Scala ❤ Graal

@flaviowbrasil
Graal workshop
CGO 2019

Agenda

1 #Deploy

Why Twitter decided to use Graal

2 #Profile

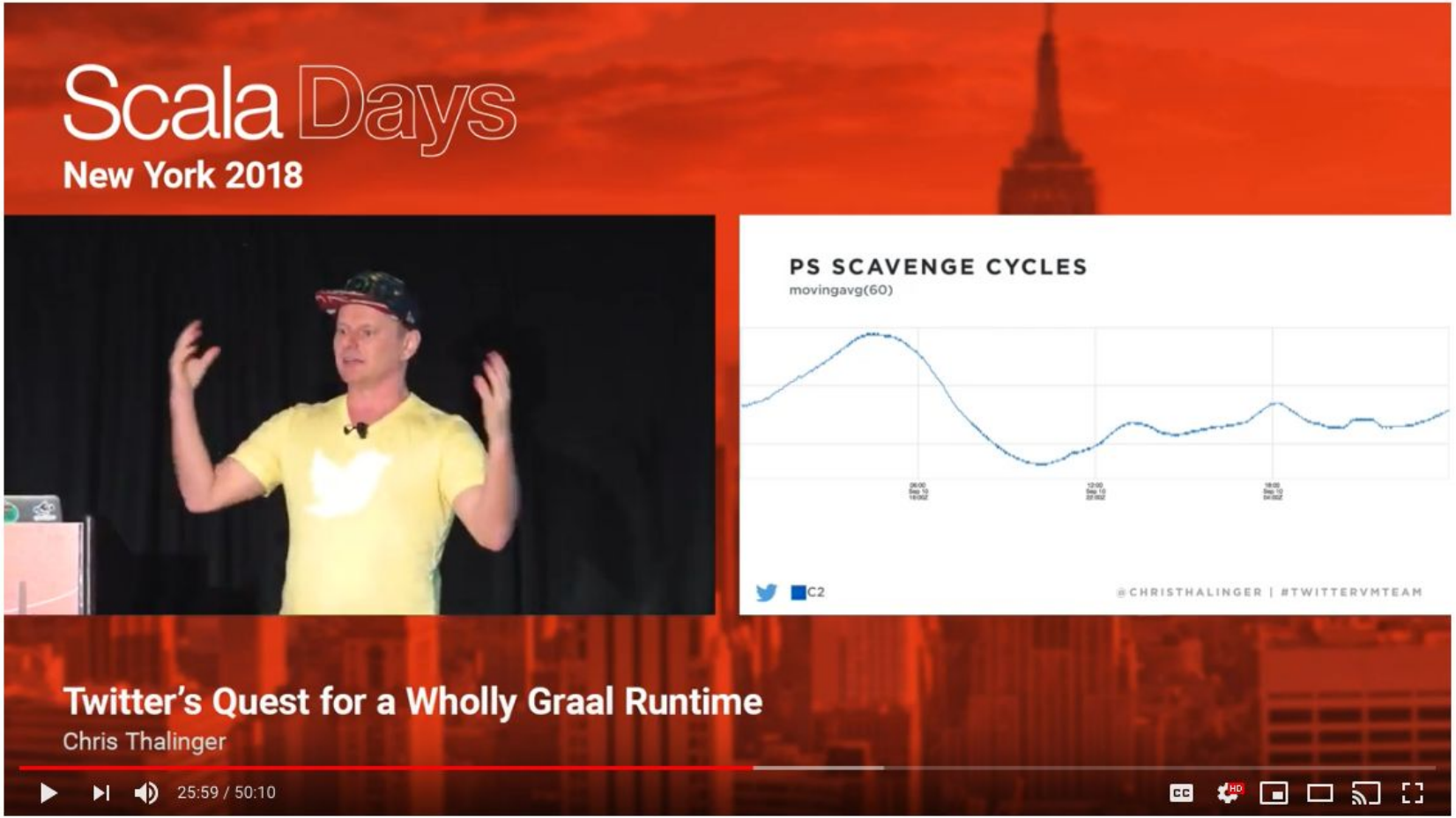
Performance challenges of Scala codebases

3 #Optimize

Optimization efforts by the #TwitterVMTeam

#Deploy

Why Twitter decided to use Graal

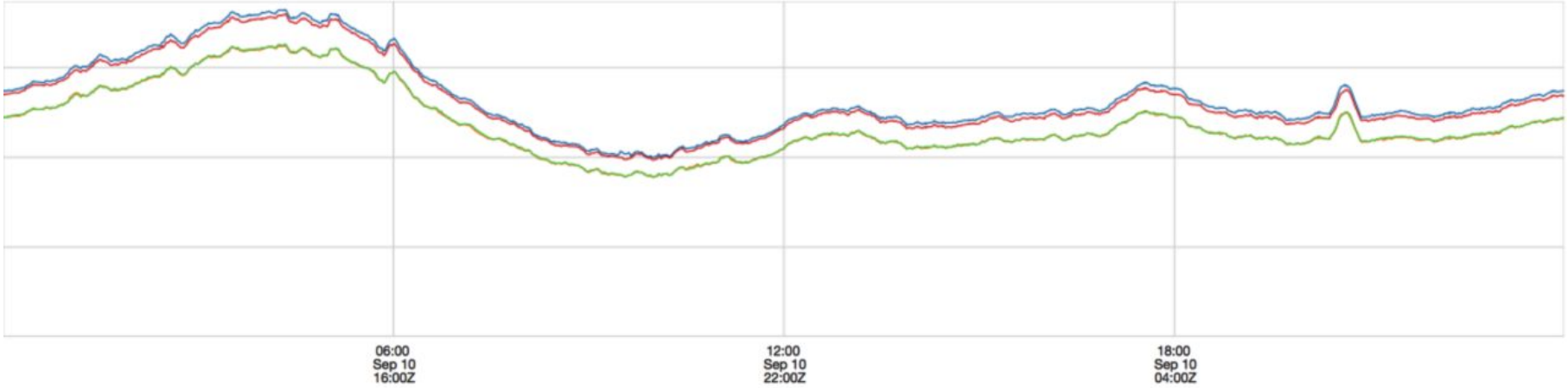


Twitter's Quest for a Wholly Graal Runtime by Chris Thalinger

Why Graal

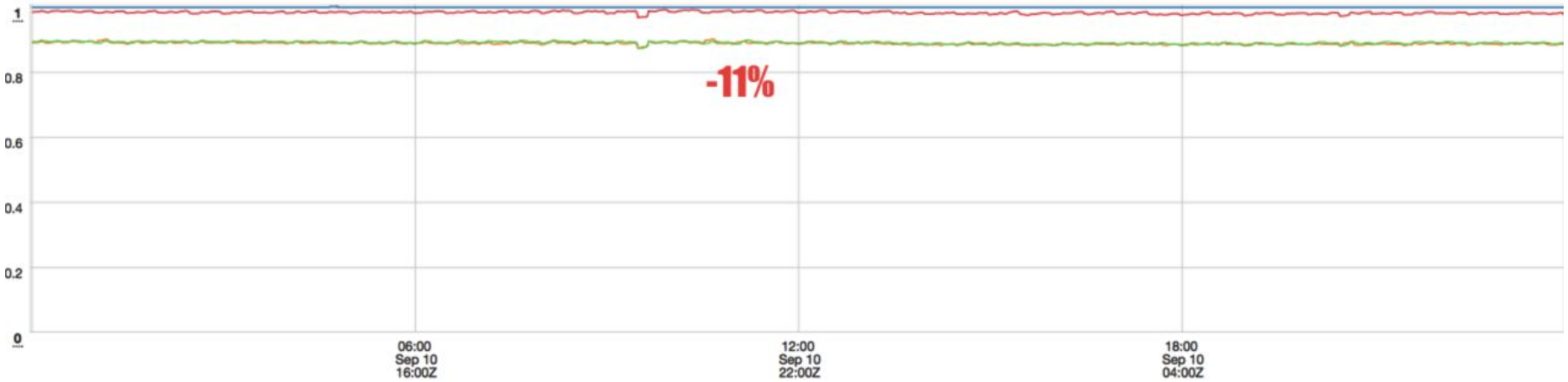
Performance
Ease of change

USER CPU TIME



C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9

oracle / graal

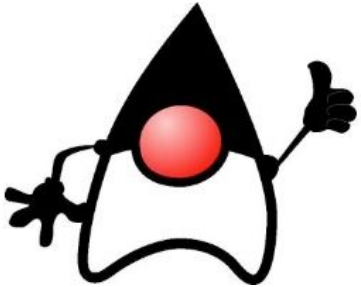
Code | Issues 241 | Pull requests 24 | Insights

GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

polyglot vm java javascript python r ruby c

Java 91.3% C 6.2% Python 1.2%

Branch: master New pull request



#Profile

Performance challenges of Scala codebases

Scala is not Java

(but is Java becoming Scala)

Advanced Hotspots Hotspots viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platfo

Grouping: Function / Call Stack

Function / Call Stack	CPU Time	Instructions Retired	CPI Rate	CPU Frequency Ratio
itable stub	159.368ms	310,800,000	1.061	0.987
com::twitter::util::Promise::continue	68.158ms	149,100,000	1.042	1.088
com::twitter::util::Promise$Transformer::apply	40.093ms	168,000,000	0.450	0.900
io::netty::handler::codec::MessageToMessage	30.069ms	25,200,000	1.667	0.667
com::twitter::finagle::http::DefaultHeaderMap::	28.065ms	111,300,000	0.660	1.250

Composable APIs

Scala’s performance nightmare

```scala
sealed trait IO[T] {
  def map[U](f: T => U): IO[U] = IO.Map(this, f)
}
```

```scala
sealed trait IO[T] {
  def map[U](f: T => U): IO[U] = IO.Map(this, f)
}

object IO {
  case class Map[T, U](io: IO[T], f: T => U) extends IO[U]

  case class Value[T](v: T) extends IO[T]
  def value[T](v: T): IO[T] = Value(v)
}
```

```scala
sealed trait IO[T] {
  def map[U](f: T => U): IO[U] = IO.Map(this, f)
}

object IO {
  case class Map[T, U](io: IO[T], f: T => U) extends IO[U]

  case class Value[T](v: T) extends IO[T]
  def value[T](v: T): IO[T] = Value(v)
}

def run[U](io: IO[U]): U =
  io match {
    case io: Value[U]  => io.v
    case io: Map[_, U] => io.f(run(io.io))
  }
```

```scala
def plus10(n: Int): IO[Int] =
  IO.value(n).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1)
```

```
@Benchmark
def inlined(): Int = {
  run(plus10(20))
}
```

```scala
@Benchmark
def inlined(): Int = {
  run(plus10(20))
}

@Benchmark
def notInlined(): Int = {
  runNotInlined(plus10(20))
}
```

```scala
@Benchmark
def inlined(): Int = {
  run(plus10(20))
}

@Benchmark
def notInlined(): Int = {
  runNotInlined(plus10(20))
}

@CompilerControl(CompilerControl.Mode.DONT_INLINE)
def runNotInlined[T](io: IO[T]): T = {
  run(io)
}
```

```
Benchmark                Mode  Cnt         Score         Error  Units
IOExample.inlined       thrpt    6  22836156.738 ± 6135579.777  ops/s
IOExample.notInlined    thrpt    6   9066384.269 ± 1293949.123  ops/s
```

Composable APIs are embedded languages

Interpreted languages are difficult to optimize

#Optimize

Optimization efforts by the #TwitterVMTeam

First target: Future

```
f.map(_ + 1).map(_ + 1).map(_ + 1)
```

```
f.map(_ + 1).map(_ + 1).map(_ + 1)
```

```
def map[U](f: T => U): Future[U] = {
  val result = new Promise[U]()
  this.addToQueue(Map(f, result))
  result
}
```

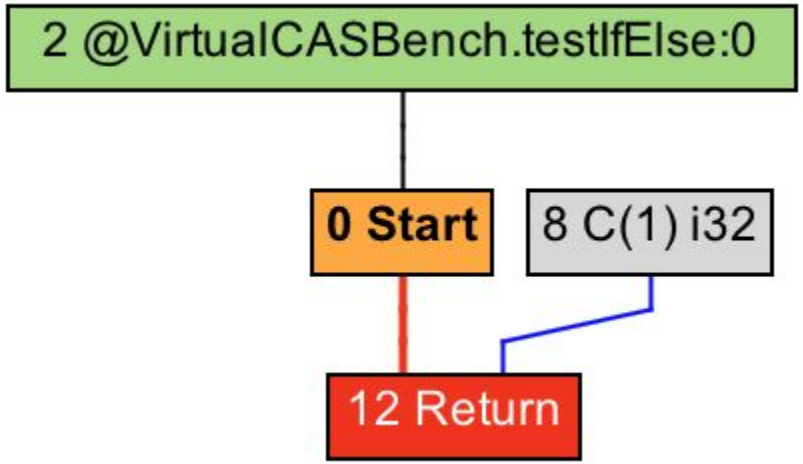
```java
public class VirtualCASBench {

  public static final long valueOffset = UnsafeUtil.fieldOffset(TestClass.class, "value");

  private static class TestClass {
    public volatile int value;

    public TestClass(int value) {
      this.value = value;
    }
  }
```

```java
public class VirtualCASBench {

  public static final long valueOffset = UnsafeUtil.fieldOffset(TestClass.class, "value");

  private static class TestClass {
    public volatile int value;

    public TestClass(int value) {
      this.value = value;
    }
  }

  @Benchmark
  public boolean testUnsafe() {
    TestClass t = new TestClass(0);
    return UnsafeUtil.unsafe.compareAndSwapInt(t, valueOffset, 0, 1);
  }
```

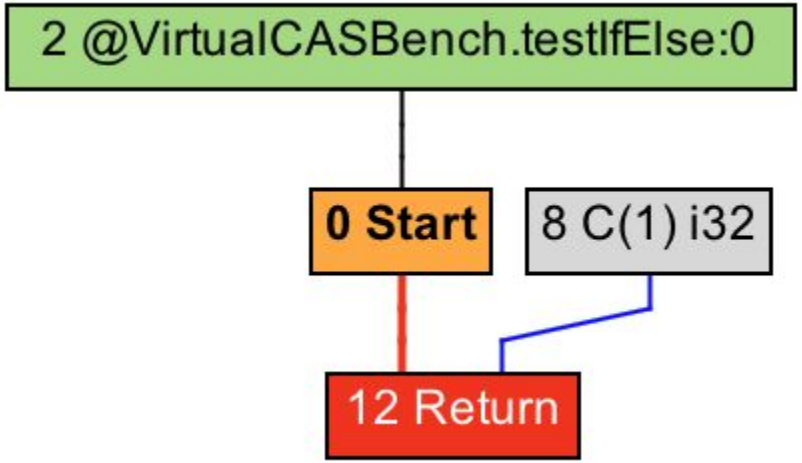
```java
public class VirtualCASBench {

  public static final long valueOffset = UnsafeUtil.fieldOffset(TestClass.class, "value");

  private static class TestClass {
    public volatile int value;

    public TestClass(int value) {
      this.value = value;
    }
  }

  @Benchmark
  public boolean testUnsafe() {
    TestClass t = new TestClass(0);
    return UnsafeUtil.unsafe.compareAndSwapInt(t, valueOffset, 0, 1);
  }

  @Benchmark
  public boolean testIfElse() {
    TestClass t = new TestClass(0);
    synchronized (t) {
      if (t.value != 0)
        return false;
      else {
        t.value = 1;
        return true;
      }
    }
  }
}
```

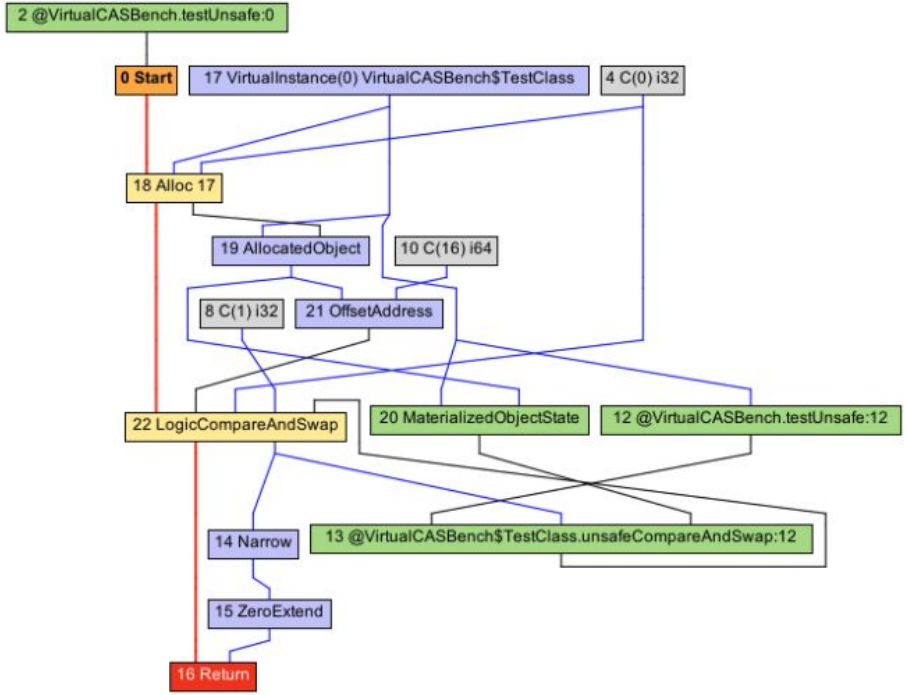
```
Benchmark                                        Mode  Cnt          Score          Error   Units
VirtualCASBench.testUnsafe                      thrpt   15   84931922.775 ±  2882353.049   ops/s
VirtualCASBench.testUnsafe:·gc.alloc.rate.norm  thrpt   15         16.125 ±        0.268    B/op
VirtualCASBench.testIfElse                      thrpt   15  320723943.152 ± 11305998.191   ops/s
VirtualCASBench.testIfElse:·gc.alloc.rate.norm  thrpt   15         ≈ 10⁻⁵                    B/op
```

```
Benchmark                                        Mode  Cnt          Score          Error   Units
VirtualCASBench.testUnsafe                      thrpt   15   84931922.775 ±  2882353.049   ops/s
VirtualCASBench.testUnsafe:·gc.alloc.rate.norm  thrpt   15         16.125 ±        0.268    B/op
VirtualCASBench.testIfElse                      thrpt   15  320723943.152 ± 11305998.191   ops/s
VirtualCASBench.testIfElse:·gc.alloc.rate.norm  thrpt   15         ≈ 10⁻⁵                    B/op
```



Benchmark	Mode	Cnt	Score		Error	Units
VirtualCASBench.testUnsafe	thrpt	15	84931922.775	±	2882353.049	ops/s
VirtualCASBench.testUnsafe:·gc.alloc.rate.norm	thrpt	15	16.125	±	0.268	B/op
VirtualCASBench.testIfElse	thrpt	15	320723943.152	±	11305998.191	ops/s
VirtualCASBench.testIfElse:·gc.alloc.rate.norm	thrpt	15	$\approx 10^{-5}$			B/op



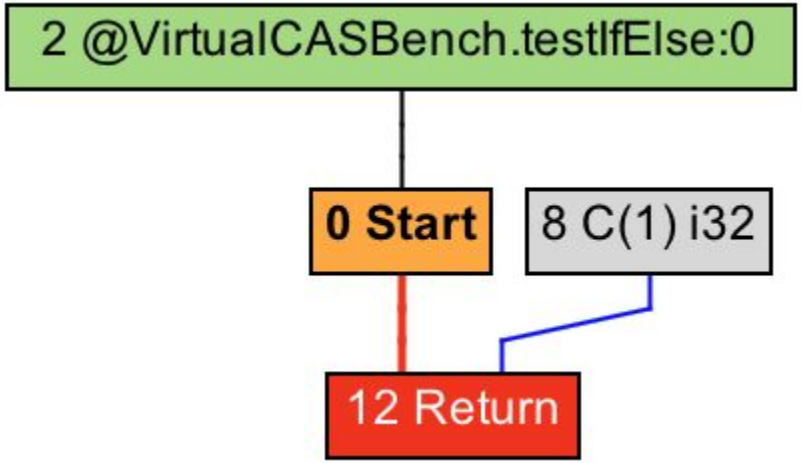


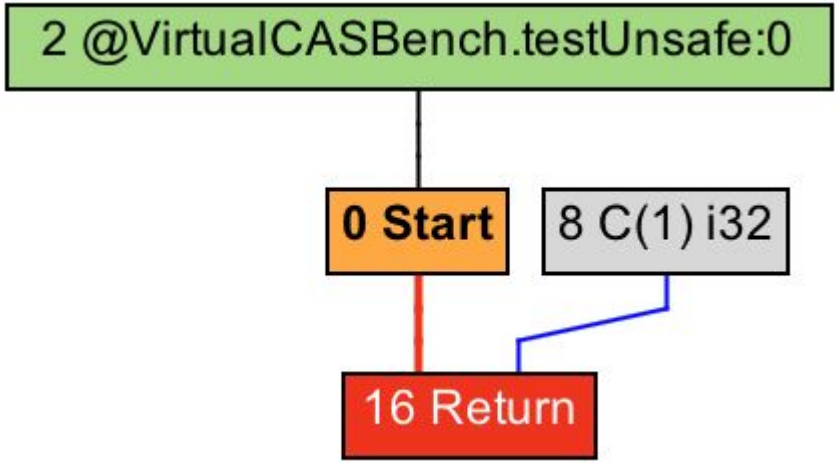
CAS Virtualization

https://github.com/oracle/graal/pull/636

```
Benchmark                                        Mode  Cnt          Score          Error   Units
VirtualCASBench.testUnsafe                      thrpt   15  299973950.364 ± 15236548.311   ops/s
VirtualCASBench.testUnsafe:·gc.alloc.rate.norm  thrpt   15         ≈ 10⁻⁵                    B/op
VirtualCASBench.testIfElse                      thrpt   15  305482126.736 ± 17237191.819   ops/s
VirtualCASBench.testIfElse:·gc.alloc.rate.norm  thrpt   15          0.001 ±        0.003    B/op
```

```
Benchmark                                        Mode  Cnt          Score          Error   Units
VirtualCASBench.testUnsafe                      thrpt   15  299973950.364 ± 15236548.311   ops/s
VirtualCASBench.testUnsafe:·gc.alloc.rate.norm  thrpt   15          ≈ 10⁻⁵                   B/op
VirtualCASBench.testIfElse                      thrpt   15  305482126.736 ± 17237191.819   ops/s
VirtualCASBench.testIfElse:·gc.alloc.rate.norm  thrpt   15          0.001 ±        0.003    B/op
```



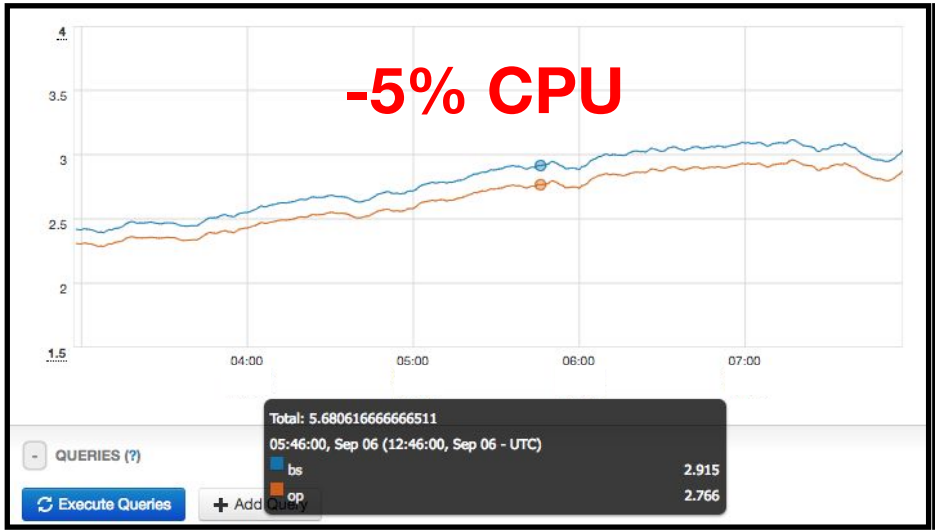


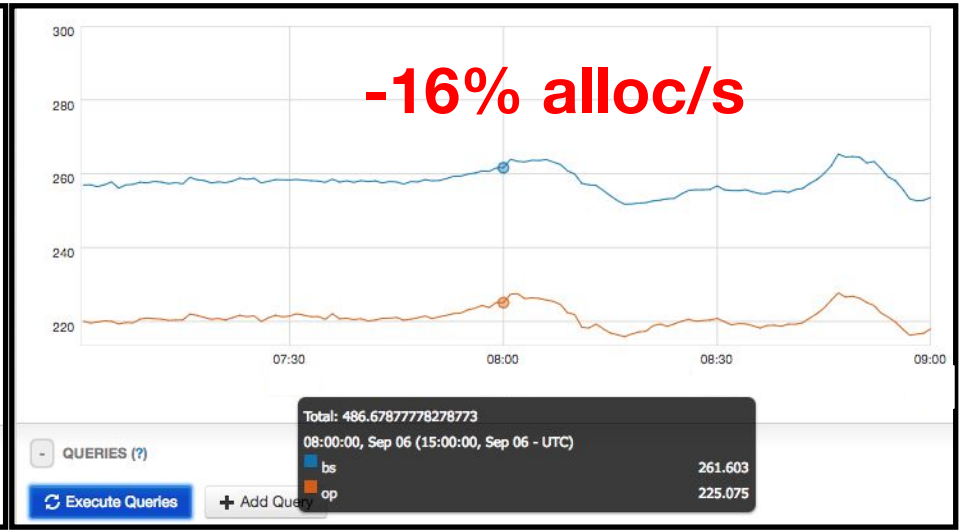
```
f.map(_ + 1).map(_ + 1).map(_ + 1)
```

```
def map[U](f: T => U): Future[U] = {
  val result = new Promise[U]()
  this.addToQueue(Map(f, result))
  result
}
```

Q4/2018
Optimizations results

CAS Optimizations
(regular load)





* Includes related code optimizations
https://github.com/twitter/util/commit/3245a8

All optimizations
(stress test)

CAS Virtualization improvement	14.97%
ABDecider Optimization	17.49%
Both optimizations	**29.84%**

Second target: itable stub calls

https://github.com/oracle/graal/issues/893
(work in progress)

Data-driven approach

```
-- Data collected through manual instrumentation
select * from invokeinterfaces;
```

base Console: postgres@localhost [2]

Output | postgres.public.invokeinterfaces

1-500 of 501+ | Tx: Auto

	invokeid	targetmethod	profiledtype	superclass	superclassmethod	nu
1	514	$plus$eq	SetBuilder	Object	$plus$eq	
2	514	$plus$eq	MapBuilder	Object	$plus$eq	
3	514	$plus$eq	HashMap	AbstractMap	$plus$eq	
4	514	$plus$eq	Queue	MutableList	$plus$eq	
5	2375	$plus	Set$Set1	AbstractSet	$plus	
6	2375	$plus	Set$Set2	AbstractSet	$plus	
7	2375	$plus	Set$Set3	AbstractSet	$plus	
8	2375	$plus	HashSet$HashTrieSet	HashSet	$plus	
9	2375	$plus	Set$Set4	AbstractSet	$plus	
10	3179	$plus	HashMap$HashMap1	HashMap	$plus	
11	3179	$plus	Map$Map4	AbstractMap	$plus	
12	143	Product.productElement	Tracer	Object	Tracer.productElement	
13	178	GenMapLike.get	HashMap$HashTrieMap	HashMap	HashMap.get	
14	178	GenMapLike.get	Map$Map2	AbstractMap	Map$Map2.get	
15	178	GenMapLike.get	Map$Map1	AbstractMap	Map$Map1.get	
16	178	GenMapLike.get	Map$EmptyMap$	AbstractMap	Map$EmptyMap$.get	
17	178	GenMapLike.get	Map$Map3	AbstractMap	Map$Map3.get	
18	178	GenMapLike.get	Map$Map4	AbstractMap	Map$Map4.get	
19	376	Iterator.next	AbstractList$Itr	Object	AbstractList$Itr.next	
20	406	Sink.end	ReduceOps$8ReducingSink	Object	Sink.end	
21	0	Function1.apply	ClassPath$$anonfun$browseJar$4	AbstractFunction1	ClassPath$$anonfun$bro…	
22	1	Iterator.next	Wrappers$JEnumerationWrapper	AbstractIterator	Wrappers$JEnumerationW…	
23	1	Iterator.next	Iterator$$anon$11	AbstractIterator	Iterator$$anon$11.next	

```
-- Top invokes
select m.targetMethod, count(1)
from (
       select i.invokeId, i.targetMethod from invokeinterfaces i group by i.invokeId, i.targetMethod) m
group by m.targetMethod
order by count desc;
```

abase Console: postgres@localhost [2]

Output | Result 44

177 rows

	targetmethod	count
1	Function1.apply	1534
2	Iterator.next	883
3	Iterator.hasNext	852
4	Function2.apply	525
5	GenMapLike.get	283
6	TraversableOnce.isEmpty	163
7	GenSetLike.contains	123
8	$plus	105
9	SeqGroup.run	80
10	CharSequence.charAt	68
11	StatsReceiver.isNull	67

Analyzing offsets

```sql
-- Invokes with all impls at the same offset
select m.targetMethod, m._offset, count(1) invokes
from (
      select *
      from (select i.invokeId, i.targetMethod, i._offset
            from invokeinterfaces i
            group by i.invokeId, i.targetMethod, i._offset) m
      where not exists(select * from invokeinterfaces i2 where i2.invokeId = m.invokeId and i2._offset != m._offset)) m
group by m.targetMethod, m._offset
order by invokes desc
```

abase Console: postgres@localhost [2]

Output | Result 41

107 rows

	targetmethod	_offset	invokes
1	Function1.apply	672	1486
2	Iterator.next	1208	583
3	Iterator.hasNext	1200	547
4	Function2.apply	912	525
5	GenMapLike.get	1920	283
6	GenSetLike.contains	1840	122
7	Function0.apply	536	55
8	$plus	1904	52
9	Map.updated	2000	50
10	GenTraversableOnce.seq	584	44
11	$plus	1856	43
12	GenericTraversableTemplate.companion	568	39
13	TraversableOnce.foldLeft	856	35

Top invokes

	targetmethod	count
1	Function1.apply	1534
2	Iterator.next	883
3	Iterator.hasNext	852
4	Function2.apply	525
5	GenMapLike.get	283
6	TraversableOnce.isEmpty	163
7	GenSetLike.contains	123
8	$plus	105
9	SeqGroup.run	80
10	CharSequence.charAt	68
11	StatsReceiver.isNull	67
12	GenTraversableOnce.seq	57
13	MapLike.get	56
14	Function0.apply	55
15	$plus$eq	53
16	StatsReceiver.addGauge	50
17	GenTraversableOnce.size	50
18	Map.updated	50
19	StatsReceiver.counter	41
20	TraversableOnce.foreach	41
21	Comparator.compare	41
22	MapGroup.run	40
23	SeqLike.length	39

Invokes with impls at the same offset

	targetmethod	_offset	invokes
1	Function1.apply	672	1486
2	Iterator.next	1208	583
3	Iterator.hasNext	1200	547
4	Function2.apply	912	525
5	GenMapLike.get	1920	283
6	GenSetLike.contains	1840	122
7	Function0.apply	536	55
8	$plus	1904	52
9	Map.updated	2000	50
10	GenTraversableOnce.seq	584	44
11	$plus	1856	43
12	GenericTraversableTemplat…	568	39
13	TraversableOnce.foldLeft	856	35
14	MapLike.get	1920	34
15	GenTraversableOnce.foreach	1304	29
16	divcolon	840	28
17	GenTraversableOnce.forall	680	21
18	GenTraversableOnce.size	1208	20
19	FloatingDecimal$ASCIIToBi…	464	20
20	ArrayDecoder.decode	560	14
21	Formattable.formatTo	464	14
22	GenericTraversableTemplat…	1056	13
23	TraversableOnce.seq	584	12

Deoptimizing is challenging

```sql
-- Deoptimize when the object doesn't extend the expected superclass
select m.superclass, count(1)
from (
        select *
        from (select i.invokeId, i.superclass, i._offset
              from invokeinterfaces i
              where i.superclass != ' Object'
              group by i.invokeId, i.superclass, i._offset) m
        where not exists(
            select *
            from invokeinterfaces i2
            where m.invokeId = i2.invokeId
              and (m._offset != i2._offset or m.superclass != i2.superclass))) m
group by m.superclass
order by count desc;
```

abase Console: postgres@localhost [2]

Output | Result 63

15 rows

	superclass	count
1	AbstractFunction1	1373
2	AbstractIterator	930
3	AbstractFunction2	525
4	AbstractFunction0	46
5	AbstractSet	38
6	AbstractMap	29
7	AbstractPartialFunction	23

Fingerprint

method.hpp

```
  int                 _compiled_invocation_count;  // Number of nmethod invocations so far (for perf. debugging)
#endif
  // Entry point for calling both from and to the interpreter.
  address _i2i_entry;           // All-args-on-stack calling convention
  // Adapter blob (i2c/c2i) for this Method*. Set once when method is linked.
  AdapterHandlerEntry* _adapter;

  Symbol* _fingerprint;
  // Entry point for calling from compiled code, to compiled code if it exists
  // or else the interpreter.
  volatile address _from_compiled_entry;        // Cache of: _code ? _code->entry_point() : _adapter->c2i_entry()
  // The entry point for calling both from and to compiled code is
  // "_code->entry_point()".  Because of tiered compilation and de-opt, this
  // field can come and go.  It can transition from NULL to not-null at any
  // time (whenever a compile completes).  It can transition from not-null to
  // NULL only at safepoints (because of a de-opt).
  nmethod* volatile _code;                      // Points to the corresponding piece of native code
  volatile address          _from_interpreted_entry; // Cache of _code ? _adapter->i2c_entry() : _i2i_entry
```

[Proposal] Invoke interface optimizations #893

Open **fwbrasil** opened this issue on Jan 3 · 6 comments

fwbrasil commented on Jan 3 • edited Contributor

I've been working on prototypes to optimize interface dispatches in Graal. I'm planning to start working on pull requests but would like to get your feedback on the overall approach before proceeding.

Problem

Scala codebases usually define computations using composable APIs that build a graph to be later executed by an interpreter. These APIs include abstractions like Scala's and Twitter's `Future`s for asynchronous computations, `Stitch` and `Clump` for batching of remote asynchronous calls, Monix's `Task` and ZIO's `IO` for defining pure computations, and many other monad-like APIs.

These abstractions introduce an execution behavior usually not present in traditional Java applications. Given that the definition of the computation graph involves several method invocations, the interpreter typically doesn't get inlined with the definition of the computation, which produces a high number of

Why Graal

Performance ✓

Ease of change ✓

Thank you!

Any questions?