;
    if (fromIndex < 0) {
        fromIndex = 0;
    } else if (fromIndex >= max) {
       return -1;
    }
    if (ch < Character.MIN_SUPPLEMENTARY_CODE_POINT) {
       final char[] value = this.value;
        for (int i = fromIndex; i < max; i++) {
            if (value[i] == ch) {
                return i;
            }
        }
        return -1;
    } else {
        return indexOfSupplementary(ch, fromIndex);
    }
}
```

```
MyAgent.incrementCounter();
```

modified j.l.String::indexOf

```
public int indexOf(int ch, int fromIndex) {
    MyAgent.incrementCounter();
    final int max = value.length;
    if (fromIndex < 0) {
        fromIndex = 0;
    } else if (fromIndex >= max) {
       return -1;
    }
    if (ch < Character.MIN_SUPPLEMENTARY_CODE_POINT) {
       final char[] value = this.value;
        for (int i = fromIndex; i < max; i++) {
            if (value[i] == ch) {
                return i;
            }
        }
        return -1;
    } else {
        return indexOfSupplementary(ch, fromIndex);
    }
}
```

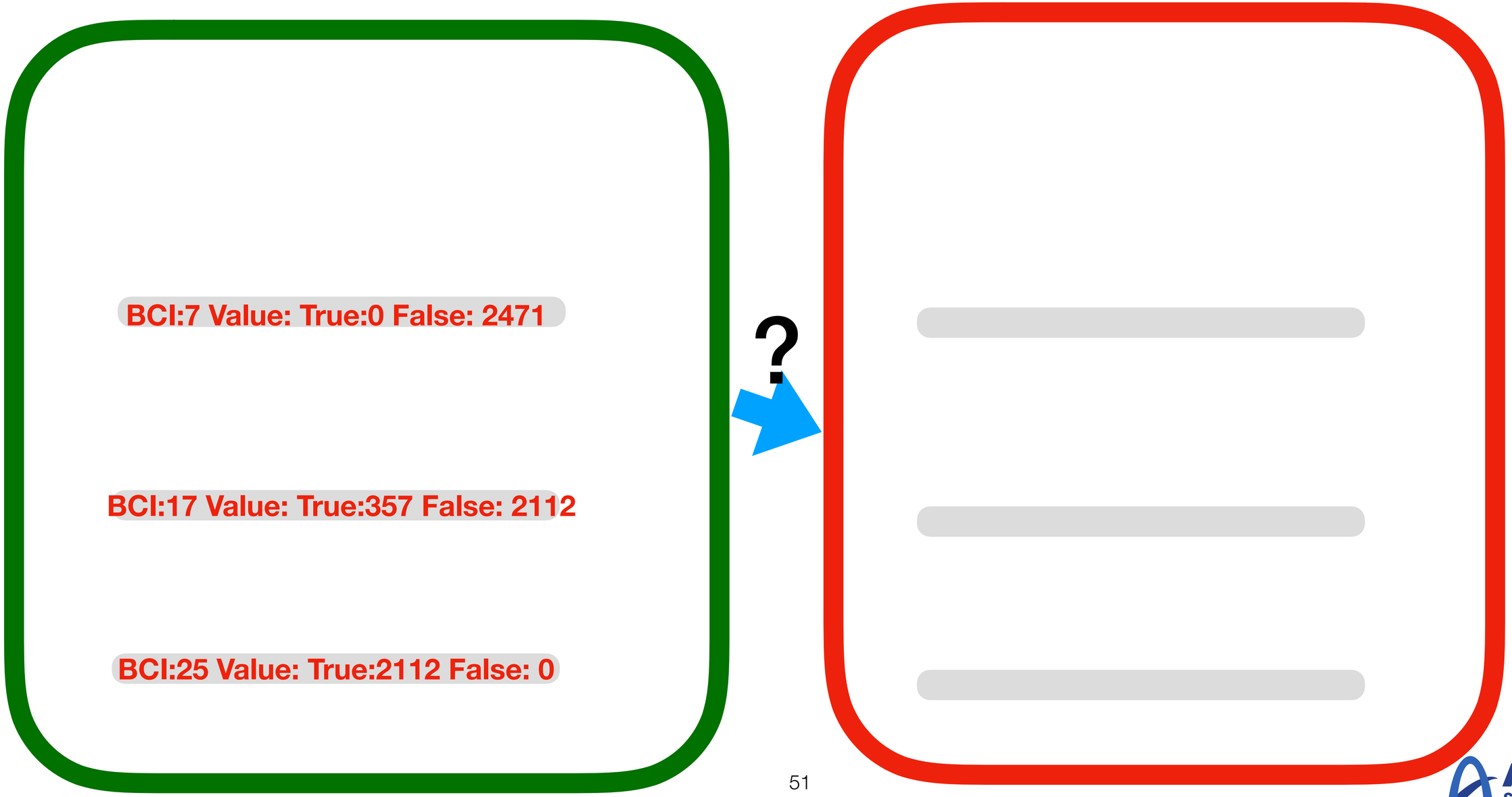
```
public int indexOf(int, int);
  Code:
     0: invokestatic  #61
     3: aload_0
     4: getfield      #3
     7: arraylength
     8: istore_3
     9: iload_2
    10: ifge          18
    13: iconst_0
    14: istore_2
    15: goto          25
    18: iload_2
    19: iload_3
    20: if_icmplt     25
    23: iconst_m1
    24: ireturn
    25: iload_1
    26: ldc           #63
    28: if_icmpge     66
    31: aload_0
     *  *  *
```

Without and with modification

```
public int indexOf(int, int);
  Code:
     0: aload_0
     1: getfield      #3
     4: arraylength
     5: istore_3
     6: load_2
     7: ifge          15
    10: iconst_0
    11: istore_2
    12: goto          22
    15: iload_2
    16: iload_3
    17: if_icmplt     22
    20: iconst_m1
    21: ireturn
    22: iload_1
    23: ldc           #62
    25: if_icmpge     63
    28: aload_0
     *   *   *
```

```
public int indexOf(int, int);
   Code:
      0: invokestatic  #61
      3: aload_0
      4: getfield      #3
      7: arraylength
      8: istore_3
      9: iload_2
     10: ifge          18
     13: iconst_0
     14: istore_2
     15: goto          25
     18: iload_2
     19: iload_3
     20: if_icmplt     25
     23: iconst_m1
     24: ireturn
     25: iload_1
     26: ldc           #63
     28: if_icmpge     66
     31: aload_0
      *   *   *
```

Without and with modification



AZUL SYSTEMS®

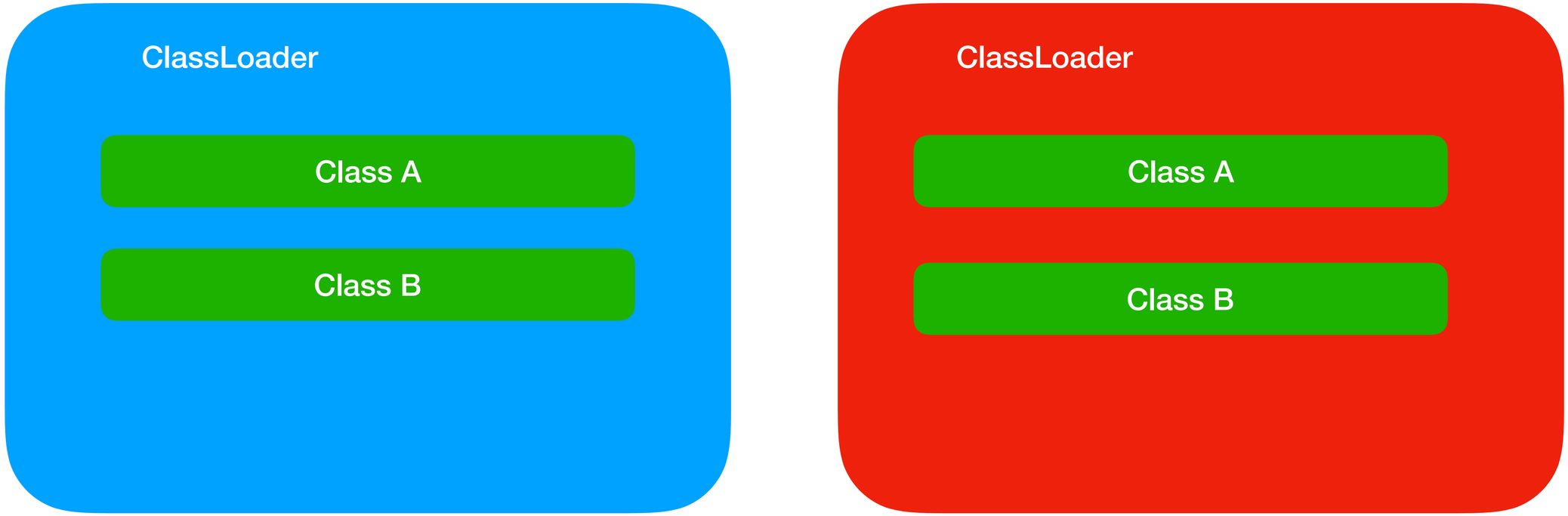
Challenges

- ✓ Class Referencing
- ✓ Profile Normalization
- ✓ Class' and method' instrumentation
- Classloader identity
- Class generators
- Static Initializers

Classloader Identification

AZUL SYSTEMS®

Classloader example



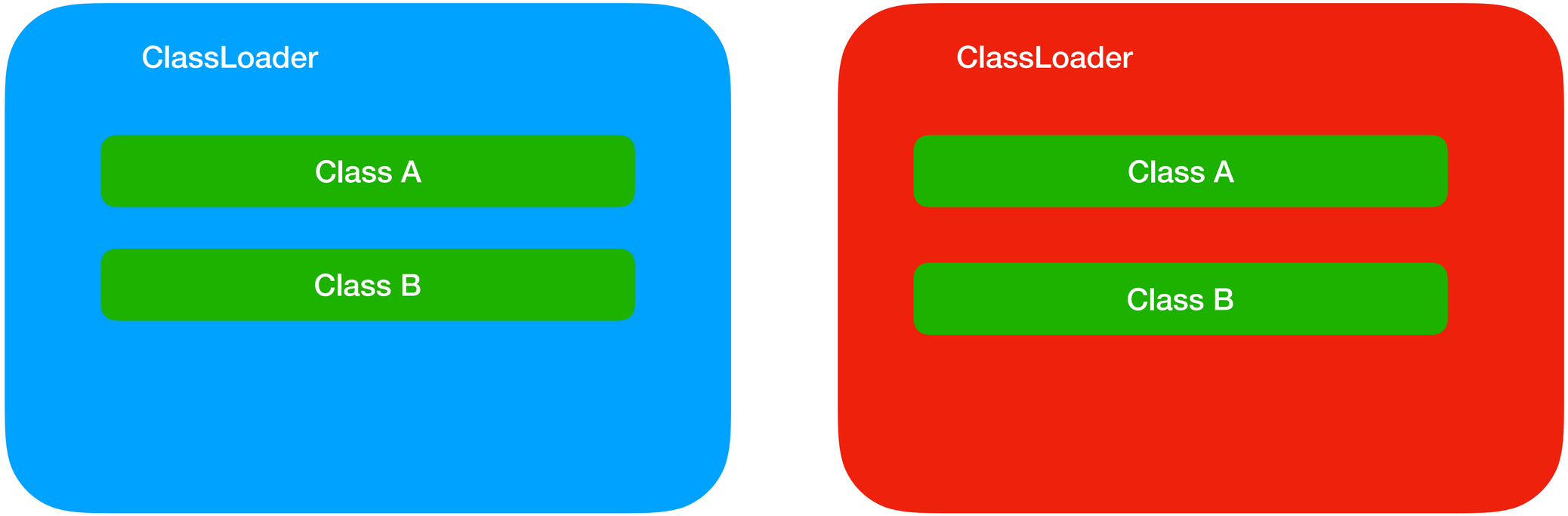
Example 1

Classloader example

```
class CLexample {

    CustomClassLoader loader1 = new CustomClassLoader();
    CustomClassLoader loader2 = new CustomClassLoader();

    Class<?> class1A  = loader1.loadClass("ClassA");
    Class<?> class2A  = loader1.loadClass("ClassA");

    Class<?> class1B  = loader2.loadClass("ClassB");
    Class<?> class2B  = loader2.loadClass(“ClassB");
}

class CustomClassLoader extends ClassLoader{
    //…
}
```

Example 1

Classloader example



Example 1

Classloader example - Java 9

```
class CLexample_v9 {

    CustomClassLoader loader1 = new CustomClassLoader(“Foo”);
    CustomClassLoader loader2 = new CustomClassLoader(“Bar”);

    Class<?> class1A  = loader1.loadClass("ClassA");
    Class<?> class2A  = loader1.loadClass("ClassA");

    Class<?> class1B  = loader1.loadClass("ClassB");
    Class<?> class2B  = loader1.loadClass("ClassB");
}

class CustomClassLoader extends ClassLoader{
    public CustomClassLoader(String name) {
        super(name, null);
    }
    ...
}
```

Example 1

Classloader names

- Same name for all - not useful
- Truly unique (i.e. Random()) - not useful
- Unique within the run, repeated across runs - good!

Classloaders identification algos

- Name
- Automatic
  - works for classloaders created for distinct purposes

Challenges

- ✓ Class Referencing
- ✓ Profile Normalization
- ✓ Class' and method' instrumentation
- ✓ Classloader identity
- Class generators
- Static Initializers

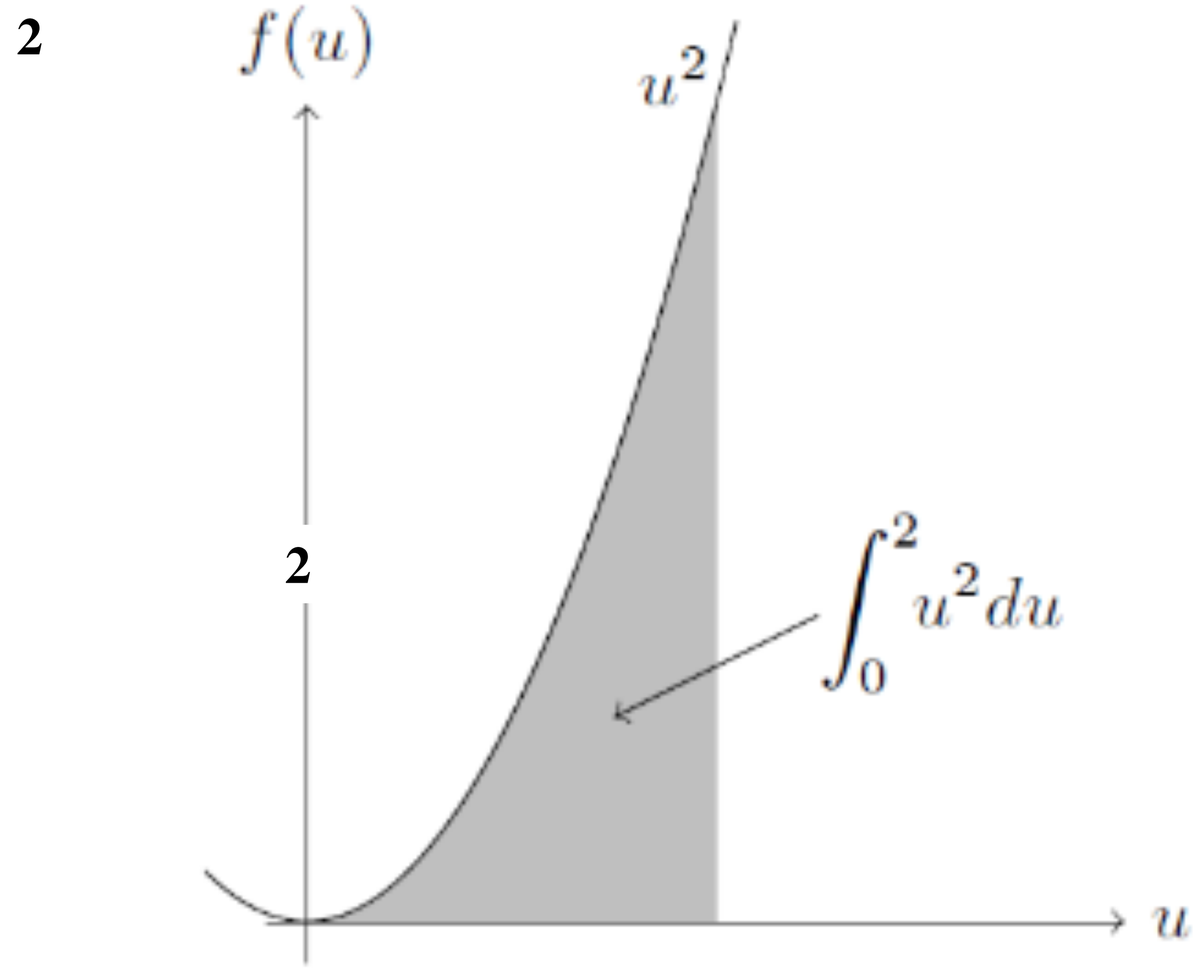
Class generators

AZUL SYSTEMS®

Class generators

Case study
Lambda metafactory

Lambda example (1)



AZUL SYSTEMS®

Lambda example (2)

```
class Square {
    public static DoubleUnaryOperator fn = d -> d * d;
}
class Cube {
    public static DoubleUnaryOperator fn = d -> d * d * d;
}
```

Lambda example (3)

```
public double computeIntegral(DoubleUnaryOperator fn) {
    double result = 0.0;
    for (double d = start; d < end + IntegralTest.espilon ; d +=step) {
        result += step * op.applyAsDouble(d);
    }
    return result;
}
```

Lambda example - two runs

```
{
 t1 = new Thread("Thread 1") {
      public void run(){
          test.integralValue1 = test.computeIntegral(Square.fn);
      }
 }.start();
 t2 = new Thread("Thread 2") {
      public void run(){
          test.integralValue2 = test.computeIntegral(Cube.fn);
      }
 }.start();
}
```

Generated Classes

```
java -Djdk.internal.lambda.dumpProxyClasses=./lambda_classes_openjdk  IntegralTest
```

First run

```
$ ls lambda_classes_openjdk

‘Square$$Lambda$1.class’
‘Cube$$Lambda$2.class'
```

Second run

```
$ ls lambda_classes_openjdk

'Square$$Lambda$2.class'
‘Cube$$Lambda$1.class'
```

TraceClassLoading

```
java -XX:+TraceClassLoading IntegralTest
```

```
Order 1

[Loaded Square$$Lambda$1/531885035 from Square]
[Loaded Cube$$Lambda$2/142257191 from Cube]
                                                Order 2

                [Loaded Cube$$Lambda$1/1418481495 from Cube]
                [Loaded Square$$Lambda$2/1044036744 from Square]
```

LMF solution

- Normalize names
- Replace LambdaMetafactory with an alternative

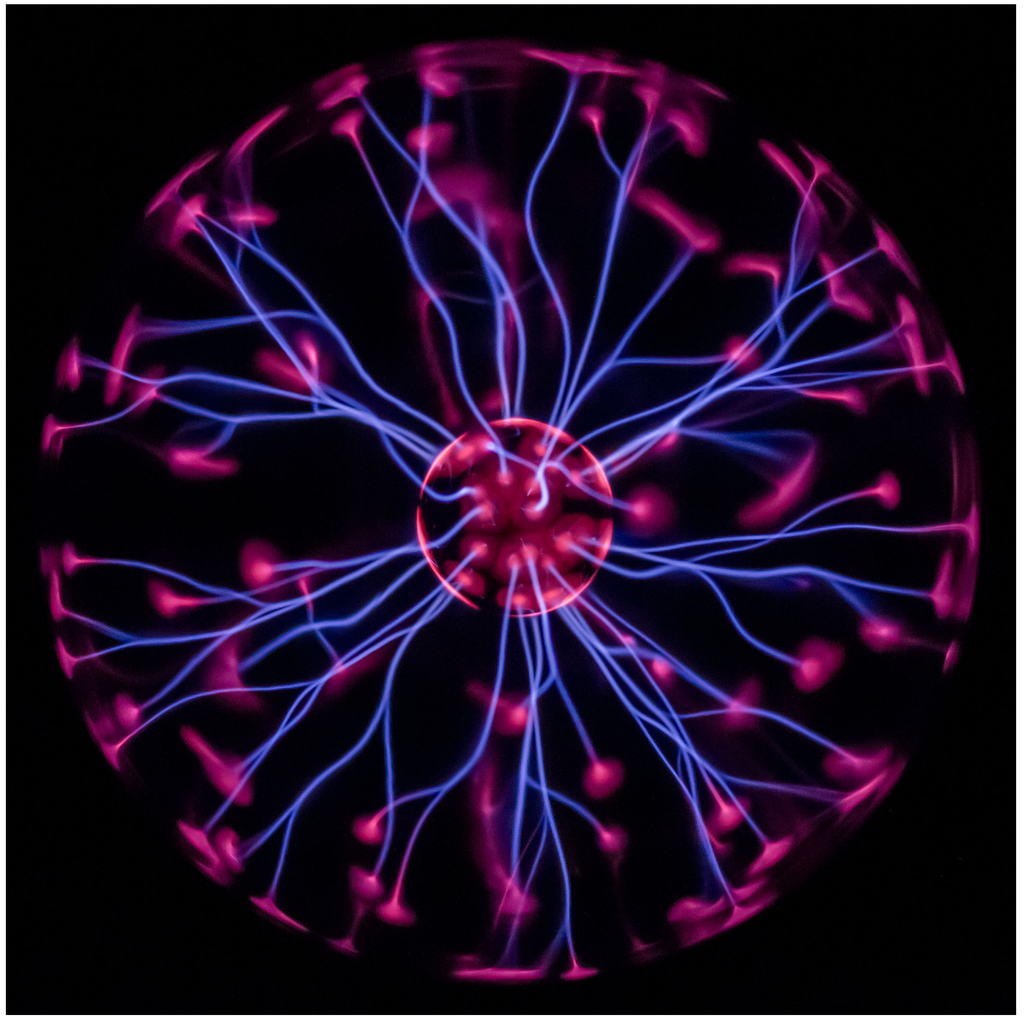
Shout to classgen writers

- Classname
  - Unique within a given run
  - Consistent across runs

Challenges

- ✓ Class Referencing
- ✓ Profile Normalization
- ✓ Class' and method' instrumentation
- ✓ Classloader identity
- ✓ Class generators
- Static Initializers

Static Initializers



AZUL SYSTEMS®

Terminology: method *bar* requires class *Foo*

```
void bar(int param) {
     if (param != 0) {
          Foo a;
          // * * *
     } else {
          // * * *
     }
}
```

When class may be initialized?

JVMLS for Java 8. Section 5.5: Initialization

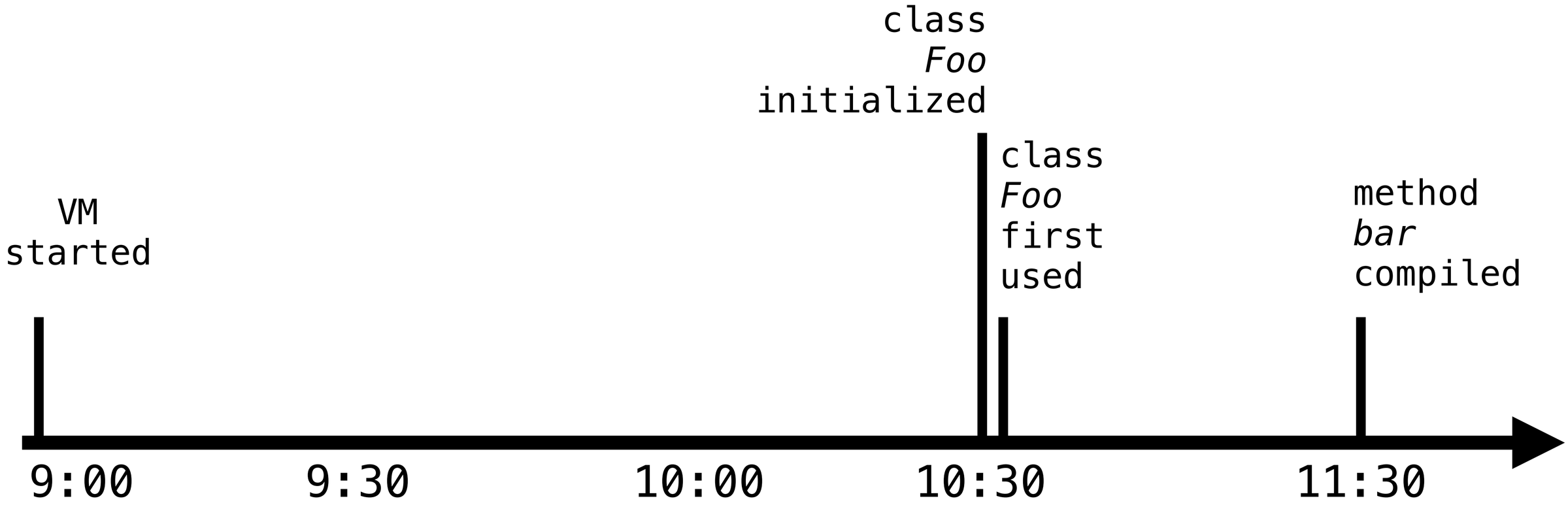
Initialization of a class or interface consists of executing its class or interface initialization method (§2.9). A class or interface C may be initialized only as a result of:

- The execution of any one of the Java Virtual Machine instructions new, getstatic, putstatic, or invokestatic that references C (§new, §getstatic, §putstatic, §invokestatic). These instructions reference a class or interface directly or indirectly through either a field reference or a method reference

…..

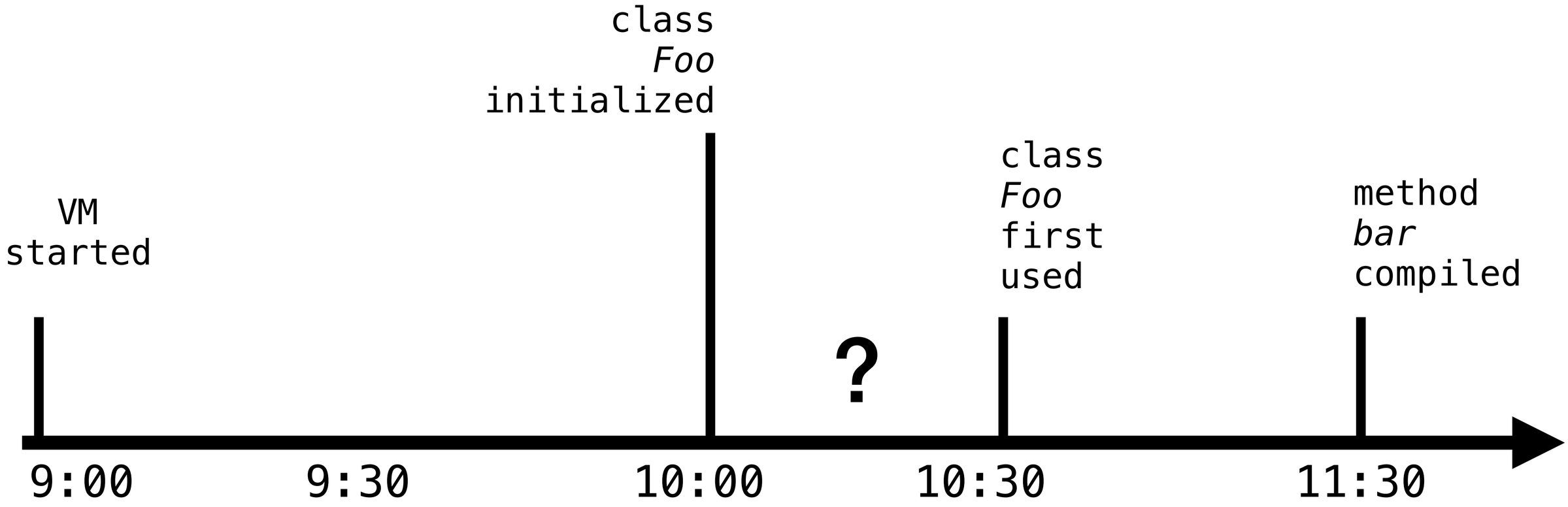
https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf

Order of events



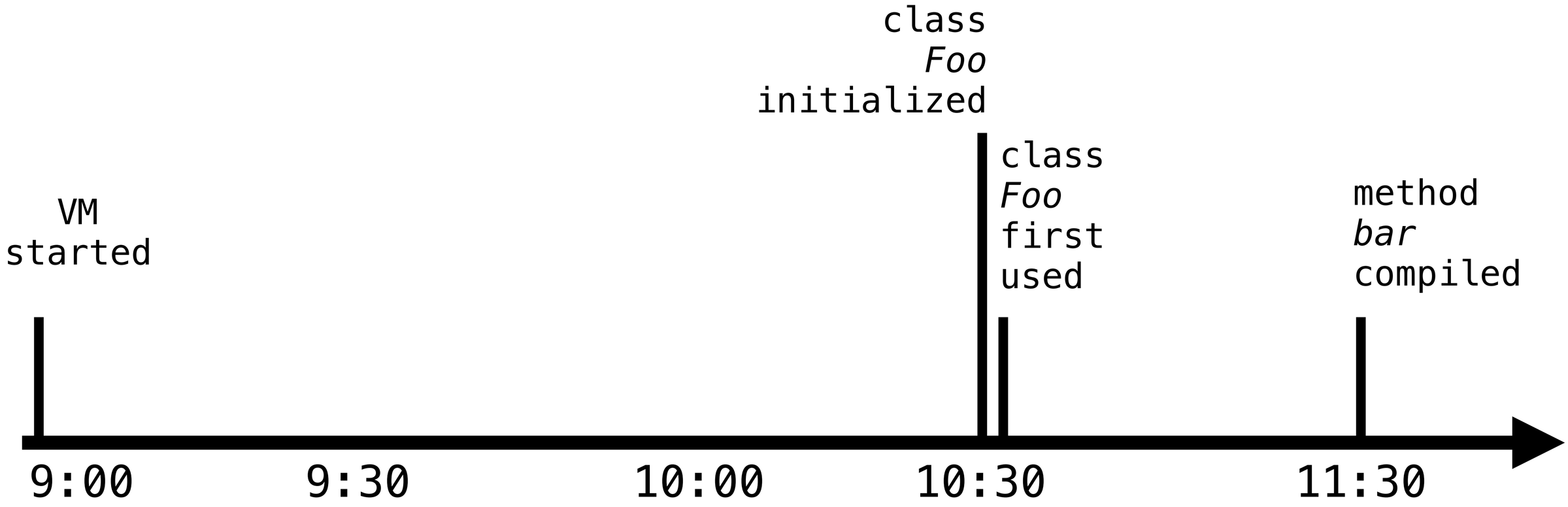
AZUL SYSTEMS®

Order of events



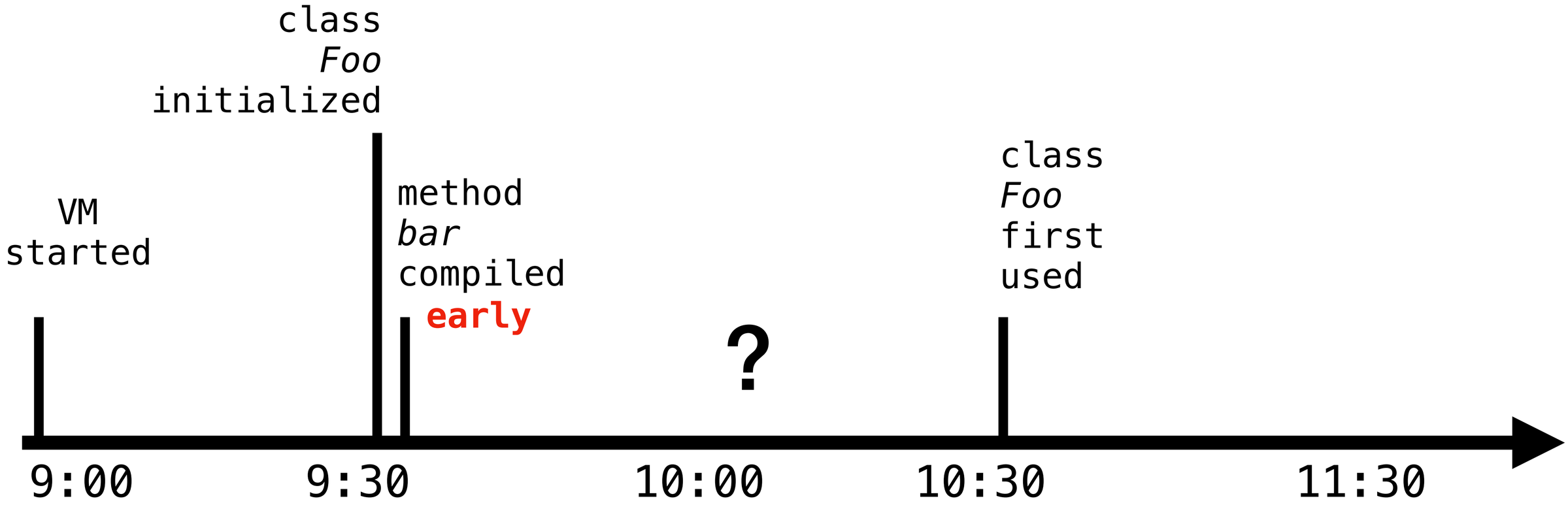
AZUL SYSTEMS®

Order of events



AZUL SYSTEMS®

Order of events



AZUL SYSTEMS®

May JVM early initialize this?

```
class Foo {
    int baz;
}
```

```
class Foo0 {
  int bar;

  Foo1();
    Code:
       0: aload_0
       1: invokespecial #1// Method j.l.Object.”<init>”:()V
       4: return
}
```

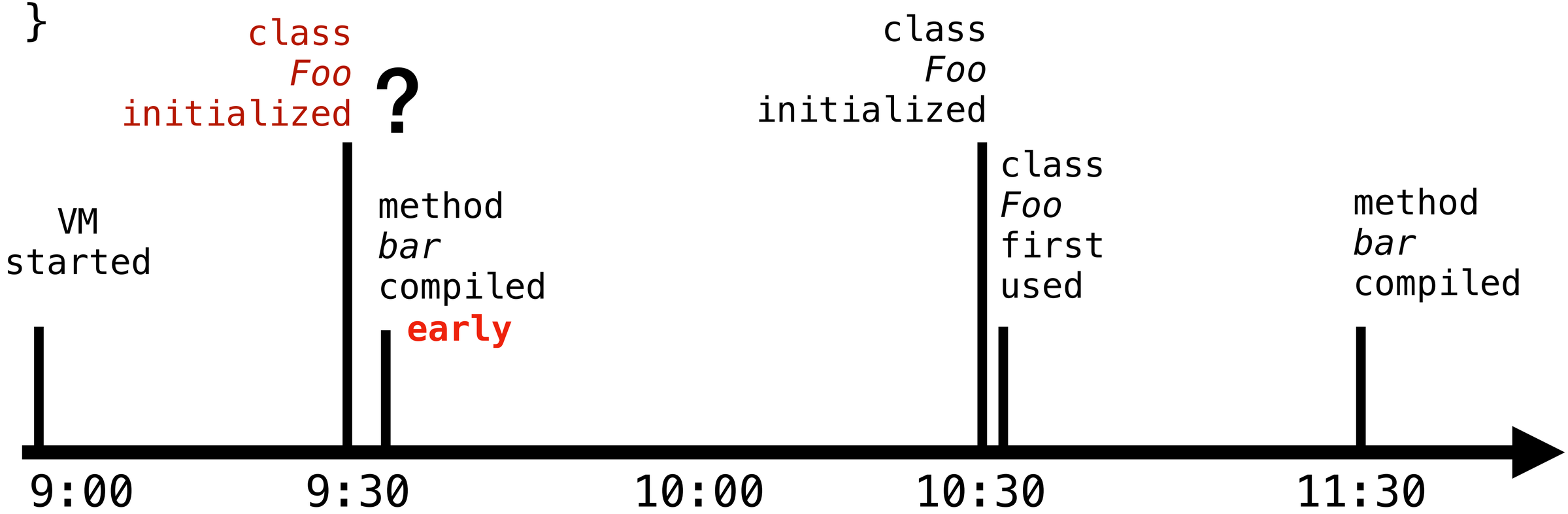
And this?

```
class Foo {
  final static Integer bad = new Integer(42);
}
```

```
class Foo {
  static final java.lang.Integer bar;

  Foo2();
    Code:
       0: aload_0
       1: invokespecial #1 // Method j.l.Object.”<init>”:()V
       4: return

  static {};
    Code:
       0: new           #2 // class j.l.Integer
       3: dup
       4: bipush        42
       6: invokespecial #3 // Method j.l.Integer.”<init>”:(I)V
       9: putstatic     #4 // Field bar:Ljava/lang/Integer;
      12: return
}
```

And this???

```
class Foo {
  static final LocalDateTime baz = LocalDateTime.now();
}
```



AZUL SYSTEMS®

And this???

```
class Foo3 {
  final LocalDateTime bar = LocalDateTime.now();
}
```

```
class Foo3 {
  static final java.time.LocalDateTime bar;

  Foo3();
    Code:
       0: aload_0
       1: invokespecial #1  // Method java.lang.Object.”<init>”:()V
       4: return

  static {};
    Code:
       0: invokestatic  #2  // Method j.time.LocalDateTime.now:()Ljava/time/LocalDateTime;
       3: putstatic     #3  // Field bar:Ljava/time/LocalDateTime;
       6: return
}
```

SimpleEnum

```
public enum SimpleEnum {
   One
};
```

```
>ll SimpleEnum.java
-rw-r--r--  1 ivan  staff  35 Feb 23 21:30 SimpleEnum.java
>ll SimpleEnum.class
-rw-r--r--  1 ivan  staff  732 Feb 23 21:30 SimpleEnum.class
```

SimpleEnum

```
public enum SimpleEnum {
   One
};
```

```
Compiled from "SimpleEnum.java"
public final class SimpleEnum extends java.lang.Enum<SimpleEnum> {
  public static final SimpleEnum One;

  public static SimpleEnum[] values();
    Code:
       0: getstatic     #1                  // Field $VALUES:[LSimpleEnum;
       3: invokevirtual #2                  // Method "[LSimpleEnum;".clone:()Ljava/lang/Object;
       6: checkcast     #3                  // class "[LSimpleEnum;"
       9: areturn

  public static SimpleEnum valueOf(java.lang.String);
    Code:
       0: ldc           #4                  // class SimpleEnum
       2: aload_0
       3: invokestatic  #5                  // Method java/lang/Enum.valueOf:(Ljava/lang/Class;Ljava/lang/String;)Ljava/lang/Enum;
       6: checkcast     #4                  // class SimpleEnum
       9: areturn

  static {};
    Code:
       0: new           #4                  // class SimpleEnum
       3: dup
       4: ldc           #7                  // String One
       6: iconst_0
       7: invokespecial #8                  // Method "<init>":(Ljava/lang/String;I)V
      10: putstatic     #9                  // Field One:LSimpleEnum;
      13: iconst_1
      14: anewarray     #4                  // class SimpleEnum
      17: dup
      18: iconst_0
      19: getstatic     #9                  // Field One:LSimpleEnum;
      22: aastore
      23: putstatic     #1                  // Field $VALUES:[LSimpleEnum;
      26: return
}
```

Is it simple?

```
public enum SimpleEnum {
   One
};
```

```
  static {
    Code:
       0: new           #4                  // class SimpleEnum
       3: dup
       4: ldc           #7                  // String One
       6: iconst_0
       7: invokespecial #8                  // Method "<init>":(Lj.l.String;I)V
      10: putstatic     #9                  // Field One:LSimpleEnum;
      13: iconst_1
      14: anewarray     #4                  // class SimpleEnum
      17: dup
      18: iconst_0
      19: getstatic     #9                  // Field One:LSimpleEnum;
      22: aastore
      23: putstatic     #1                  // Field $VALUES:[LSimpleEnum;
      26: return
}
```

Challenges

- ✓ Class Referencing
- ✓ Profile Normalization
- ✓ Class’ and method’ instrumentation
- ✓ Classloader identity
- ✓ Class generators
- ✓ Static Initializers

Conclusions

- Innovations in JITs and Runtimes continue
- ”Assisted” warmup is challenging

Can I try it?

- ReadyNow
  - Since 04/2015
- Falcon
  - Since 12/2016
- Compile Stashing
  - Since 12/2017

Questions?

How to try?

http://docs.azul.com/zing/zing-quick-start-fp.htm
http://www.azul.com/zingtrial/

Iván Krȳlov
@JohnWings

AZUL SYSTEMS®

Pictures references

https://www.publicdomainpictures.net/en/view-image.php?image=133995&picture=honda-generator
https://cdn.pixabay.com/photo/2016/09/01/17/18/profile-1636642_960_720.png
http://www.publicdomainpictures.net/download-picture.php?id=117770&check=ae32542cbc42a32769782673144d168d
http://www.picserver.org/highway-signs2/images/reference.jpg
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTR6oN4158MtVGPSuQ_hBoQLlaVTe7eq_U3CTCFqr9LOE2KprrD
http://techfrag.com/wp-content/uploads/2015/10/Intel_Xeon_Skylake_0.jpg
https://cdn.comsol.com/wordpress/2014/04/integral.png
https://www.maxpixel.net/static/photo/2x/Hashtag-Hash-Social-Analytics-Hash-Tag-Pencils-2998837.jpg
https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Plasma_globe_60th.jpg/1920px-Plasma_globe_60th.jpg
https://tenyearsago.files.wordpress.com/2012/06/bourne-2_2.jpg