

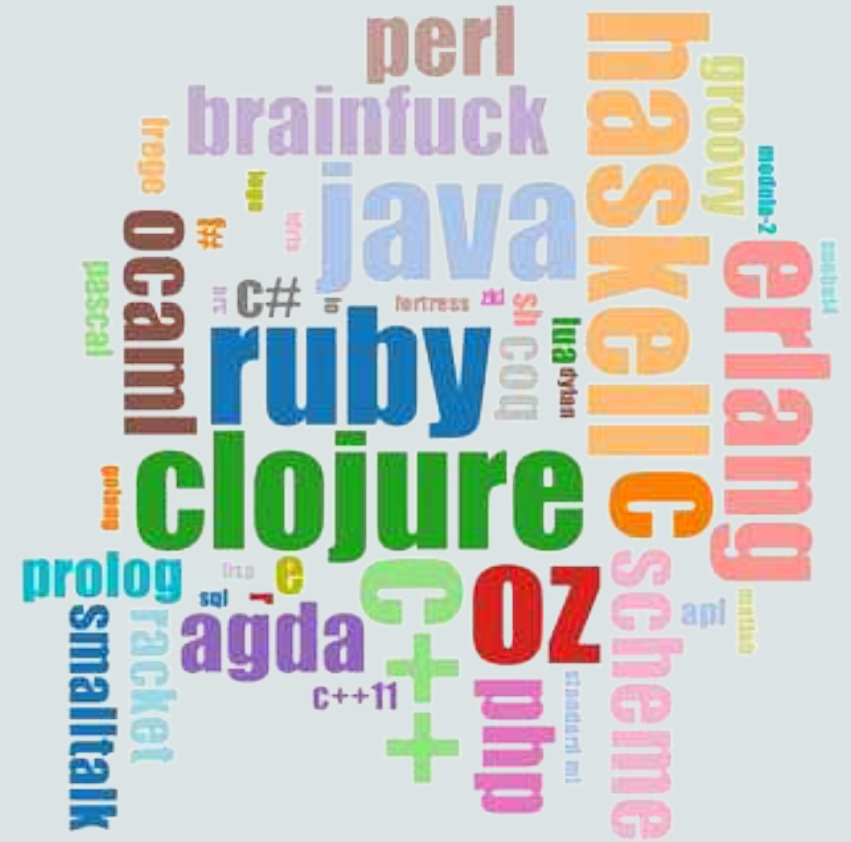
ORACLE®

Understanding How Graal Works

a Java JIT Compiler Written in Java



@JaroslavTulach
Oracle Labs



Jaroslav Tulach

NetBeans Founder – 20 years ago

NetBeans Initial Architect – up to 2001

Practical API Design book – published 2008

Java/JavaScript/co. Interop – 2012 - now

Oracle Labs: Graal/Truffle – 2015 - now



Program Agenda

Short & mostly incorrect history

What is a JIT compiler?

Why write JIT compiler in Java?

Inevitable polyglot destiny

It all begins in MatFyz

- MatFyz 1992-1998 & Xelfi 1995-1997
- NetBeans & Roman Staněk
 - IDE vs. Platform
- Sun Microsystems 1999
 - Open Source – May 2000
- Oracle 2010
 - JDeveloper & OSGi



Safe Harbor Statement

The preceding and following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

<http://chrisseaton.com/rubytruffle/jokerconf17/>

```
$ git clone https://github.com/dmlloyd/openjdk.git
```

```
openjdk/hotspot/src/share/vm/opto
```



```
divnode.cpp — ~/Documents/jokerconf17/demo/openjdk
Project
  convertnode.hpp
  countbitsnode.cpp
  countbitsnode.hpp
  divnode.cpp
  divnode.hpp
  doCall.cpp
  domgraph.cpp
  escape.cpp
  escape.hpp
  gcm.cpp
  generateOptoStub.cpp
  graphKit.cpp
  graphKit.hpp
  IdealGraphPrinter.cpp
  IdealGraphPrinter.hpp
  IdealKit.cpp
  IdealKit.hpp
  ifg.cpp
  ifnode.cpp
  indexSet.cpp
  indexSet.hpp
  Intrinsicnode.cpp
  Intrinsicnode.hpp
  lcm.cpp

568
569 //-----Idealize-----
570 // Dividing by a power of 2 is a shift.
571 Node *DivlNode::Ideal( PhaseGVN *phase, bool can_reshape ) {
572     if ( in(0) && remove_dead_region(phase, can_reshape) ) return this;
573     // Don't bother trying to transform a dead node
574     if( in(0) && in(0)->is_top() ) return NULL;
575
576     const Type *t = phase->type( in(2) );
577     if( t == TypeLong::ONE ) // Identity?
578         return NULL; // Skip it.
579
580     const TypeLong *tl = t->isa_long();
581     if( !tl ) return NULL;
582
583     // Check for useless control input
584     // Check for excluding div-zero case
585     if ( in(0) && (tl->hi < 0 || tl->lo > 0) ) {
586         set_req(0, NULL); // Yank control input
587         return this;
588     }
589
590     if( !tl->is_con() ) return NULL;
591     jlong l = tl->get_con(); // Get divisor
592
593     if ( l == 0 ) return NULL; // Dividing by zero constant does not idealize
594
595     // Dividing by MINLONG does not optimize as a power-of-2 shift.
596     if( l == min_jlong ) return NULL;
597
598     return transform_long_divide( phase, in(1), l );

```

hotspot/src/share/vm/opto/divnode.cpp 571:60

LF UTF-8 C++ jdk9/jdk9 0 files



<https://www.youtube.com/watch?v=Hqw57GJSrac>

Things I won't do again...

- Write a VM in C/C++
 - Java plenty fast now
 - Mixing OOPS in a non-GC language a total pain
 - Forgetting 'this' is an OOP
 - Across a GC-allowable call
 - Roll-your-own malloc pointless now

<https://www.youtube.com/watch?v=Hqw57GJSrac>

Setting up Graal

```
$ export JAVA_HOME=/bin/jdk-11.0.5/  
$ export PATH=$JAVA_HOME/bin:$PATH  
$ java -version  
java version "11.0.5" 2019-10-15 LTS
```

```
$ git clone https://github.com/graalvm/mx.git  
$ (cd mx; git checkout 5.252.3)  
$ export PATH=`pwd`/mx:$PATH
```

```
$ git clone https://github.com/oracle/graal.git --branch vm-20.0.0
```

```
$ cd graal/compiler  
$ mx --java-home=$JAVA_HOME build
```

Setting up IDE

```
$ unzip netbeans-11.3-bin.zip  
$ export PATH=`pwd`/netbeans/bin:$PATH
```

```
$ cd graal/compiler  
$ mx --java-home=$JAVA_HOME netbeansinit  
$ netbeans -open src/org.graalvm.*
```

Demo

Graal Sources in NetBeans

```
class Demo {  
    public static void main(String[] args) {  
        while (true) {  
            workload(14, 2);  
        }  
    }  
  
    private static int workload(int a, int b) {  
        return a + b;  
    }  
}
```

```
$ javac Demo.java
$ java -XX:+PrintCompilation -XX:CompileOnly=Demo::workload -cp . Demo
...
 78      1      3      Demo::workload (4 bytes)
 78      2      1      Demo::workload (4 bytes)
 78      1      3      Demo::workload (4 bytes)   made not entrant
...
```



```
$ cd graal/compiler
$ mx build && mx unittest GraalAPITest
Updating/creating ./mxbuild/linux-amd64/graaljdk11-cmp
$ /mxbuild/linux-amd64/graaljdk11-cmp/bin/java \
  -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI \
  -XX:+UseJVMCICompiler -XX:-TieredCompilation -XX:+PrintCompilation \
  -XX:CompileOnly=Demo::workload \
  -agentlib:jdwp=transport=dt_socket,server=y,address=8000,suspend=y \
  -cp . Demo
...
8642    17          Demo::workload (4 bytes)
...
```

The JVM compiler interface

```
interface JVMTICompiler {  
    byte[] compileMethod(byte[] bytecode);  
}
```

```
interface JVMCICompiler {
    void compileMethod(CompilationRequest request);
}

interface CompilationRequest {
    JavaMethod getMethod();
}

interface JavaMethod {
    byte[] getCode();
    int getMaxLocals();
    int getMaxStackSize();
    ProfilingInfo getProfilingInfo();
    ...
}
```

```
HotSpot.installCode(targetCode);
```

```
graal - jdk.vm.ci.runtime.JVMCICompiler - Eclipse
JVMCICompiler.class
* Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved.
23 package jdk.vm.ci.runtime;
24
25 import jdk.vm.ci.code.CompilationRequest;
26 import jdk.vm.ci.code.CompilationRequestResult;
27
28 public interface JVMCICompiler {
29     int INVOCATION_ENTRY_BCI = -1;
30
31     /**
32      * Services a compilation request. This object should compile the method to machine code and
33      * install it in the code cache if the compilation is successful.
34      */
35     CompilationRequestResult compileMethod(CompilationRequest request);
36 }
37 |
```

Read-Only Smart Insert 37 : 1

```
graal - org.graalvm.compiler.hotspot/src/org/graalvm/compiler/hotspot/HotSpotGaalCompiler.java - Eclipse
HotSpotGaalCompiler.java
//
78 public class HotSpotGaalCompiler implements GraalJVMCICompiler {
79
80     private final HotSpotJVMCIRuntimeProvider jvmciRuntime;
81     private final HotSpotGaalRuntimeProvider graalRuntime;
82     private final CompilationCounters compilationCounters;
83     private final BootstrapWatchDog bootstrapWatchDog;
84     private List<DebugHandlersFactory> factories;
85
86     HotSpotGaalCompiler(HotSpotJVMCIRuntimeProvider jvmciRuntime, HotSpotGaalRuntimeProvider graalRuntime, OptionValues
87         this.jvmciRuntime = jvmciRuntime;
88         this.graalRuntime = graalRuntime;
89         // It is sufficient to have one compilation counter object per Graal compiler object.
90         this.compilationCounters = Options.CompilationCountLimit.getValue(options) > 0 ? new CompilationCounters(options)
91         this.bootstrapWatchDog = graalRuntime.isBootstrapping() && !DebugOptions.BootstrapInitializeOnly.getValue(options)
92     }
93
94     public List<DebugHandlersFactory> getDebugHandlersFactories() {
95         if (factories == null) {
96             factories = Collections.singletonList(new GraalDebugHandlersFactory(graalRuntime.getHostProviders()).getSnippet
97         }
98         return factories;
99     }
100
101     @Override
102     public HotSpotGaalRuntimeProvider getGraalRuntime() {
103         return graalRuntime;
104     }
105
106     @Override
107     public CompilationRequestResult compileMethod(CompilationRequest request) {
108         ...
109     }
110 }
```

```
graal - org.graalvm.compiler.hotspot/src/org/graalvm/compiler/hotspot/HotSpotGraalCompiler.java - Eclipse
HotSpotGraalCompiler.java
103     return graalRuntime;
104 }
105
106 @Override
107 public CompilationRequestResult compileMethod(CompilationRequest request) {
108     return compileMethod(request, true);
109 }
110
111 @SuppressWarnings("try")
112 CompilationRequestResult compileMethod(CompilationRequest request, boolean installAsDefault) {
113     if (graalRuntime.isShutdown()) {
114         return HotSpotCompilationRequestResult.failure(String.format("Shutdown entered"), false);
115     }
116
117     ResolvedJavaMethod method = request.getMethod();
118     OptionValues options = graalRuntime.getOptions(method);
119
120     if (graalRuntime.isBootstrapping()) {
121         if (DebugOptions.BootstrapInitializeOnly.getValue(options)) {
122             return HotSpotCompilationRequestResult.failure(String.format("Skip compilation because %s is enabled", Det
123         }
124         if (bootstrapWatchDog != null) {
125             if (bootstrapWatchDog.hitCriticalCompilationRateOrTimeout()) {
126                 // Drain the compilation queue to expedite completion of the bootstrap
127                 return HotSpotCompilationRequestResult.failure("hit critical bootstrap compilation rate or timeout", t
128             }
129         }
130     }
131     HotSpotCompilationRequest hsRequest = (HotSpotCompilationRequest) request;
132     try (CompilationWatchDog w1 = CompilationWatchDog.watch(method, hsRequest.getId(), options);
133         BootstrapWatchDog watch w2 = bootstrapWatchDog == null ? null : bootstrapWatchDog.watch(request);
```



```
class HotSpotGraalCompiler implements JVMCICompiler {  
    CompilationRequestResult compileMethod(CompilationRequest request) {  
        System.err.println("Going to compile " + request.getMethod().getName());  
        ...  
    }  
}
```

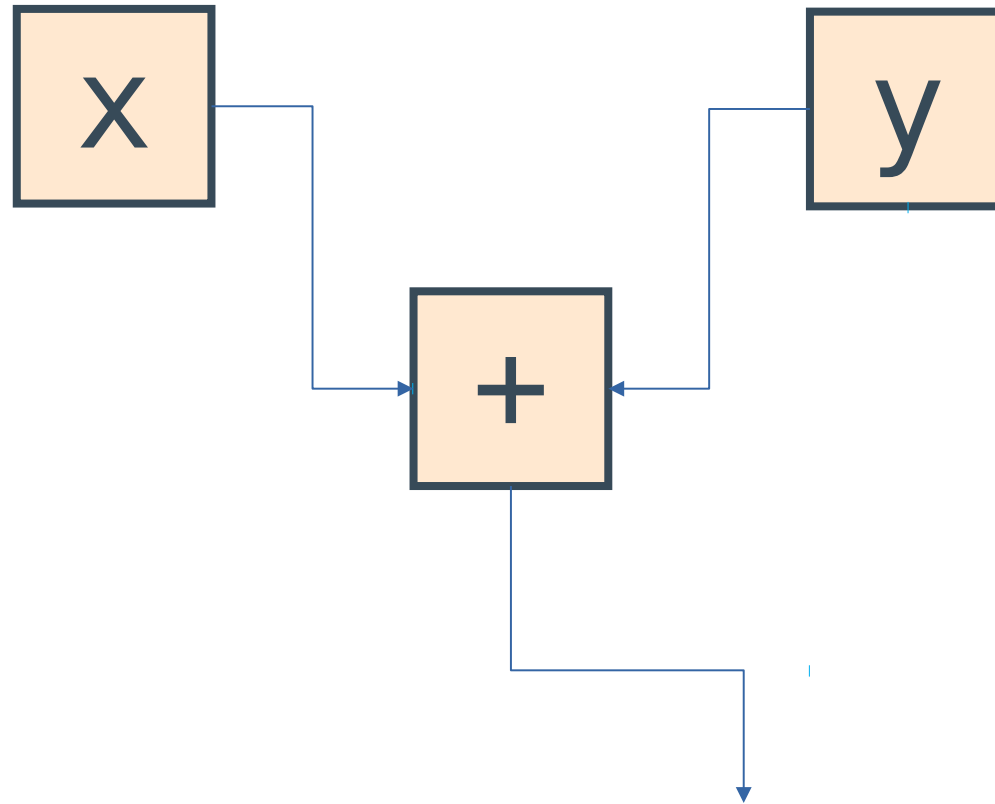
```
graal - org.graalvm.compiler.hotspot/src/org/graalvm/compiler/hotspot/HotSpotGraalCompiler.java - Eclipse
*HotSpotGraalCompiler.java
103     return graalRuntime;
104 }
105
106 @Override
107 public CompilationRequestResult compileMethod(CompilationRequest request) {
108     System.err.println("Going to compile " + request.getMethod().getName());
109     return compileMethod(request, true);
110 }
111
112 @SuppressWarnings("try")
113 CompilationRequestResult compileMethod(CompilationRequest request, boolean installAsDefault) {
114     if (graalRuntime.isShutdown()) {
115         return HotSpotCompilationRequestResult.failure(String.format("Shutdown entered"), false);
116     }
117
118     ResolvedJavaMethod method = request.getMethod();
119     OptionValues options = graalRuntime.getOptions(method);
120
121     if (graalRuntime.isBootstrapping()) {
122         if (DebugOptions.BootstrapInitializeOnly.getValue(options)) {
123             return HotSpotCompilationRequestResult.failure(String.format("Skip compilation because %s is enabled", Det
124         }
125         if (bootstrapWatchDog != null) {
126             if (bootstrapWatchDog.hitCriticalCompilationRateOrTimeout()) {
127                 // Drain the compilation queue to expedite completion of the bootstrap
128                 return HotSpotCompilationRequestResult.failure("hit critical bootstrap compilation rate or timeout", t
129             }
130         }
131     }
132     HotSpotCompilationRequest hsRequest = (HotSpotCompilationRequest) request;
133     try (CompilationWatchDog w1 = CompilationWatchDog.watch(method, hsRequest.getId(), options);
```



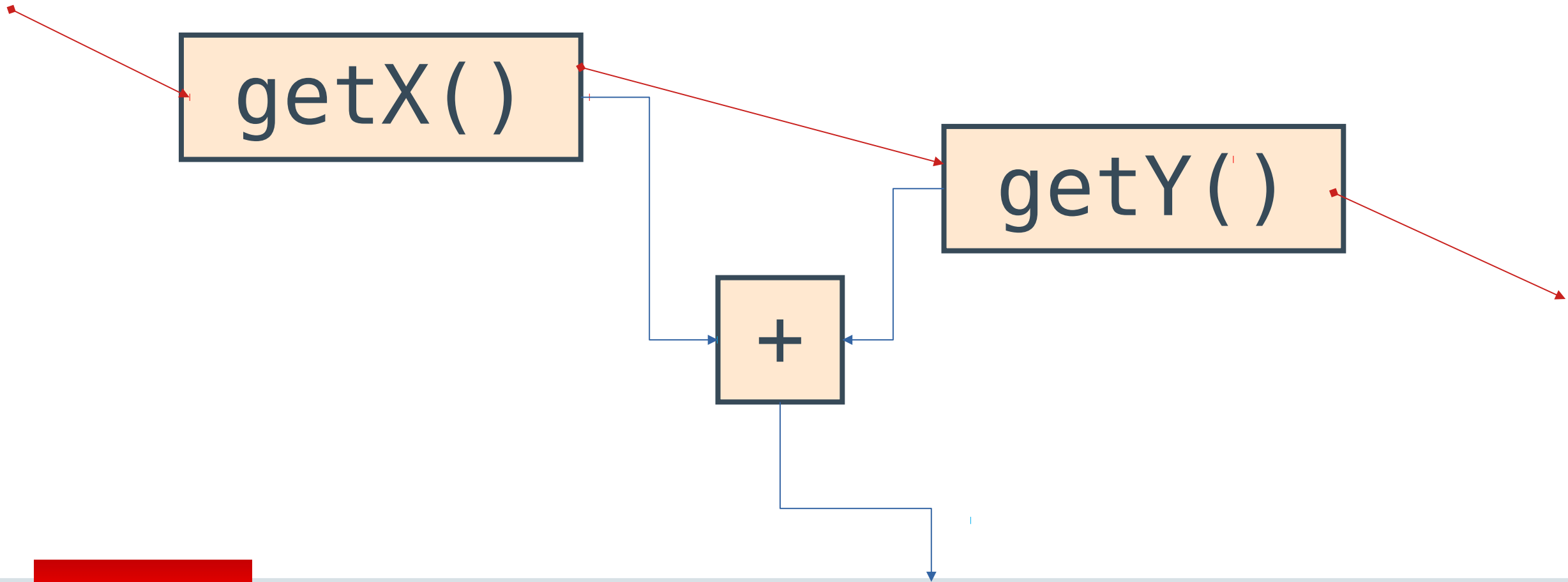
```
$ cd graal/compiler
$ mx build && mx unittest GraalAPITest
Updating/creating ./mxbuild/linux-amd64/graaljdk11-cmp
$ /mxbuild/linux-amd64/graaljdk11-cmp/bin/java \
  -XX:+UnlockExperimentalVMOptions -XX:+EnableJVMCI \
  -XX:+UseJVMCICompiler -XX:-TieredCompilation -XX:+PrintCompilation \
  -XX:CompileOnly=Demo::workload \
  -agentlib:jdwp=transport=dt_socket,server=y,address=8000,suspend=y \
  -cp . Demo
...
Going to compile workload
 8642   17           Demo::workload (4 bytes)
...
```

The Graal graph

$x + y$



getX() + getY()

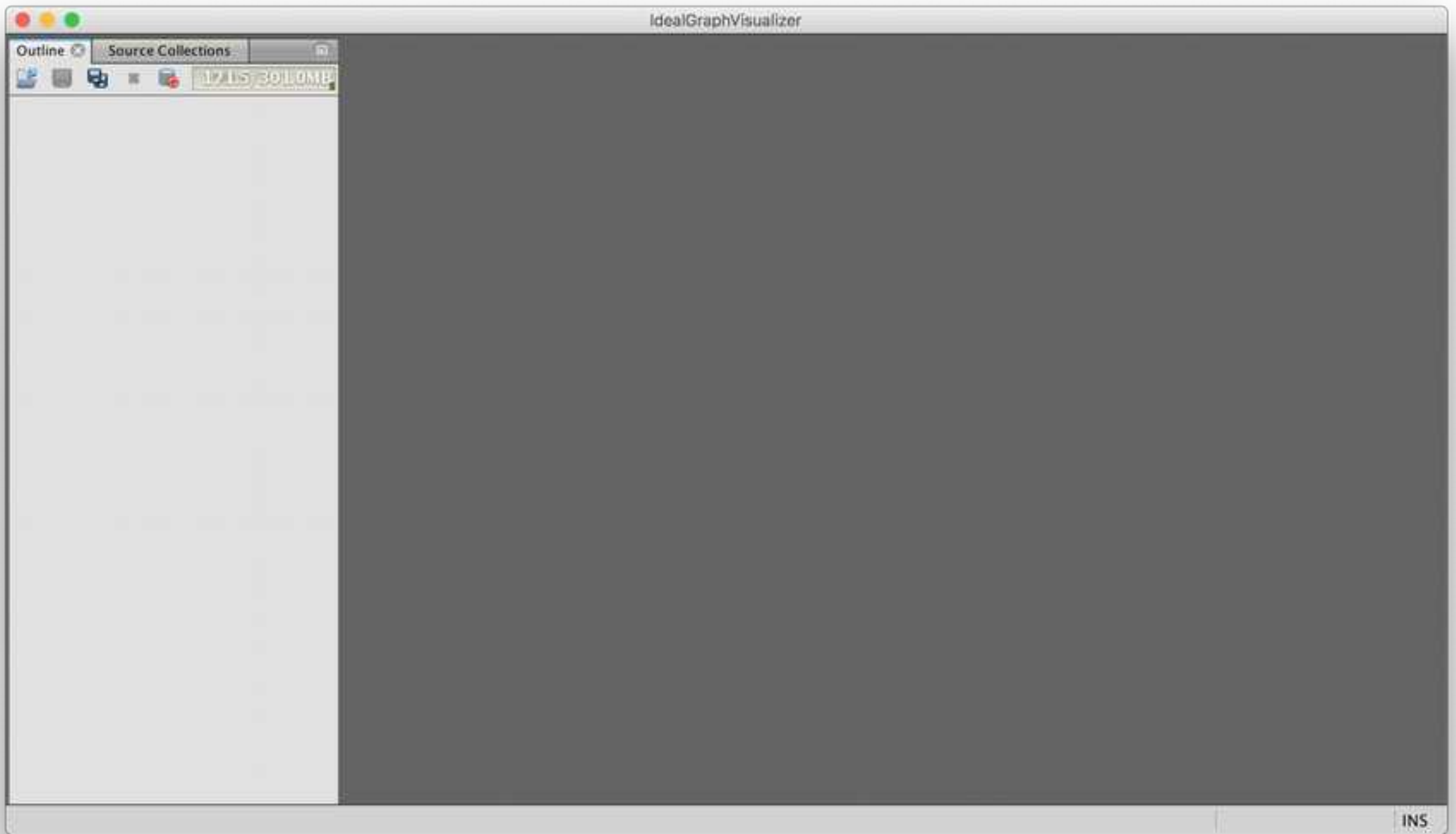


Setting IGV

<https://www.oracle.com/downloads/graalvm-downloads.html>

```
$ unzip /Downloads/idealgraphvisualizer-20.0.0-all.zip  
$ export PATH=`pwd`/idealgraphvisualizer/bin:$PATH
```

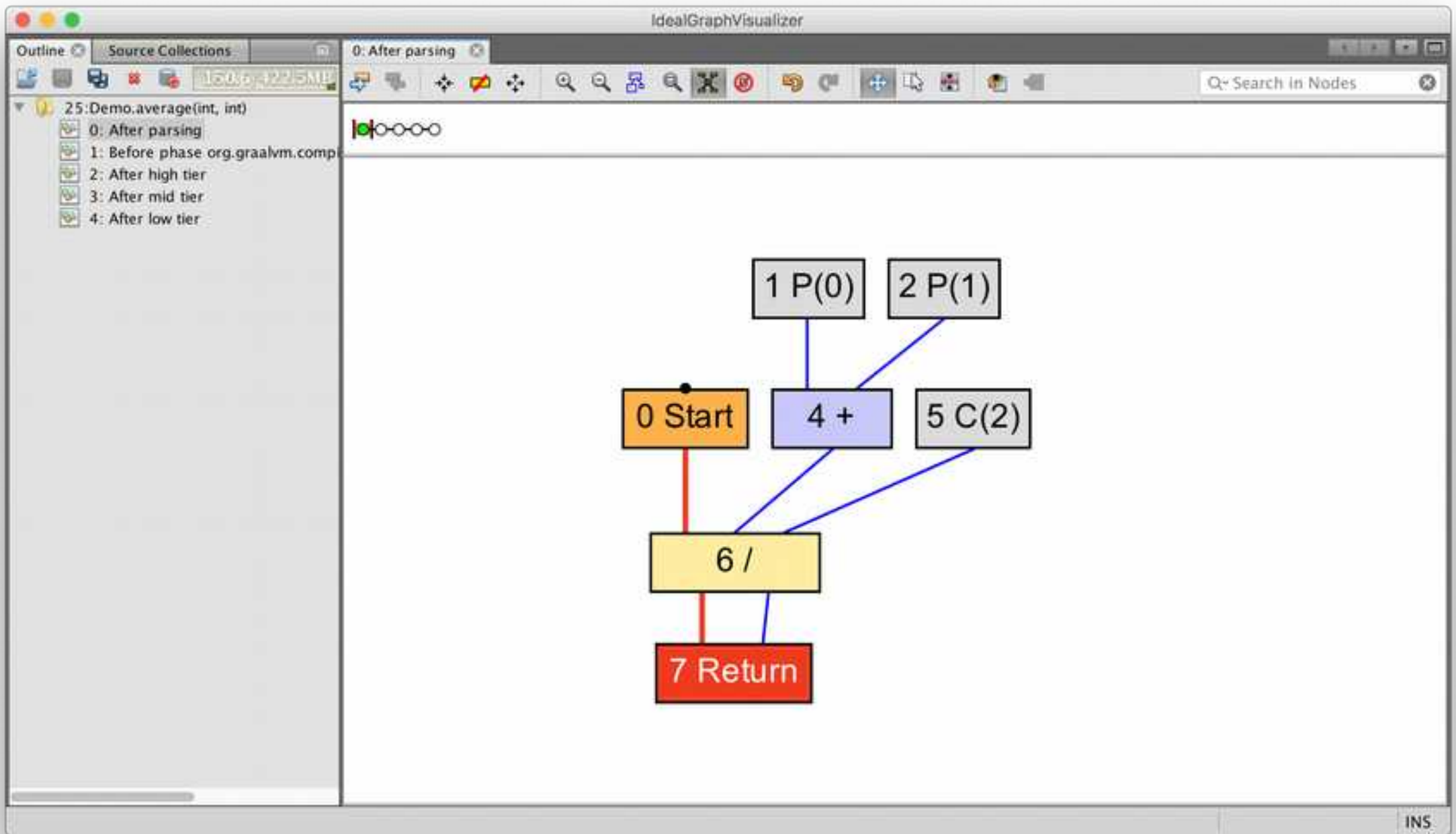
```
$ cd graal/compiler  
$ idealgraphvisualizer -open . # right click & open project
```



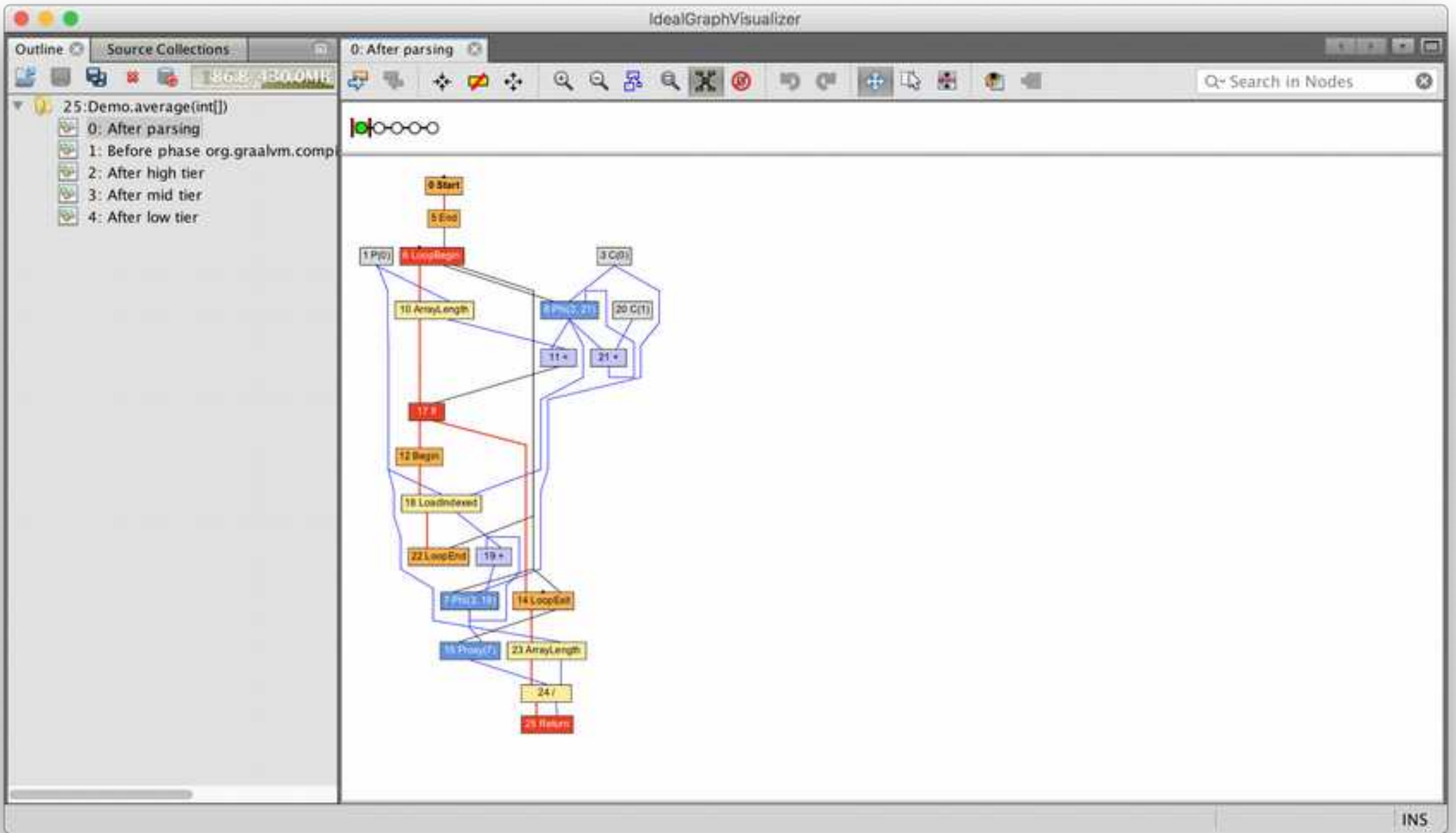
-Dgraal.Dump

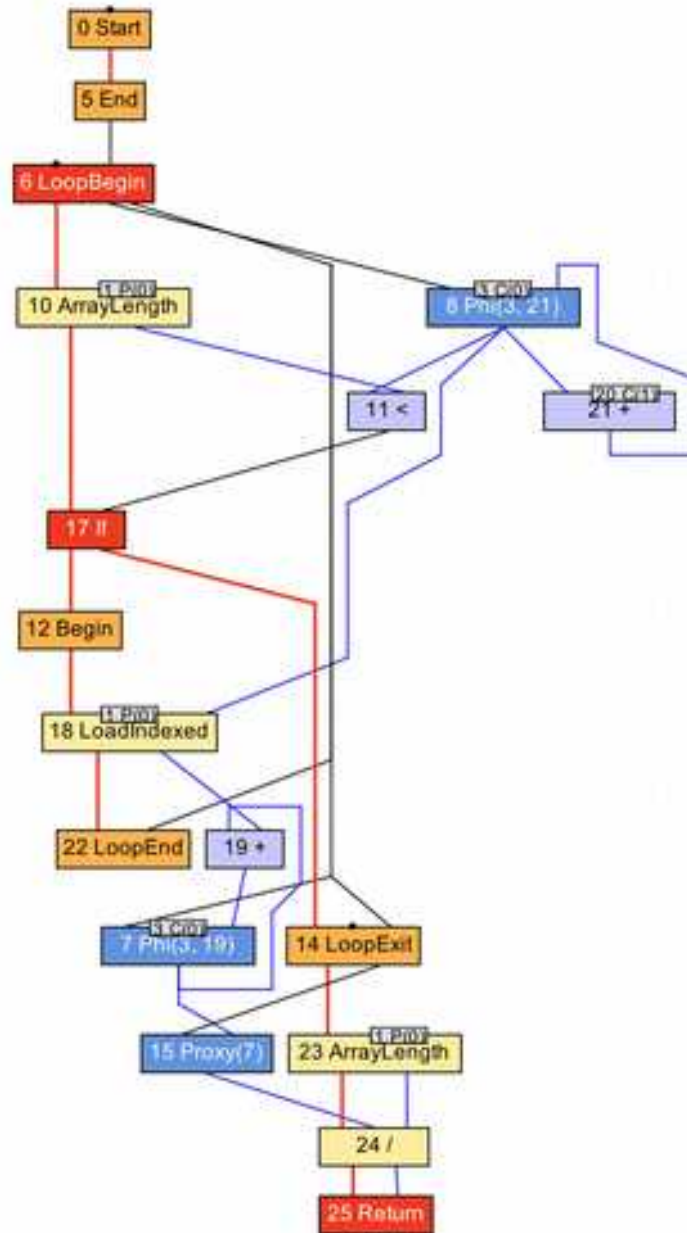
```
$ ./mxbuild/linux-amd64/graaljdk8/jdk11-cmp/bin/java \  
-XX:+UnlockExperimentalVMOptions \  
-XX:+UseJVMCICompiler -XX:-TieredCompilation \  
-XX:CompileOnly=Demo::workload \  
-Dgraal.Dump=:1 -Dgraal.PrintGraph=Network \  
-cp . Demo
```

```
int average(int a, int b) {  
    return (a + b) / 2;  
}
```



```
int average(int[] values) {  
    int sum = 0;  
    for (int n = 0; n < values.length; n++) {  
        sum += values[n];  
    }  
    return sum / values.length;  
}
```





From bytecode to machine code

Bytecode in...


```
int workload(int a, int b) {  
    return a + b;  
}
```

```
class HotSpotGraalCompiler implements JVMCICompiler {  
    CompilationRequestResult compileMethod(CompilationRequest request) {  
        System.err.println(request.getMethod().getName() + " bytecode: "  
            + Arrays.toString(request.getMethod().getCode()));  
        ...  
    }  
}
```

workload bytecode: [26, 27, 96, -84]

The bytecode parser...

```
graal - org.graalvm.compiler.nodes/src/org/graalvm/compiler/nodes/calc/AddNode.java - Eclipse
AddNode.java
25 import org.graalvm.compiler.core.common.type.ArithmeticOpTable;
40
41 @NodeInfo(shortName = "+")
42 public class AddNode extends BinaryArithmeticNode<Add> implements NarrowableArithmeticNode, BinaryCommutative<ValueNode> {
43
44     public static final NodeClass<AddNode> TYPE = NodeClass.create(AddNode.class);
45
46     public AddNode(ValueNode x, ValueNode y) {
47         this(TYPE, x, y);
48     }
49
50     protected AddNode(NodeClass<? extends AddNode> c, ValueNode x, ValueNode y) {
51         super(c, ArithmeticOpTable::getAdd, x, y);
52     }
53
54     public static ValueNode create(ValueNode x, ValueNode y) {
55         BinaryOp<Add> op = ArithmeticOpTable.forStamp(x.stamp()).getAdd();
56         Stamp stamp = op.foldStamp(x.stamp(), y.stamp());
57         ConstantNode tryConstantFold = tryConstantFold(op, x, y, stamp);
58         if (tryConstantFold != null) {
59             return tryConstantFold;
60         }
61         if (x.isConstant() && !y.isConstant()) {
62             return canonical(null, op, y, x);
63         } else {
64             return canonical(null, op, x, y);
65         }
66     }
67
68     private static ValueNode canonical(AddNode addNode, BinaryOp<Add> op, ValueNode forX, ValueNode forY) {
69         AddNode self = addNode;
```

graal - org.graalvm.compiler.nodes/src/org.graalvm/compiler/nodes/calc/AddNode.java - Eclipse

```
54 public static ValueNode create(ValueNode x, ValueNode y) {
55     BinaryOp<Add> op = ArithmeticOpTable.forStamp(x.stamp()).getAdd();
56     Stamp stamp = op.foldStamp(x.stamp(), y.stamp());
57     ConstantNode tryConstantFold = tryConstantFold(op, x, y, stamp);
58     if (tryConstantFold != null) {
59         return tryConstantFold;
60     }
61     if (x.isConstant() && !y.isConstant()) {
62         return canonical(null, op, y, x);
63     } else {
64         return canonical(null, op, x, y);
65     }
66 }
67
68 private static ValueNode canonical(AddNode addNode, BinaryOp<Add> op, ValueNode forX, ValueNode forY) {
69     AddNode self = addNode;
```

Call Hierarchy

Members calling 'create(ValueNode, ValueNode)' - in workspace

- add(StructuredGraph, ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.nodes.calc.BinaryArithmeticNode
- add(ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.nodes.calc.BinaryArithmeticNode
- canonical(MulNode, BinaryOp<Mul>, Stamp, ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.nodes.calc.MulNode (2 matches)
- findSynonym(ValueNode, ValueNode) : LogicNode - org.graalvm.compiler.nodes.calc.IntegerLessThanNode.LessThanOp (2 matches)
- genFloatAdd(ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.java.BytecodeParser
- genIntegerAdd(ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.java.BytecodeParser**
 - genArithmeticOp(JavaKind, int) : void - org.graalvm.compiler.java.BytecodeParser
 - processBytecode(int, int) : void - org.graalvm.compiler.java.BytecodeParser (4 matches)
 - genIncrement() : void - org.graalvm.compiler.java.BytecodeParser

org.graalvm.compiler.java.BytecodeParser.genIntegerAdd(ValueNode, ValueNode) : ValueNode - org.graalvm.compiler.java/src

```
graal - org.graalvm.compiler.java/src/org/graalvm/compiler/java/BytecodeParser.java - Eclipse
AddNode.java BytecodeParser.java
3416     genStoreIndexed(array, index, kind, value);
3417 }
3418
3419 private void genArithmeticOp(JavaKind kind, int opcode) {
3420     ValueNode y = frameState.pop(kind);
3421     ValueNode x = frameState.pop(kind);
3422     ValueNode v;
3423     switch (opcode) {
3424         case IADD:
3425         case LADD:
3426             v = genIntegerAdd(x, y);
3427             break;
3428         case FADD:
3429         case DADD:
3430             v = genFloatAdd(x, y);
3431             break;
3432         case ISUB:
3433         case LSUB:
3434             v = genIntegerSub(x, y);
3435             break;
3436         case FSUB:
3437         case DSUB:
3438             v = genFloatSub(x, y);
3439             break;
3440         case IMUL:
3441         case LMUL:
3442             v = genIntegerMul(x, y);
3443             break;
3444         case FMUL:
3445         case DMUL:
3446             v = genFloatMul(x, y);

```

```
private void genArithmeticOp(JavaKind kind, int opcode) {
    ValueNode y = frameState.pop(kind);
    ValueNode x = frameState.pop(kind);
    ValueNode v;
    switch (opcode) {
        ...
        case LADD:
            v = genIntegerAdd(x, y);
            break;
        ...
    }
    frameState.push(kind, append(v));
}
```

Emitting assembly...


```
void generate(Generator gen) {  
    gen.emitAdd(a, b);  
}
```

```
int workload(int a) {  
    return a + 1;  
}
```

```
void incl(Register dst) {
    int encode = prefixAndEncode(dst.encoding);
    emitByte(0xFF);
    emitByte(0xC0 | encode);
}

void emitByte(int b) {
    data.put((byte) (b & 0xFF));
}
```

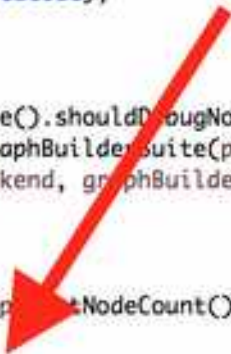
```
graal - org.graalvm.compiler.asm.amd64/src/org.graalvm.compiler.asm.amd64/AMD64Assembler.java - Eclipse
AMD64Assembler.java
1831
1832 public final void imull(Register dst, Register src, int value) {
1833     if (isByte(value)) {
1834         AMD64RMIOp.IMUL_SX.emit(this, DWORD, dst, src, value);
1835     } else {
1836         AMD64RMIOp.IMUL.emit(this, DWORD, dst, src, value);
1837     }
1838 }
1839
1840 protected final void indl(AMD64Address dst) {
1841     prefix(dst);
1842     emitByte(0xFF);
1843     emitOperandHelper(0, dst, 0);
1844 }
1845
1846 public void jcc(ConditionFlag cc, int jumpTarget, boolean forceDisp32) {
1847     int shortSize = 2;
1848     int longSize = 6;
1849     long disp = jumpTarget - position();
1850     if (!forceDisp32 && isByte(disp - shortSize)) {
1851         // 0111 tttt #8-bit disp
1852         emitByte(0x70 | cc.getValue());
1853         emitByte((int) ((disp - shortSize) & 0xFF));
1854     } else {
1855         // 0000 1111 1000 tttt #32-bit disp
1856         assert isInt(disp - longSize) : "must be 32bit offset (call4)";
1857         emitByte(0x0F);
1858         emitByte(0x80 | cc.getValue());
1859         emitInt((int) (disp - longSize));
1860     }
1861 }
```

```
graal - org.graalvm.compiler.asm/src/org.graalvm/compiler/asm/Buffer.java - Eclipse
AMD64Assembler.java Assembler.java Buffer.java
96     }
97   }
98
99   public void emitBytes(byte[] arr, int off, int len) {
100     ensureSize(data.position() + len);
101     data.put(arr, off, len);
102   }
103
104   public void emitByte(int b) {
105     assert NumUtil.isUByte(b) || NumUtil.isByte(b);
106     ensureSize(data.position() + 1);
107     data.put((byte) (b & 0xFF));
108   }
109
110   public void emitShort(int b) {
111     assert NumUtil.isUShort(b) || NumUtil.isShort(b);
112     ensureSize(data.position() + 2);
113     data.putShort((short) b);
114   }
115
116   public void emitInt(int b) {
117     ensureSize(data.position() + 4);
118     data.putInt(b);
119   }
120
121   public void emitLong(long b) {
122     ensureSize(data.position() + 8);
123     data.putLong(b);
124   }
125
126   public void emitBytes(byte[] arr, int pos) {
```

Machine code out...

```
class HotSpotGraalCompiler implements JVMCICompiler {  
    CompilationResult compileHelper(...) {  
        ...  
        System.err.println(method.getName() + " machine code: "  
            + Arrays.toString(result.getTargetCode()));  
        ...  
    }  
}
```

```
graal - org.graalvm.compiler.hotspot/src/org/graalvm/compiler/hotspot/HotSpotGraalCompiler.java - Eclipse
HotSpotGraalCompiler.java
174 Suites suites = getSuites(providers, options);
175 LIRSuites lirSuites = getLIRSuites(providers, options);
176 ProfilingInfo profilingInfo = useProfilingInfo ? method.getProfilingInfo(!isOSR, isOSR) : DefaultProfilingInfo.get
177 OptimisticOptimizations optimisticOpts = getOptimisticOpts(profilingInfo, options);
178
179 /*
180  * Cut off never executed code profiles if there is code, e.g. after the osr loop, that is never
181  * executed.
182  */
183 if (isOSR && !OnStackReplacementPhase.Options.DeoptAfterOSR.getValue(options)) {
184     optimisticOpts.remove(Optimization.RemoveNeverExecutedCode);
185 }
186
187 result.setEntryBCI(entryBCI);
188 boolean shouldDebugNonSafepoints = providers.getCodeCache().shouldDebugNonSafepoints();
189 PhaseSuite<HighTierContext> graphBuilderSuite = configGraphBuilderSuite(providers.getSuites().getDefaultGraphBuild
GraalCompiler.compileGraph(graph, method, providers, backend, graphBuilderSuite, optimisticOpts, profilingInfo, su
191
192 if (!isOSR && useProfilingInfo) {
193     ProfilingInfo profile = profilingInfo;
194     profile.setCompilerIRSize(StructuredGraph.class, graph.getNodeCount());
195 }
196
197 System.err.println(method.getName() + " machine code: " + Arrays.toString(result.getTargetCode()));
198
199 return result;
200 }
201
202 public CompilationResult compile(ResolvedJavaMethod method, int entryBCI, boolean useProfilingInfo, CompilationIdentif
203 StructuredGraph graph = createGraph(method, entryBCI, useProfilingInfo, compilationId, options, debug);
204
```




```
$ ./mxbuild/linux-amd64/graaljdk8/jdk11-cmp/bin/java \  
-XX:+UnlockExperimentalVMOptions \  
-XX:+UnlockDiagnosticVMOptions \  
-XX:+UseJVMCICompiler -XX:-TieredCompilation \  
-XX:CompileOnly=Demo::workload -XX:+PrintAssembly \  
-cp . Demo
```

workload machine code: [15, 31, 68, 0, 0, 3, -14, -117, -58, -123, 5, ...]

...

0x000000010f71cda0: nopl 0x0(%rax,%rax,1)

0x000000010f71cda5: add %edx,%esi ;*iadd {reexecute=0 rethrow=0 return_oop=0}
; - Demo::workload@2 (line 10)

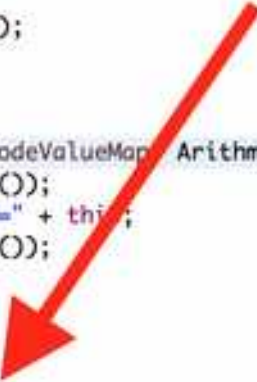
0x000000010f71cda7: mov %esi,%eax ;*ireturn {reexecute=0 rethrow=0 return_oop=0}
; - Demo::workload@3 (line 10)

0x000000010f71cda9: test %eax,-0xcba8da9(%rip) # 0x0000000102b74006
; {poll_return}

0x000000010f71cdaf: vzeroupper

0x000000010f71cdb2: retq

```
graal - org.graalvm.compiler.nodes/src/org/graalvm/compiler/nodes/calc/AddNode.java - Eclipse
AddNode.java
114     if (ret != this) {
115         return ret;
116     }
117
118     if (forX.isConstant() && !forY.isConstant()) {
119         // we try to swap and canonicalize
120         ValueNode improvement = canonical(tool, forY, forX);
121         if (improvement != this) {
122             return improvement;
123         }
124         // if this fails we only swap
125         return new AddNode(forY, forX);
126     }
127     BinaryOp<Add> op = getOp(forX, forY);
128     return canonical(this, op, forX, forY);
129 }
130
131 @Override
132 public void generate(NodeLIRBuilderTool nodeValueMap, ArithmeticLIRGeneratorTool gen) {
133     Value op1 = nodeValueMap.operand(getX());
134     assert op1 != null : getX() + ", this=" + this;
135     Value op2 = nodeValueMap.operand(getY());
136     if (shouldSwapInputs(nodeValueMap)) {
137         Value tmp = op1;
138         op1 = op2;
139         op2 = tmp;
140     }
141     nodeValueMap.setResult(this, gen.emitAdd(op1, op2, false));
142 }
143 }
144
```



```
workload mechine code: [15, 31, 68, 0, 0, 43, -14, -117, -58, -123, 5, ...]
0x0000000107f451a0: nopl    0x0(%rax,%rax,1)
0x0000000107f451a5: sub     %edx,%esi          ;*iadd {reexecute=0 rethrow=0 return_oop=0}
                                ; - Demo::workload@2 (line 10)

0x0000000107f451a7: mov     %esi,%eax       ;*ireturn {reexecute=0 rethrow=0 return_oop=0}
                                ; - Demo::workload@3 (line 10)

0x0000000107f451a9: test   %eax,-0x1db81a9(%rip)      # 0x000000010618d006
                                ; {poll_return}

0x0000000107f451af: vzeroupper
0x0000000107f451b2: retq
```

[26, 27, 96, -84] → [15, 31, 68, 0, 0, 43, -14, -117, -58, -123, 5, ...]

Optimisations

Canonicalisation

```
interface Phase {  
    void run(Graph graph);  
}
```



```
interface Node {  
    Node canonical();  
}
```

```
class NegateNode implements Node {
    Node canonical() {
        if (value instanceof NegateNode) {
            return ((NegateNode) value).getValue();
        } else {
            return this;
        }
    }
}
```

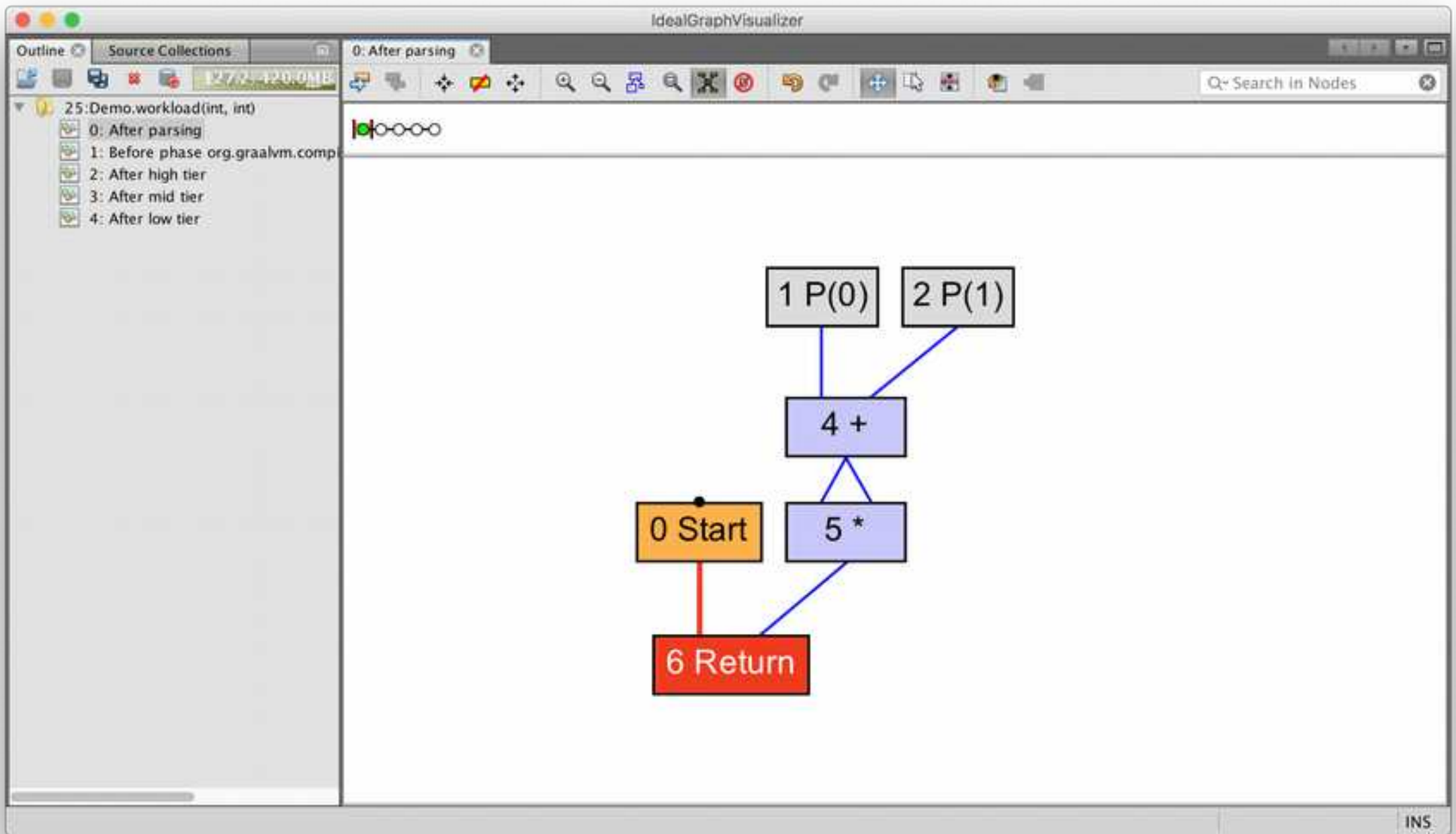
```
graal - org.graalvm.compiler.nodes/src/org/graalvm/compiler/nodes/calc/NegateNode.java - Eclipse
NegateNode.java
58     }
59
60     @Override
61     public ValueNode canonical(CanonicalizerTool tool, ValueNode forValue) {
62         ValueNode synonym = findSynonym(forValue, getOp(forValue));
63         if (synonym != null) {
64             return synonym;
65         }
66         return this;
67     }
68
69     protected static ValueNode findSynonym(ValueNode forValue) {
70         ArithmeticOpTable.UnaryOp<Neg> negOp = ArithmeticOpTable.forStamp(forValue.stamp()).getNeg();
71         ValueNode synonym = UnaryArithmeticNode.findSynonym(forValue, negOp);
72         if (synonym != null) {
73             return synonym;
74         }
75         if (forValue instanceof NegateNode) {
76             return ((NegateNode) forValue).getValue();
77         }
78         if (forValue instanceof SubNode && !(forValue.stamp() instanceof FloatStamp)) {
79             SubNode sub = (SubNode) forValue;
80             return SubNode.create(sub.getY(), sub.getX());
81         }
82         return null;
83     }
84
85     @Override
86     public void generate(NodeLIRBuilderTool nodeValueMap, ArithmeticLIRGeneratorTool gen) {
87         nodeValueMap.setResult(this, gen.emitNegate(nodeValueMap.operand(getValue())));
88     }

```

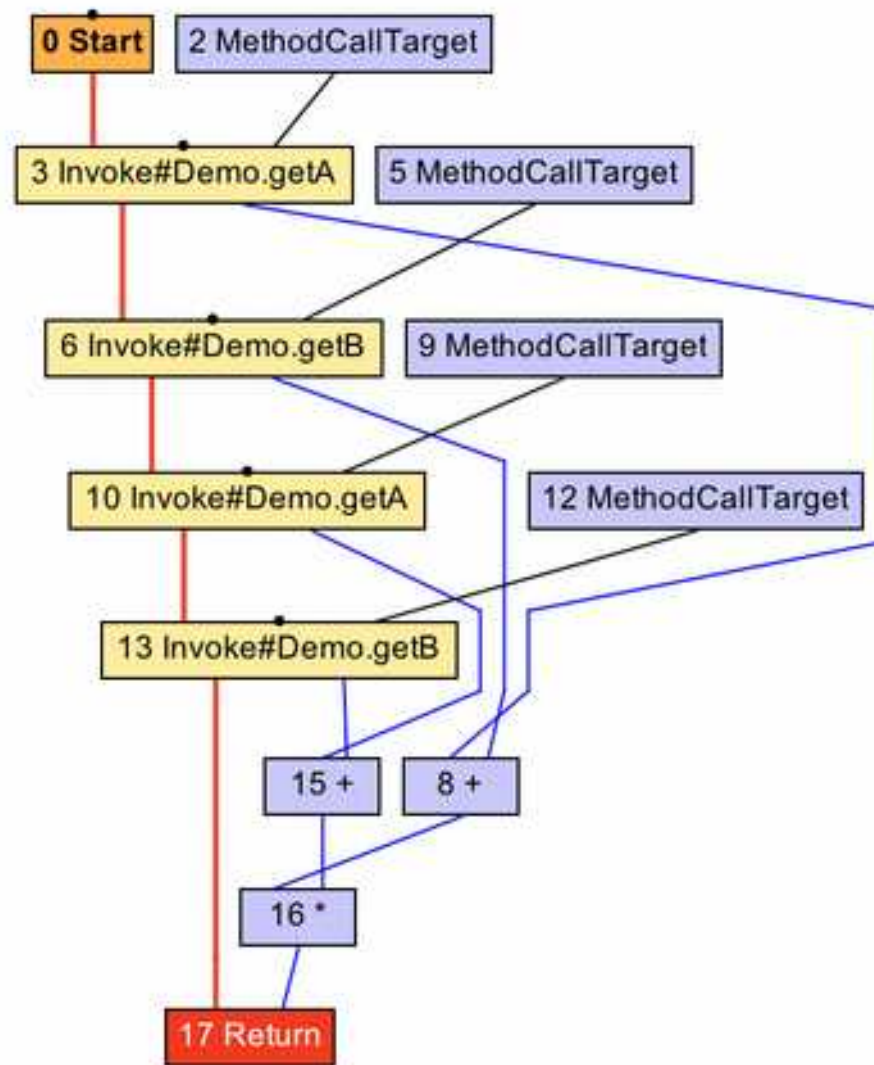
Global value numbering

```
int workload(int a, int b) {  
    return (a + b) * (a + b);  
}
```

```
graal - org.graalvm.compiler.phases.common/src/org/graalvm/compiler/phases/common/CanonicalizerPhase.java - Eclipse
CanonicalizerPhase.java
274     }
275     valueNode.usages().forEach(workList::add);
276     }
277 }
278 return false;
279 }
280
281 public boolean tryGlobalValueNumbering(Node node, NodeClass<?> nodeClass) {
282     if (nodeClass.valueNumberable()) {
283         Node newNode = node.graph().findDuplicate(node);
284         if (newNode != null) {
285             assert !(node instanceof FixedNode || newNode instanceof FixedNode);
286             node.replaceAtUsagesAndDelete(newNode);
287             COUNTER_GLOBAL_VALUE_NUMBERING_HITS.increment(debug);
288             debug.log("GVN applied and new node is %1s", newNode);
289             return true;
290         }
291     }
292     return false;
293 }
294
295 private AutoCloseable getCanonicalizableContractAssertion(Node node) {
296     boolean needsAssertion = false;
297     assert (needsAssertion = true) == true;
298     if (needsAssertion) {
299         Mark mark = node.graph().getMark();
300         return () -> {
301             assert mark.equals(node.graph().getMark()) : "new node created while canonicalizing " + node.getClass().getName();
302             node.graph().getNewNodes(mark).snapshot();
303         };
304     } else {
```



```
int workload() {  
    return (getA() + getB()) * (getA() + getB());  
}
```

Lock coarsening

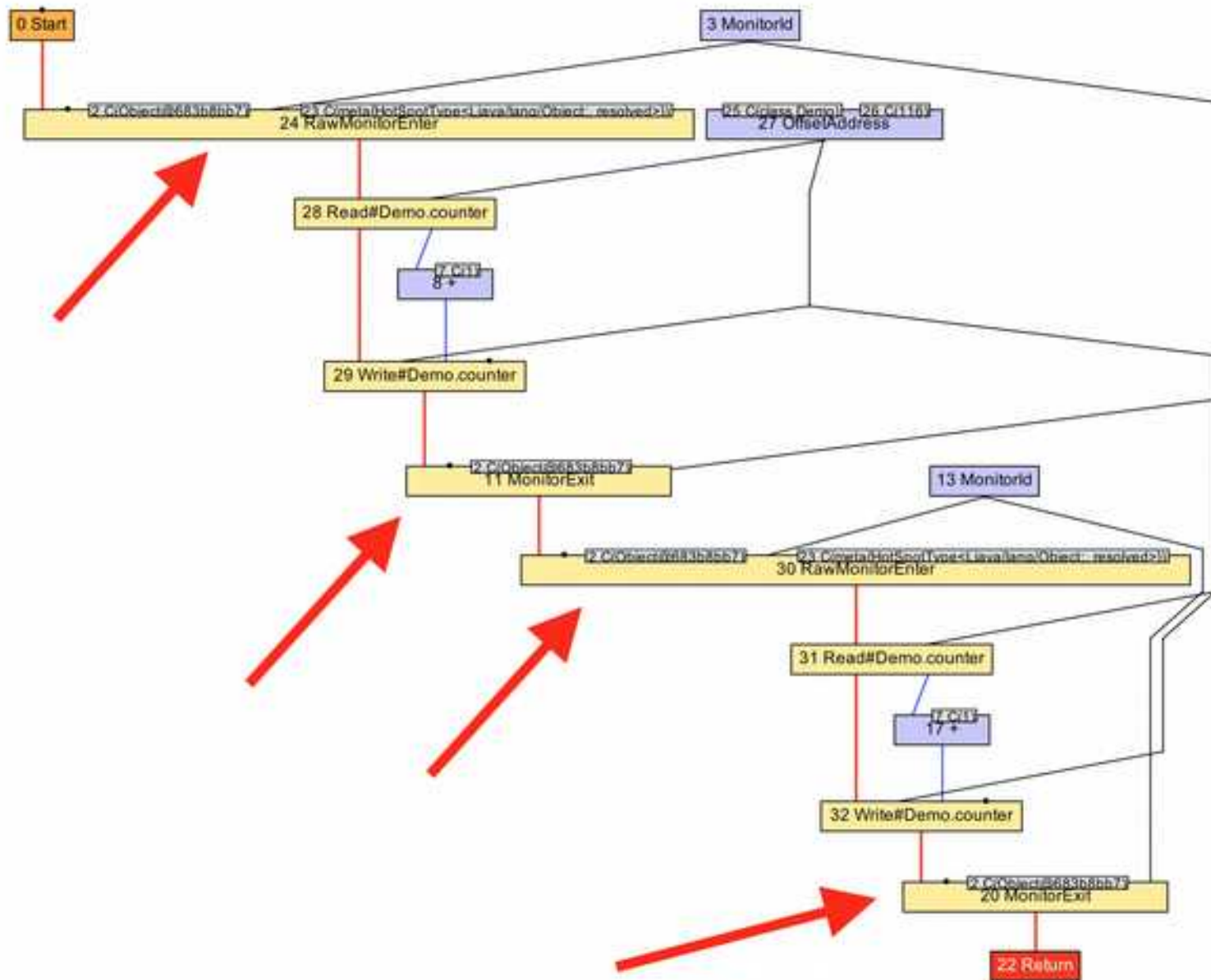
```
void workload() {  
    synchronized (monitor) {  
        counter++;  
    }  
    synchronized (monitor) {  
        counter++;  
    }  
}
```

```
void workload() {  
    monitor.enter();  
    counter++;  
    monitor.exit();  
    monitor.enter();  
    counter++;  
    monitor.exit();  
}
```

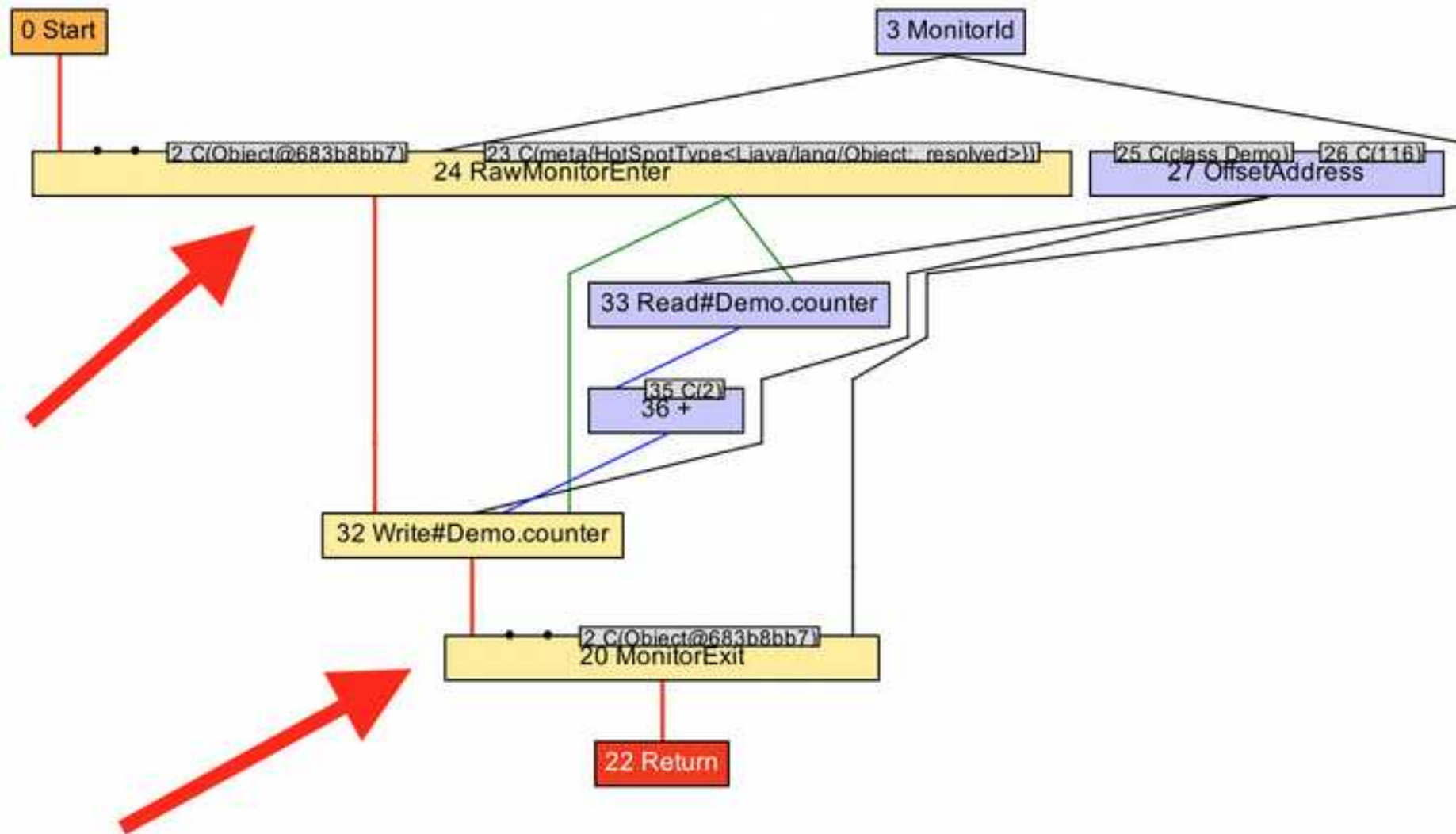
```
void workload() {  
    monitor.enter();  
    counter++;  
    counter++;  
    monitor.exit();  
}
```

```
void run(StructuredGraph graph) {  
    for (monitorExitNode monitorExitNode : graph.getNodes(monitorExitNode.class)) {  
        FixedNode next = monitorExitNode.next();  
        if (next instanceof monitorEnterNode) {  
            AccessmonitorNode monitorEnterNode = (AccessmonitorNode) next;  
            if (monitorEnterNode.object() == monitorExitNode.object()) {  
                monitorExitNode.remove();  
                monitorEnterNode.remove();  
            }  
        }  
    }  
}
```

```
graal - org.graalvm.compiler.phases.common/src/org.graalvm/compiler/phases/common/LockEliminationPhase.java - Eclipse
LockEliminationPhase.java
35
36 public class LockEliminationPhase extends Phase {
37
38     @Override
39     protected void run(StructuredGraph graph) {
40         for (MonitorExitNode monitorExitNode : graph.getNodes(MonitorExitNode.TYPE)) {
41             FixedNode next = monitorExitNode.next();
42             if ((next instanceof MonitorEnterNode || next instanceof RawMonitorEnterNode)) {
43                 // should never happen, OSR monitor enters are always direct successors of the graph
44                 // start
45                 assert !(next instanceof OSRMonitorEnterNode);
46                 AccessMonitorNode monitorEnterNode = (AccessMonitorNode) next;
47                 if (GraphUtil.unproxify(monitorEnterNode.object()) == GraphUtil.unproxify(monitorExitNode.object())) {
48                     /*
49                      * We've coarsened the lock so use the same monitor id for the whole region,
50                      * otherwise the monitor operations appear to be unrelated.
51                      */
52                     MonitorIdNode enterId = monitorEnterNode.getMonitorId();
53                     MonitorIdNode exitId = monitorExitNode.getMonitorId();
54                     if (enterId != exitId) {
55                         enterId.replaceAndDelete(exitId);
56                     }
57                     GraphUtil.removeFixedWithUnusedInputs(monitorEnterNode);
58                     GraphUtil.removeFixedWithUnusedInputs(monitorExitNode);
59                 }
60             }
61         }
62     }
63 }
64
```

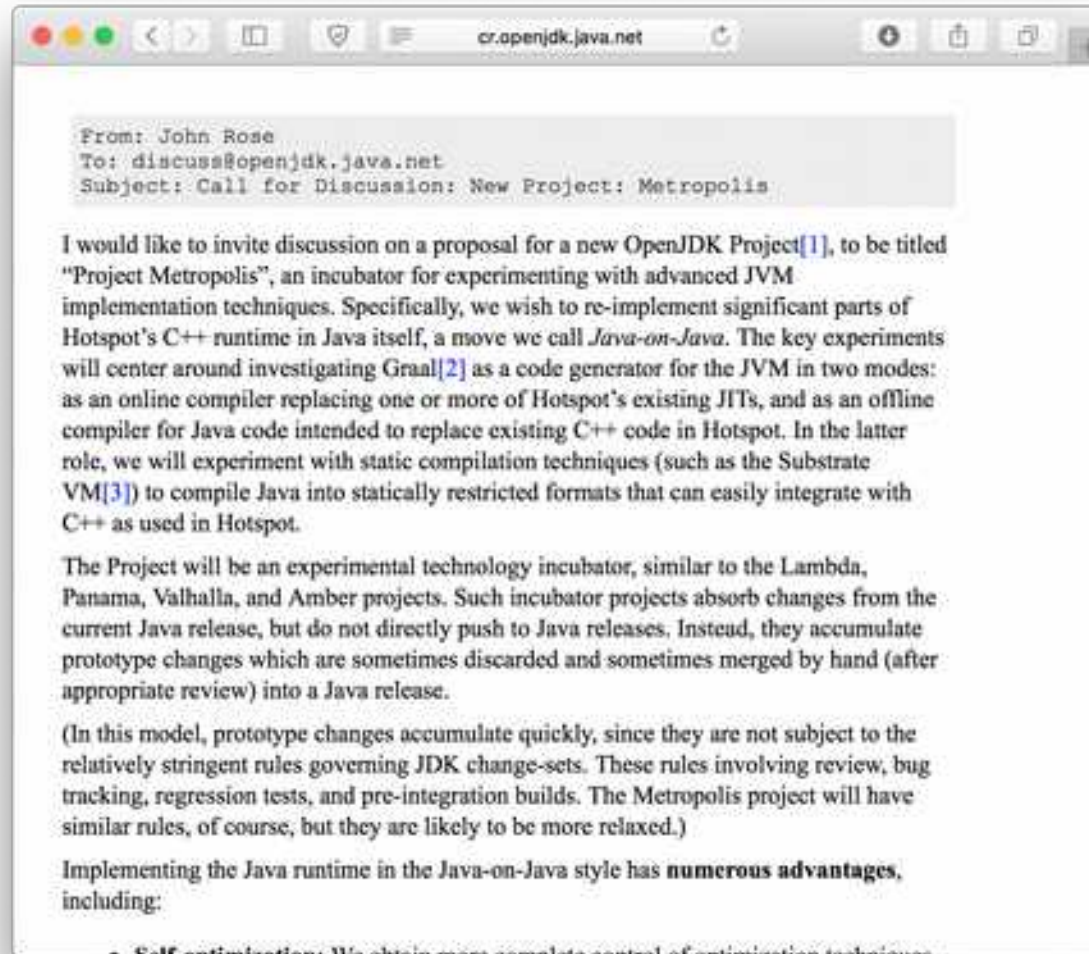



```
void workload() {  
    monitor.enter();  
    counter += 2;  
    monitor.exit();  
}
```



A final tier compiler

`-XX:+UseJVMCICompiler`



<http://cr.openjdk.java.net/~jrose/metropolis/Metropolis-Proposal.html>

Your own specific optimisations

Ahead of time compilation

```
$ javac Hello.java
```

```
$ graalvm-0.28.2/bin/native-image Hello
```

```
  classlist:      966.44 ms  
    (cap):      804.46 ms  
    setup:     1,514.31 ms  
(typeflow):     2,580.70 ms  
(objects):      719.04 ms  
(features):      16.27 ms  
  analysis:     3,422.58 ms  
  universe:      262.09 ms  
    (parse):     528.44 ms  
    (inline):   1,259.94 ms  
(compile):     6,716.20 ms  
  compile:     8,817.97 ms  
    image:     1,070.29 ms  
  debuginfo:      672.64 ms  
    write:     1,797.45 ms  
  [total]:    17,907.56 ms
```



```
$ ls -lh hello
-rwxr-xr-x 1 chrisseaton staff 6.6M 4 Oct 18:35 hello

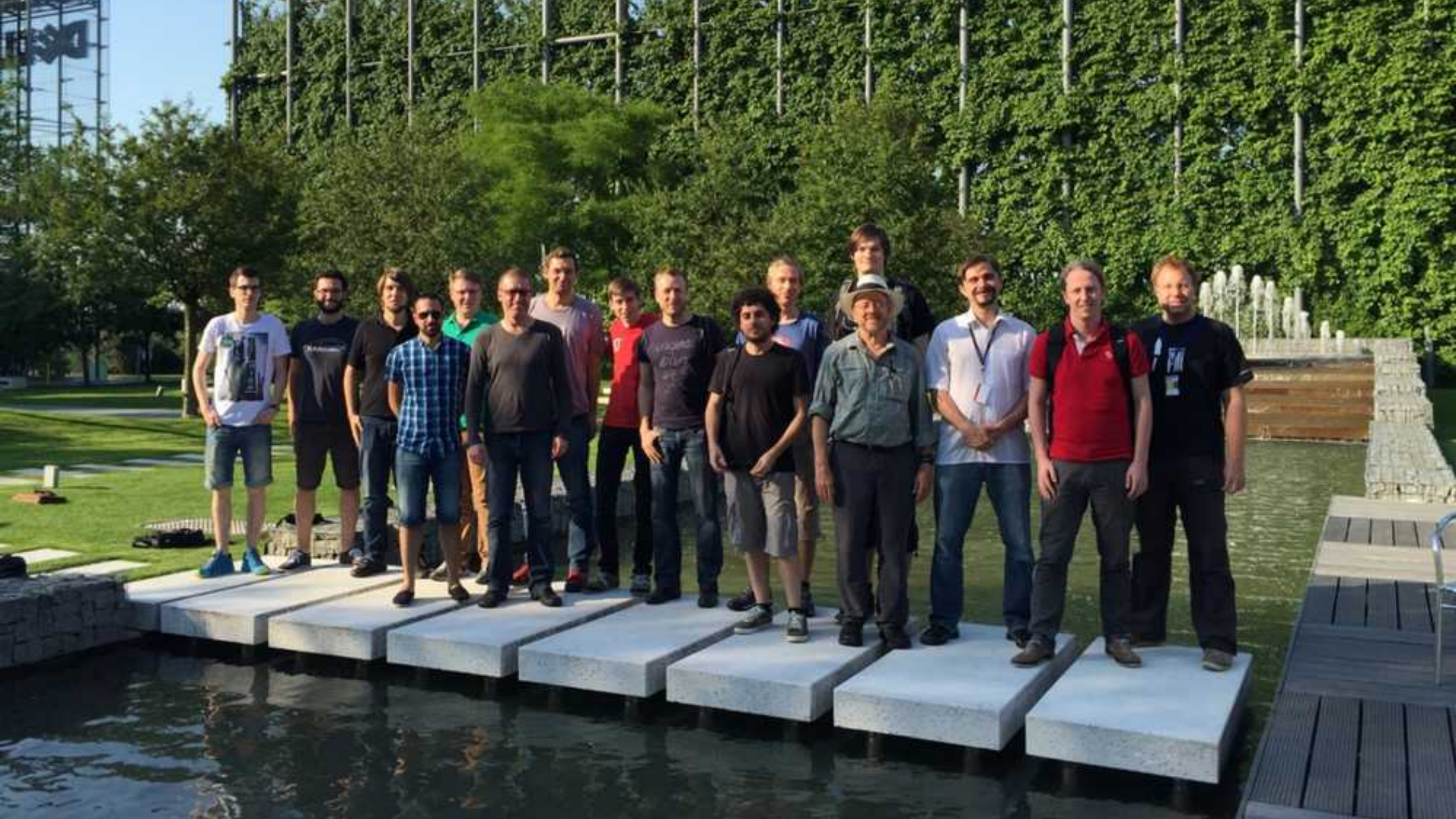
$ file ./hello
./hellojava: Mach-O 64-bit executable x86_64

$ time ./hello
Hello!

real    0m0.010s
user    0m0.003s
sys     0m0.003s
```

Truffle

Summary



Team

Oracle

Florian Angerer
Danilo Ansaloni
Stefan Anzinger
Martin Balin
Cosmin Basca
Daniele Bonetta
Dušan Bálek
Matthias Brantner
Lucas Braun
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Svatopluk Dědic
Martin Entlicher
Pit Fender
Francois Farquet
Brandon Fish
Matthias Grimmer
Christian Häubl
Peter Hofer
Bastian Hossbach
Christian Humer
Tomáš Hůrka
Mick Jordan

Oracle (continued)

Vojin Jovanovic
Anantha Kandukuri
Harshad Kasture
Cansu Kaynak
Peter Kessler
Duncan MacGregor
Jiří Maršík
Kevin Menard
Miloslav Metelka
Tomáš Myšík
Petr Pišl
Oleg Pliss
Jakub Podlešák
Aleksandar Prokopec
Tom Rodriguez
Roland Schatz
Benjamin Schlegel
Chris Seaton
Jiří Sedláček
Doug Simon
Štěpán Šindelář
Zbyněk Šlajchrt
Boris Spasojević
Lukas Stadler
Codrut Stancu

Oracle (continued)

Jan Štola
Tomáš Stupka
Farhan Tauheed
Jaroslav Tulach
Alexander Ulrich
Michael Van De Vanter
Aleksandar Vitorovic
Christian Wimmer
Christian Wirth
Paul Wögerer
Mario Wolczko
Andreas Wöß
Thomas Würthinger
Tomáš Zezula
Yudi Zheng

Red Hat

Andrew Dinn
Andrew Haley

Intel

Michael Berg

Twitter

Chris Thalinger

Oracle Interns

Brian Belleville
Ondrej Douda
Juan Fumero
Miguel Garcia
Hugo Guiroux
Shams Imam
Berkin Ilbeyi
Hugo Kapp
Alexey Karyakin
Stephen Kell
Andreas Kunft
Volker Lanting
Gero Leinemann
Julian Lettner
Joe Nash
Tristan Overney
Aleksandar Pejovic
David Piorkowski
Philipp Riedmann
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Oracle Alumni

Erik Eckstein
Michael Haupt
Christos Kotselidis
David Leibs
Adam Welc
Till Westmann

JKU Linz

Hanspeter Mössenböck
Benoit Daloze
Josef Eisl
Thomas Feichtinger
Josef Haider
Christian Huber
David Leopoldseder
Stefan Marr
Manuel Rigger
Stefan Rumzucker
Bernhard Urban

TU Berlin:

Volker Markl
Andreas Kunft
Jens Meiners
Tilmann Rabl

University of Edinburgh

Christophe Dubach
Juan José Fumero Alfonso
Ranjeet Singh
Toomas Remmelg

LaBRI

Floréal Morandat

University of California, Irvine

Michael Franz
Yeoul Na
Mohaned Qunaibit
Gulfem Savrun Yeniceri
Wei Zhang

Purdue University

Jan Vitek
Tomas Kalibera
Petr Maj
Lei Zhao

T. U. Dortmund

Peter Marwedel
Helena Kotthaus
Ingo Korb

University of California, Davis

Duncan Temple Lang
Nicholas Ulle

University of Lugano, Switzerland

Walter Binder
Sun Haiyang

Safe Harbor Statement

The preceding is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

Q&A

@JaroslavTulach
Oracle Labs

Integrated Cloud

Applications & Platform Services

ORACLE®