Scala & Graal @ Twitter

Flavio W. Brasil
@flaviowbrasil
Twitter, VM Team

#TwitterVMTeam

Agenda

1 #Deploy

Why Twitter adopted Graal

2 #Profile

Scala performance challenges

3 #Optimize

New optimizations


#Deploy
Why Twitter adopted Graal

What is Graal?

Graal != GraalVM

Graal != GraalVM

Maxine: An Approachable Virtual Machine For, and In, Java

CHRISTIAN WIMMER, MICHAEL HAUPT, MICHAEL L. VAN DE VANTER,
MICK JORDAN, LAURENT DAYNÈS, and DOUGLAS SIMON, Oracle Labs

ACM Transactions on Architecture and Code Optimization, Vol. 9, No. 4, Article 30, Publication date: January 2013.

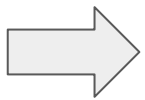
What is a JIT compiler?

Source code

```
@tailrec
protected final def continue(k: K[A]): Unit = state match {
  case waitq: WaitQueue[A] =>
    if (!cas(waitq, WaitQueue(k, waitq)))
      continue(k)
  case s: Interruptible[A] =>
    if (!cas(s, new Interruptible(WaitQueue(k, s.waitq), s.handler)))
      continue(k)
  case s: Transforming[A] =>
    if (!cas(s, new Transforming(WaitQueue(k, s.waitq), s.other)))
      continue(k)
  case s: Interrupted[A] =>
    if (!cas(s, new Interrupted(WaitQueue(k, s.waitq), s.signal)))
      continue(k)
  case v: Try[A] /* Done */ => k.runInScheduler(v)
  case p: Promise[A] /* Linked */ => p.continue(k)
}
```

Source code

```
@tailrec
protected final def continue(k: K[A]): Unit = state match {
  case waitq: WaitQueue[A] =>
    if (!cas(waitq, WaitQueue(k, waitq)))
      continue(k)
  case s: Interruptible[A] =>
    if (!cas(s, new Interruptible(WaitQueue(k, s.waitq), s.handler)))
      continue(k)
  case s: Transforming[A] =>
    if (!cas(s, new Transforming(WaitQueue(k, s.waitq), s.other)))
      continue(k)
  case s: Interrupted[A] =>
    if (!cas(s, new Interrupted(WaitQueue(k, s.waitq), s.signal)))
      continue(k)
  case v: Try[A] /* Done */ => k.runInScheduler(v)
  case p: Promise[A] /* Linked */ => p.continue(k)
}
```



Bytecode

```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem             # re
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
```

Source code

```
@tailrec
protected final def continue(k: K[A]): Unit = state match {
  case waitq: WaitQueue[A] =>
    if (!cas(waitq, WaitQueue(k, waitq)))
      continue(k)
  case s: Interruptible[A] =>
    if (!cas(s, new Interruptible(WaitQueue(k, s.waitq), s.handler)))
      continue(k)
  case s: Transforming[A] =>
    if (!cas(s, new Transforming(WaitQueue(k, s.waitq), s.other)))
      continue(k)
  case s: Interrupted[A] =>
    if (!cas(s, new Interrupted(WaitQueue(k, s.waitq), s.signal)))
      continue(k)
  case v: Try[A] /* Done */ => k.runInScheduler(v)
  case p: Promise[A] /* Linked */ => p.continue(k)
}
```

Bytecode

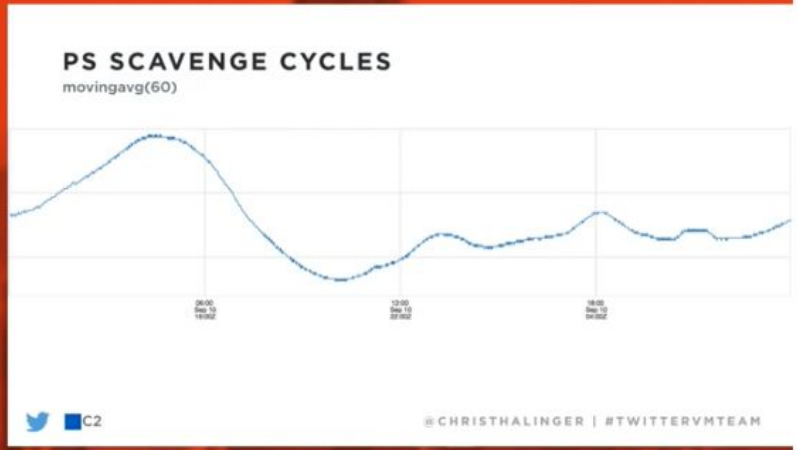
```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem             # re
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
```

Native

```
Block 116:
nop
mov rsi, rax
mov qword ptr [rsp+0x8], r13
jmp 0x7fb2dca5d000 <Block 92>
Block 117:
mov qword ptr [rsp+0x38], r11
mov rsi, 0x7f1191f20
shl r10, 0x3
mov qword ptr [rsp+0x40], r10
call 0x7fb2d97c27e0
Block 118:
nop
mov r8, r13
jmp 0x7fb2dca5d0a3 <Block 99>
```

Scala Days

New York 2018



Twitter's Quest for a Wholly Graal Runtime

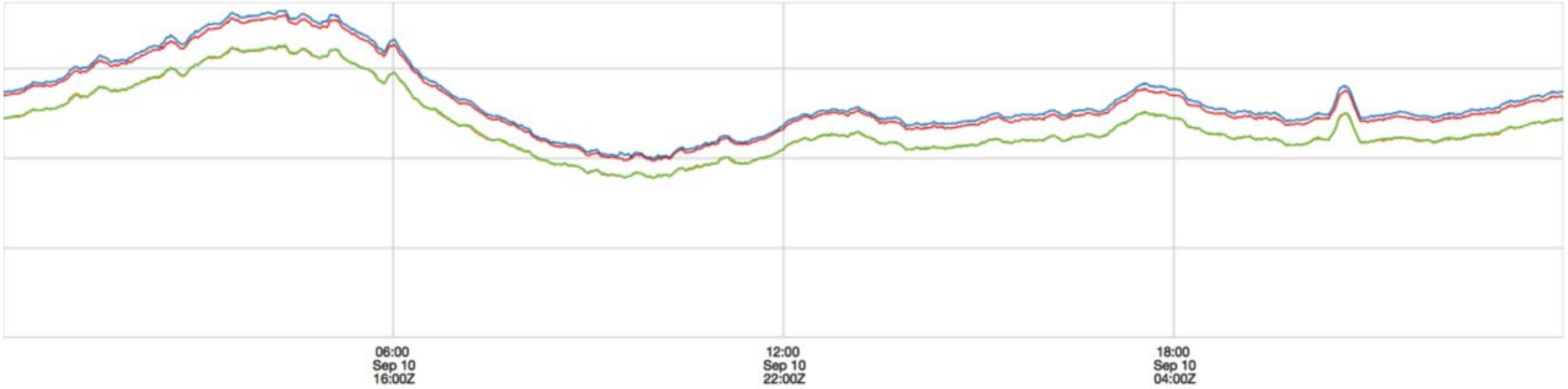
Chris Thalinger

25:59 / 50:10

Why Graal

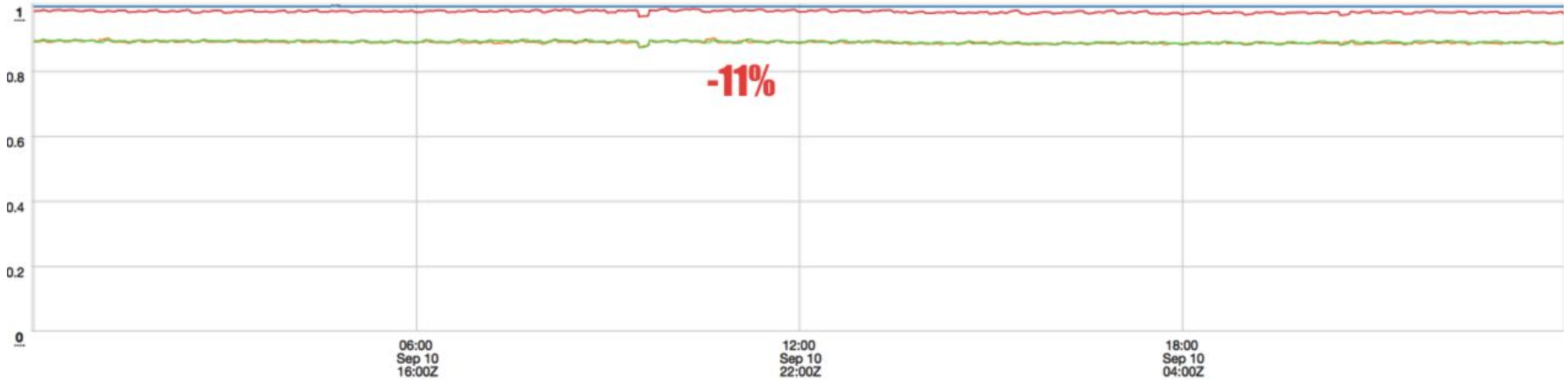
Performance
Ease of change

USER CPU TIME



C2 Graal C2 JDK9 Graal JDK9

USER CPU TIME - RATIO



C2 Graal C2 JDK9 Graal JDK9

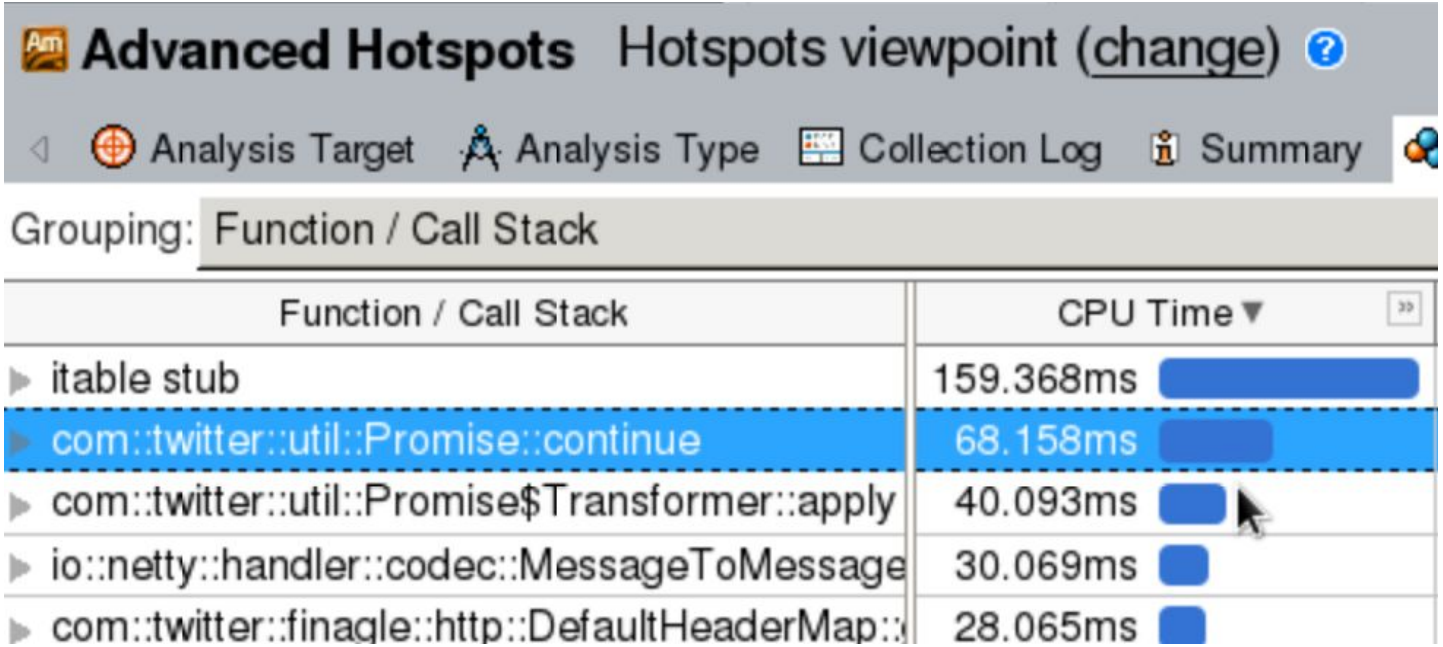
C2 is a 20yo codebase in C++

Graal is much easier to change


#Profile
Scala’s performance challenges

Scala is not Java

(but Java is becoming Scala)


Advanced Hotspots
Hotspots viewpoint (change)
Analysis Target
Analysis Type
Collection Log
Summary
Grouping: Function / Call Stack
Function / Call Stack
CPU Time
itable stub
159.368ms
com::twitter::util::Promise::continue
68.158ms
com::twitter::util::Promise$Transformer::apply
40.093ms
io::netty::handler::codec::MessageToMessage
30.069ms
com::twitter::finagle::http::DefaultHeaderMap::
28.065ms

Composable APIs

Scala's performance nightmare

Composable APIs are interpreted languages

```
sealed trait IO[I] {

}
```

```scala
sealed trait IO[I] {
  def map[U](f: I => U): IO[U]
}
```

```scala
sealed trait IO[I] {
  def map[U](f: I => 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]



}
```

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

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

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

```
def run[U](io: IO[U]): U =
  io match {


  }
```

```
def run[U](io: IO[U]): U =
  io match {
    case io: Value[U]  => io.v

  }
```

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

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

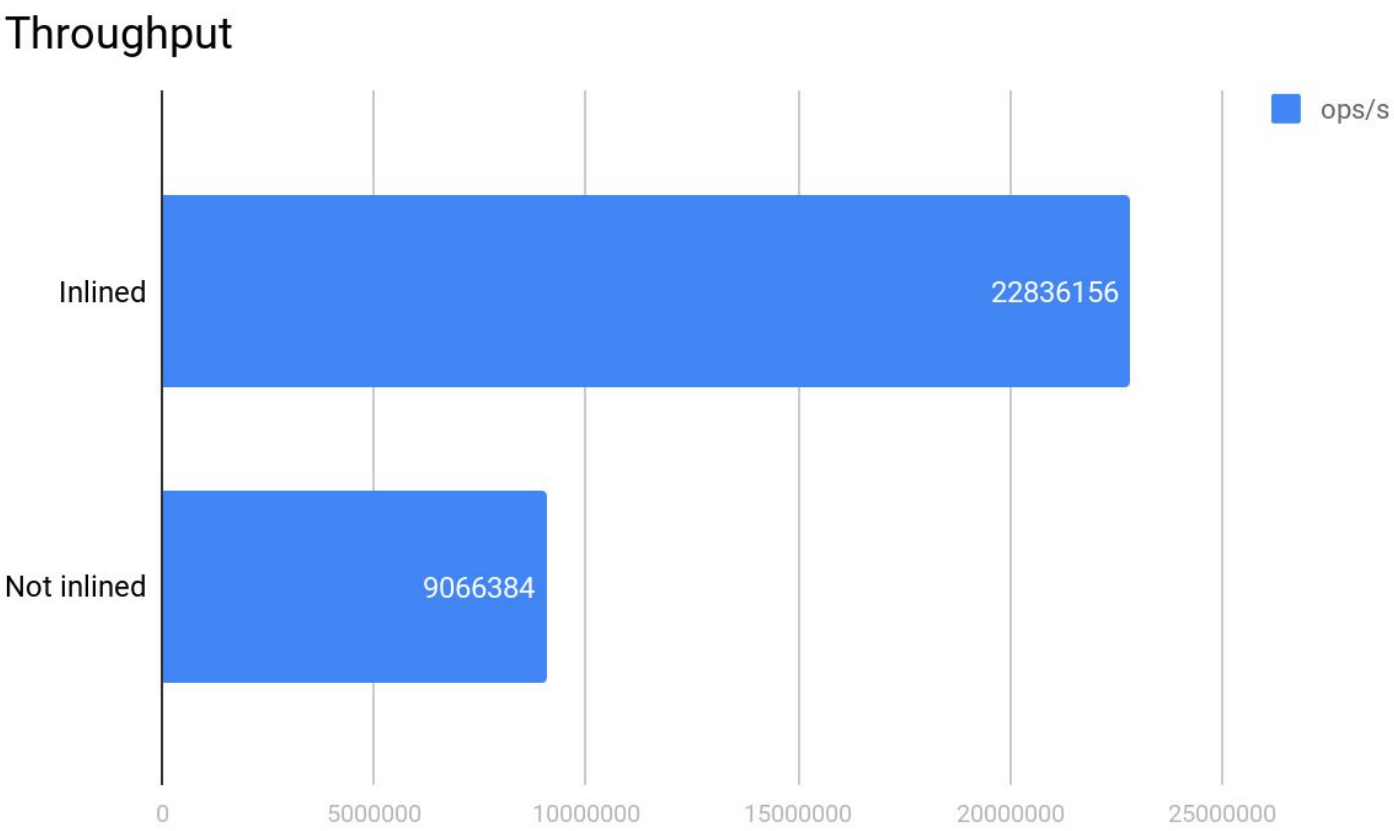
```scala
def plus10(n: Int): IO[Int] =
  IO.value(n).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1)
```

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

```
@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))
}


def runNotInlined[T](io: IO[T]): T = {
  run(io)
}
```

```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)
}
```


Throughput
ops/s
Inlined
22836156
Not inlined
9066384
0
5000000
10000000
15000000
20000000
25000000

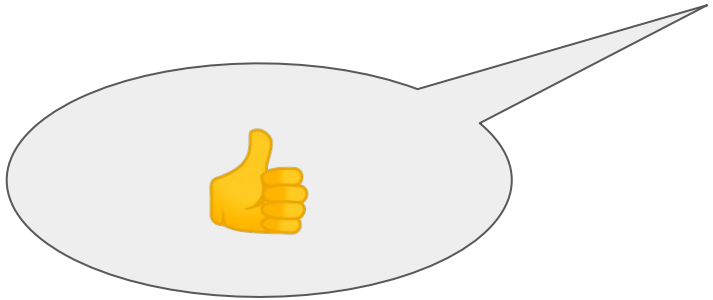
Why does a single indirection have this effect?

```
run(plus10(20))
```

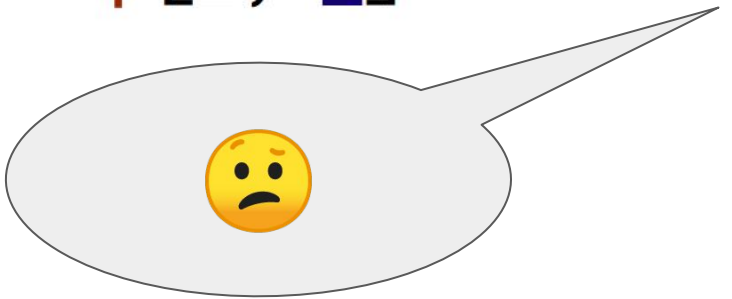
```
run(
  IO.value(20).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1).map(_ + 1)
    .map(_ + 1).map(_ + 1).map(_ + 1)
)
```

```
IO.value(20).map(_ + 1).map(_ + 1).map(_ + 1)
  .map(_ + 1).map(_ + 1).map(_ + 1).map(_ + 1)
  .map(_ + 1).map(_ + 1).map(_ + 1) match {
    case io: Value[Int]  => io.v
    case io: Map[_, Int] => io.f(run(io.io))
  }
```

```
IO.value(20).map(_ + 1).map(_ + 1).map(_ + 1)
  .map(_ + 1).map(_ + 1).map(_ + 1).map(_ + 1)
  .map(_ + 1).map(_ + 1).map(_ + 1) match {
    case io: Value[Int]  => io.v
    case io: Map[_, Int] => io.f(run(io.io))
  }
```



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



Interpreted languages are difficult to optimize

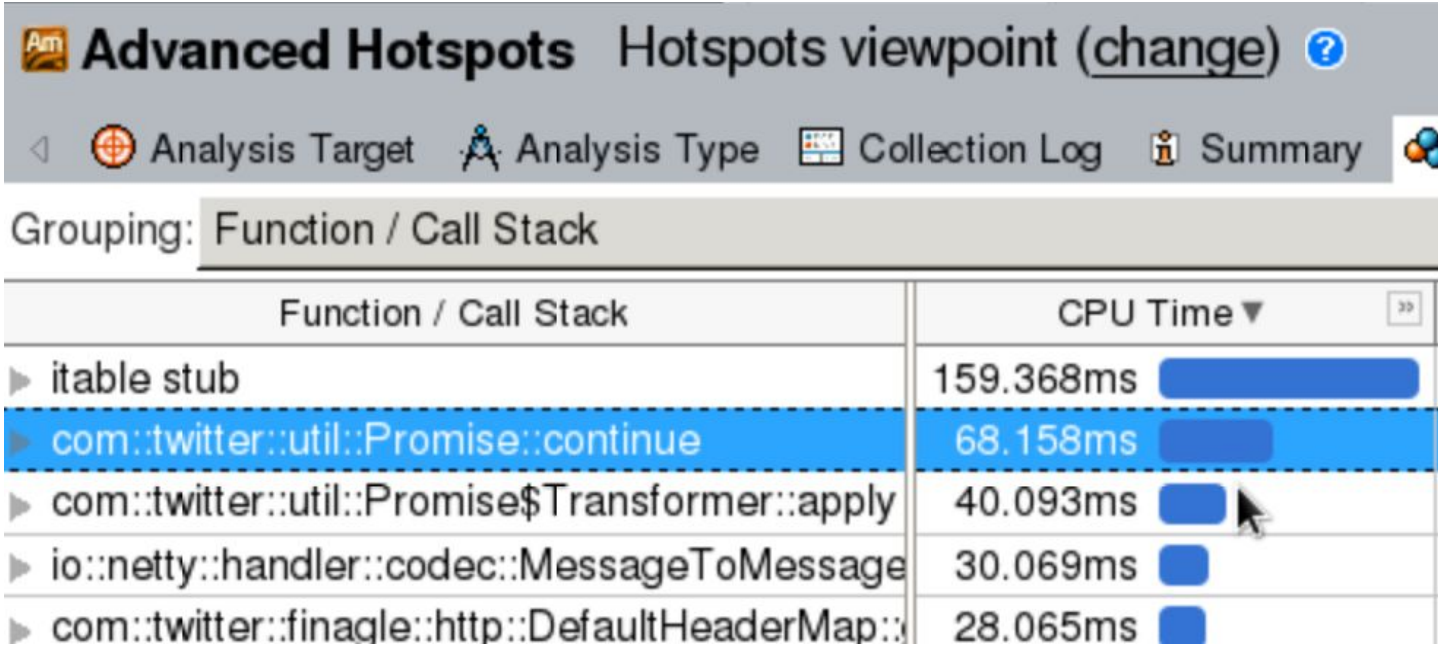
Embedded

Interpreted languages are

even more

difficult to optimize


#Optimize
New optimizations by the #TwitterVMTeam


Advanced Hotspots
Hotspots viewpoint (change)
Analysis Target
Analysis Type
Collection Log
Summary
Grouping: Function / Call Stack
Function / Call Stack
CPU Time
itable stub
159.368ms
com::twitter::util::Promise::continue
68.158ms
com::twitter::util::Promise$Transformer::apply
40.093ms
io::netty::handler::codec::MessageToMessage
30.069ms
com::twitter::finagle::http::DefaultHeaderMap::
28.065ms

What is an itable stub?

Virtual call

```
class Mammal {
  def speak = "oh"
}
```

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Object	
getClass	1
toString	2

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Object	
getClass	1
toString	2

Mammal	
getClass	1
toString	2

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Object	
getClass	1
toString	2

Mammal	
getClass	1
toString	2
speak	3

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Object	
getClass	1
toString	2

Mammal	
getClass	1
toString	2
speak	3

Human	
getClass	1
toString	2
speak	3

Virtual call

```
class Mammal {
  def speak = "oh"
}

class Human extends Mammal {

  def speak(lang: String) =
    if (lang == "Hindi")
      "Namaste"
    else
      "Hello"

  override def speak = "hello"

  override def toString = "A Human"
}
```

```
def test(m: Mammal) =
  m.speak
```

Object	
getClass	1
toString	2

Mammal	
getClass	1
toString	2
speak	3

Human	
getClass	1
toString	2
speak	3
speak(String)	4

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}
```

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

Interface call

```scala
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```scala
def test(m: Mammal) =
  m.speak
```

Human	
Named	1
Mammal	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

Human	
Named	1
Mammal	2

Human/ Named	
name	1

Human/ Mammal	
speak	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

?

Human	
Named	1
Mammal	2

Human/ Named	
name	1

Human/ Mammal	
speak	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

❌

Human	
Named	1
Mammal	2

Human/ Named	
name	1

Human/ Mammal	
speak	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

❌ ❓

Human	
Named	1
Mammal	2

Human/ Named	
name	1

Human/ Mammal	
speak	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

❌ ✔️

Human	
Named	1
Mammal	2

Human/ Named	
name	1

Human/ Mammal	
speak	2

Interface call

```
trait Named {
  def name: String
}

trait Mammal {
  def speak: String
}

class Human
  extends Named
  with Mammal {

  override def speak = "hello"
  override def name = "Someone"
}
```

```
def test(m: Mammal) =
  m.speak
```

❌ ✔️

Human	
Named	1
Mammal	2

Human/ Named	
name	1

✔️

Human/ Mammal	
speak	2

How can we optimize it?

```
def test(m: Mammal) =
  m.speak
```

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	

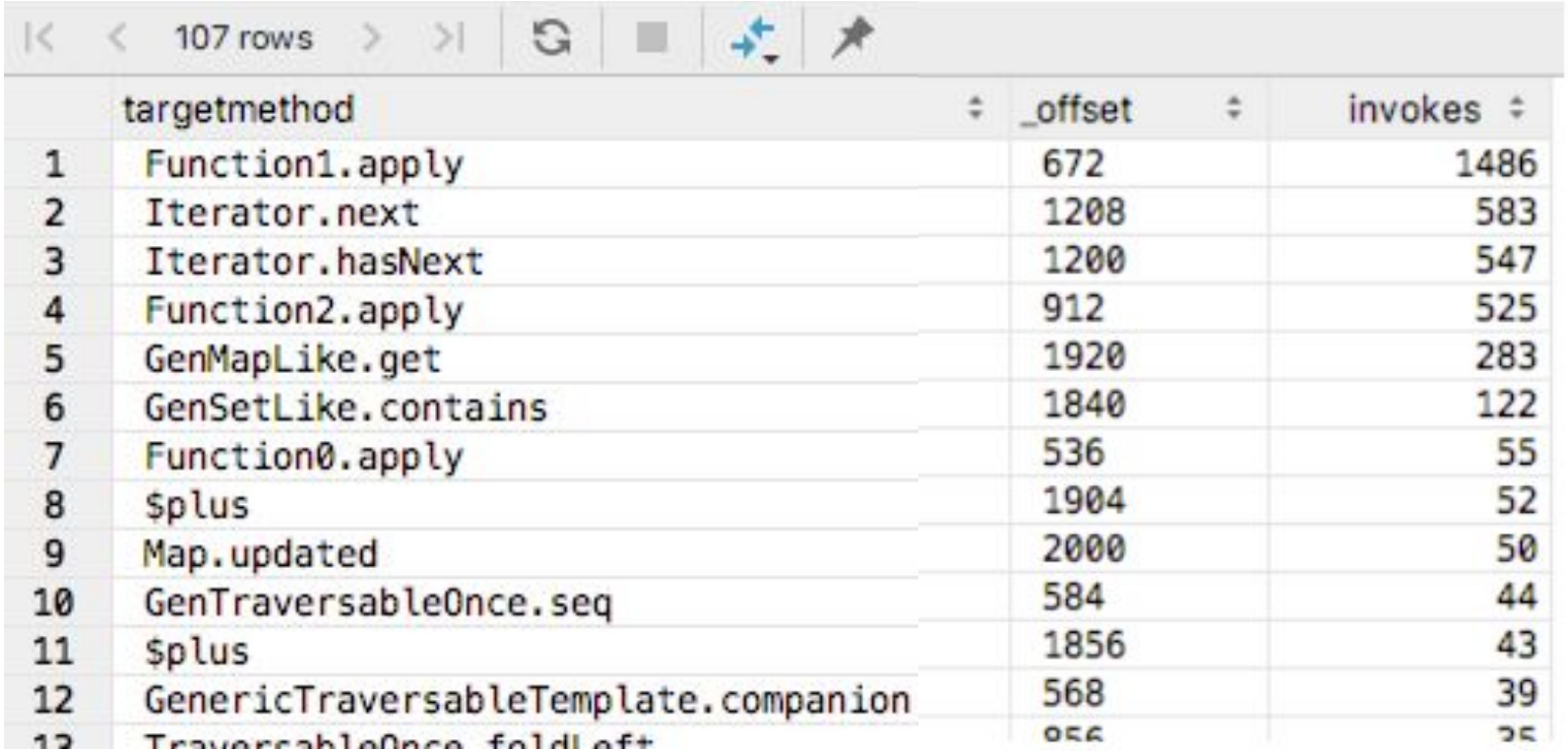
Top invokes

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

Common offsets

Common offsets



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

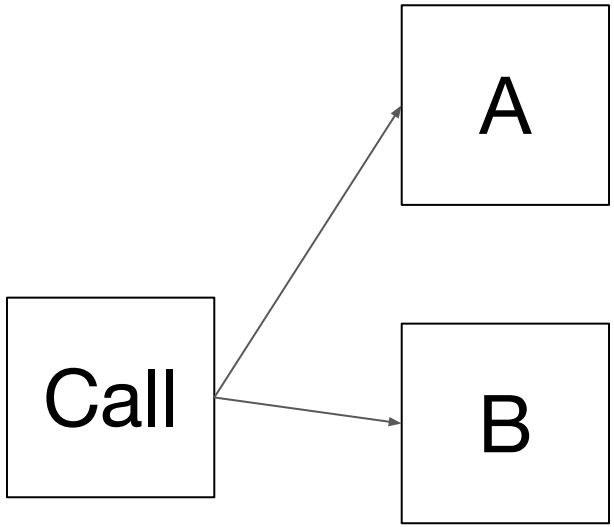
Top invokes

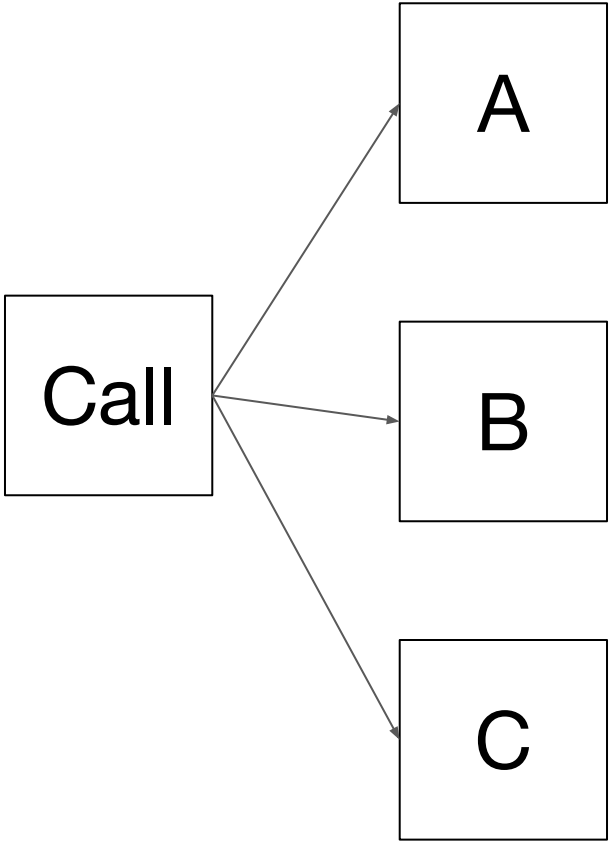
	targetmethod	count
1	Function1.apply	1534
2	Iterator.next	883
3	Iterator.hasNext	852
4	Function2.apply	525
5	GenMapLike.get	283
6	TraversableOnce.isEmpty	163
7	GenSetLike.contains	123
8	$plus	105
9	SeqGroup.run	80
10	CharSequence.charAt	68
11	StatsReceiver.isNull	67
12	GenTraversableOnce.seq	57
13	MapLike.get	56
14	Function0.apply	55

Common offsets

	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

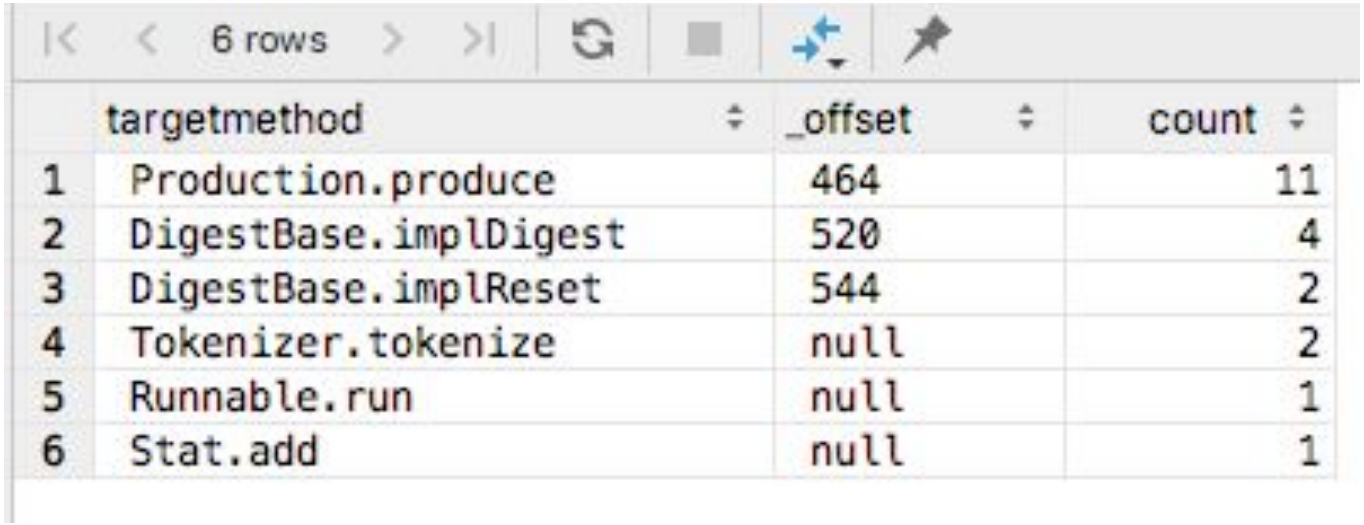
What if a new class is observed?


A
Call
B


A
Call
B
C

Deoptimization is challenging

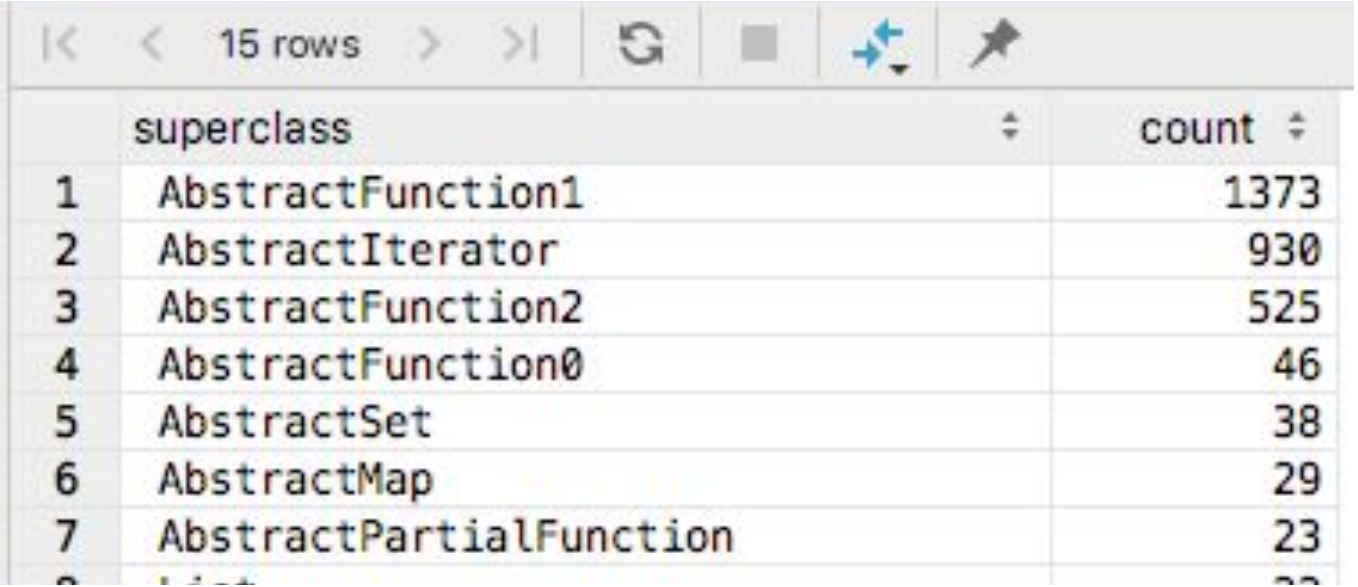
Single method



6 rows

	targetmethod	_offset	count
1	Production.produce	464	11
2	DigestBase.implDigest	520	4
3	DigestBase.implReset	544	2
4	Tokenizer.tokenize	null	2
5	Runnable.run	null	1
6	Stat.add	null	1

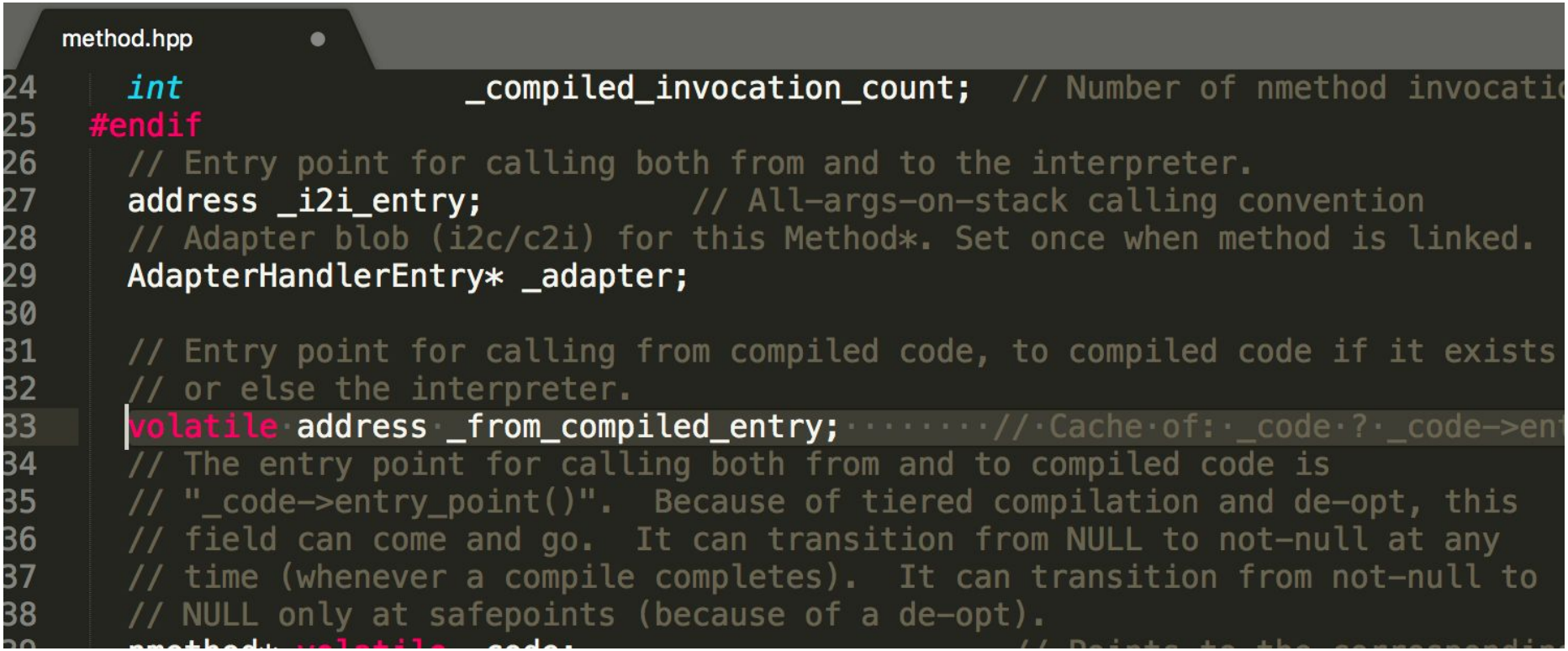
Common super class

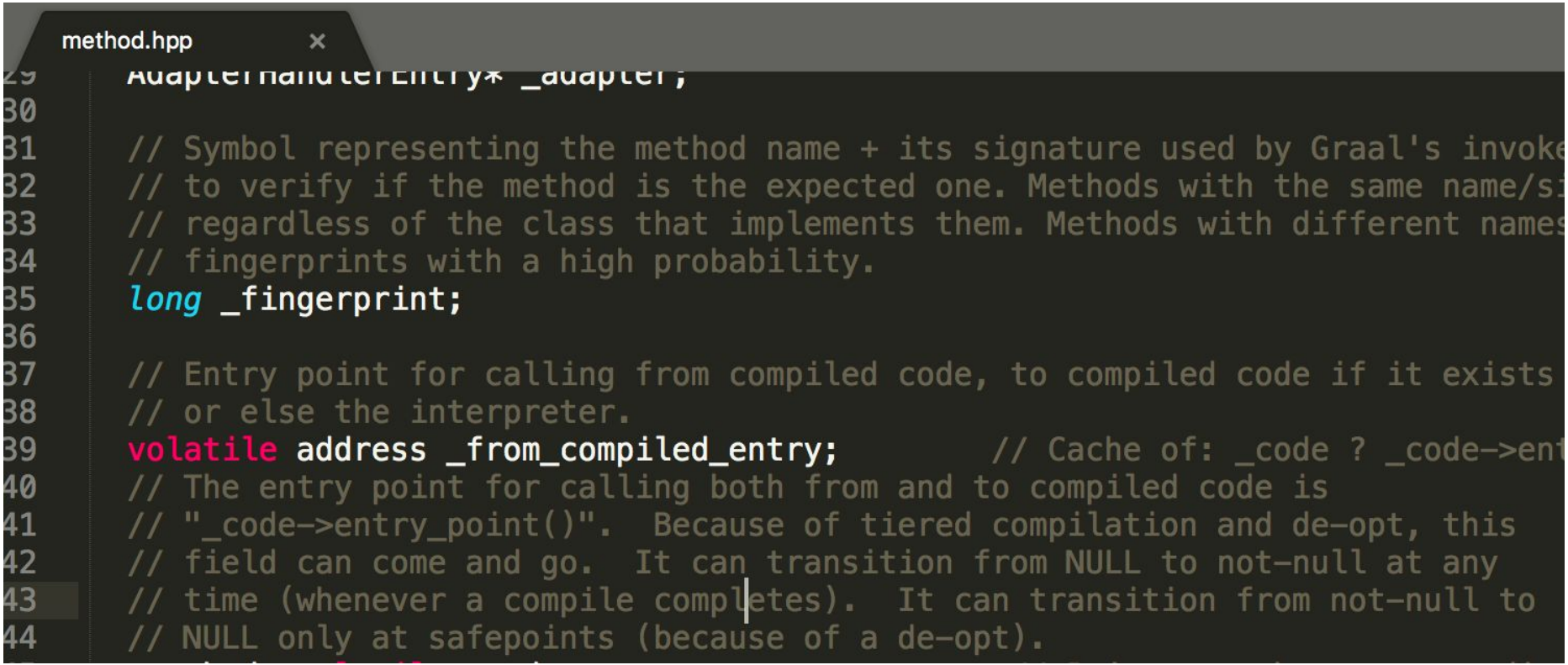


15 rows

	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
24 int _compiled_invocation_count; // Number of nmethod invocati
25 #endif
26 // Entry point for calling both from and to the interpreter.
27 address _i2i_entry; // All-args-on-stack calling convention
28 // Adapter blob (i2c/c2i) for this Method*. Set once when method is linked.
29 AdapterHandlerEntry* _adapter;
30
31 // Entry point for calling from compiled code, to compiled code if it exists
32 // or else the interpreter.
33 volatile address _from_compiled_entry; //Cache of: _code ? _code->ent
34 // The entry point for calling both from and to compiled code is
35 // "_code->entry_point()". Because of tiered compilation and de-opt, this
36 // field can come and go. It can transition from NULL to not-null at any
37 // time (whenever a compile completes). It can transition from not-null to
38 // NULL only at safepoints (because of a de-opt).


method.hpp
AdapterHandlerEntry* _adapter;
// Symbol representing the method name + its signature used by Graal's invoke
// to verify if the method is the expected one. Methods with the same name/si
// regardless of the class that implements them. Methods with different names
// fingerprints with a high probability.
long _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->ent
// 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).

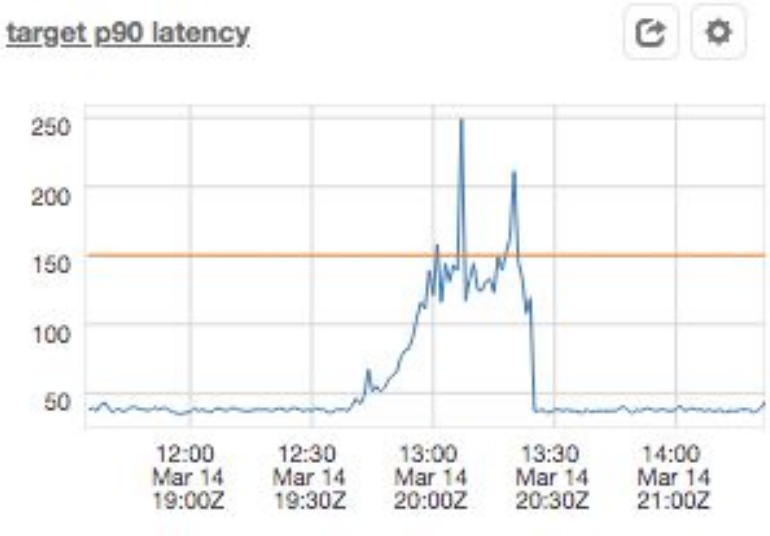
Results

Invoke interface optimization
(stress test)

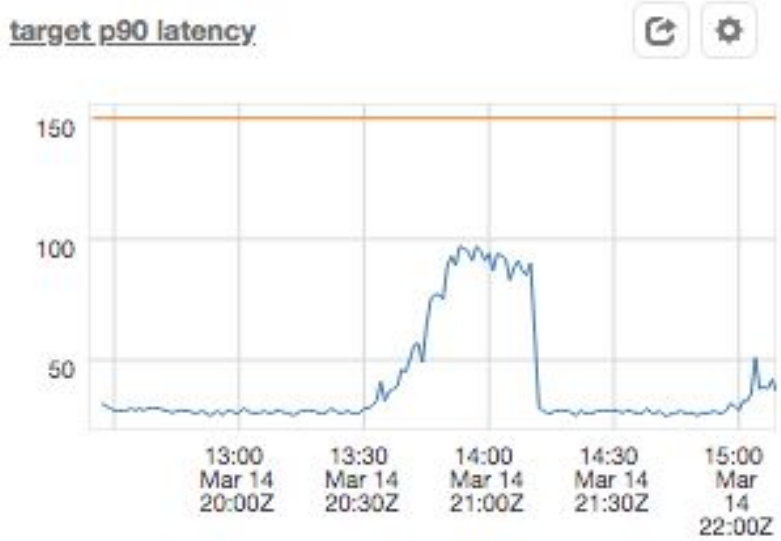
	Max RPS	Max QPS
Baseline	735	3762
Optimized	1349	4548
Improvement	**83%**	**20%**

Invoke interface optimization
(stress test)

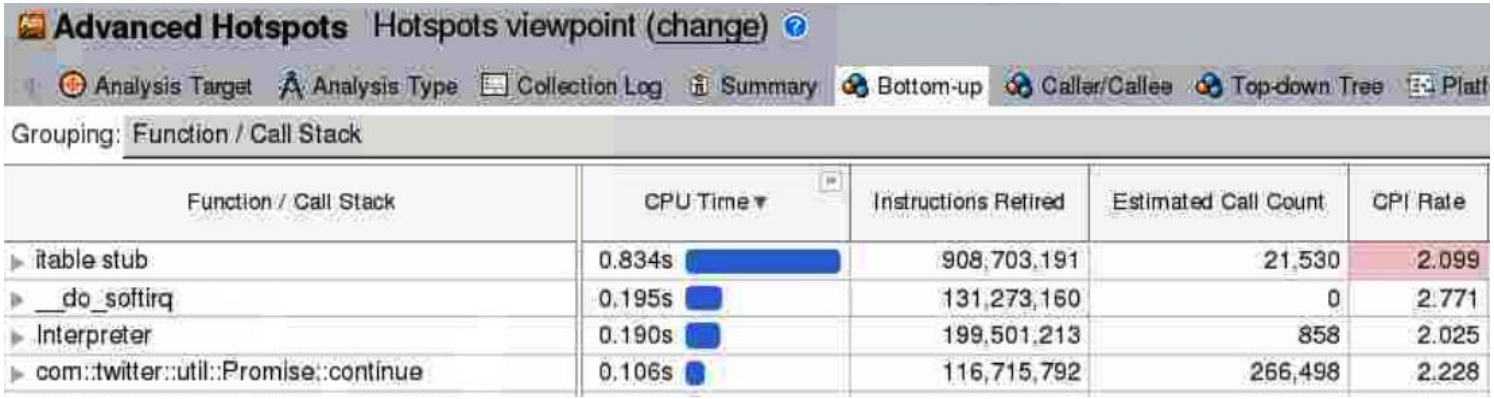
Before



After



Before



Function / Call Stack	CPU Time ▾	Instructions Retired	Estimated Call Count	CPI Rate
itable stub	0.834s	908,703,191	21,530	2.099
__do_softirq	0.195s	131,273,160	0	2.771
Interpreter	0.190s	199,501,213	858	2.025
com::twitter::util::Promise::continue	0.106s	116,715,792	266,498	2.228

After

Function / Call Stack	CPU Time ▾	Instructions Retired	Estimated Call Count	CPI Rate
itable stub	0.340s	316,326,316	87,393	2.303
__do_softirq	0.162s	103,213,645	0	2.964
Interpreter	0.153s	153,226,386	616	2.140
com::twitter::util::Promise$Transformer::apply	0.126s	108,692,416	634,240	2.469

Why Graal

Performance ✔
Ease of change ✔

Thank you!