KotlinConf

The Cost of Kotlin Language Features

Duncan McGregor
@duncanmcg

Why do I care?

```
fun arbitraryFunction(s: String): String {
   val upperCased = s.toUpperCase()
   return if (!upperCased.endsWith(" BANANA")) upperCased
   else upperCased + " BANANA"
}
```

```
fun arbitraryFunction(s: String) = s.toUpperCase().let {
   if (!it.endsWith(" BANANA")) it
   else it + " BANANA"
}
```

Aims

1. Are there language features that we should habitually avoid?
   - tl;dr - not that I found
     - For the places I looked
     - Java 8 on the JVM
     - Kotlin 1.1.50
     - Raspberry Pi running Raspbian
     - There are lies, damn lies, and statistics about micro-benchmarks
2. How to investigate costs for yourself

Language Features Examined

- Let
- Null Safety
- String interpolation
- Properties
- First-class functions
- Iteration (mapping)
- Default collections

github.com/dmcg/kostings

Let

Let

```
fun baseline(state: LetState): LetState {
    val v = state
    return v
}


fun let(state: LetState) = state.let {
    it
}
```

Let

```
fun baseline(state: LetState): LetState {
    val v = state
    return v
}
```

```
public final baseline(LcostOfKotlin/let/LetState;)LcostOfKotlin/let/LetState;
 L0
  LINENUMBER 12 L0
  ALOAD 1
  ASTORE 2
 L1
  LINENUMBER 13 L1
  ALOAD 2
  ARETURN
```

Let

```
fun let(state: LetState) = state.let {
    it
}
```

```
public final let(LcostOfKotlin/let/LetState;)LcostOfKotlin/let/LetState;
  L0
   LINENUMBER 17 L0
   ALOAD 1
   ASTORE 2
  L1
   ALOAD 2
   ASTORE 3
  L2
   LINENUMBER 18 L2
   ALOAD 3
  L3
  L4
   LINENUMBER 17 L4
   NOP
  L5
```

Let

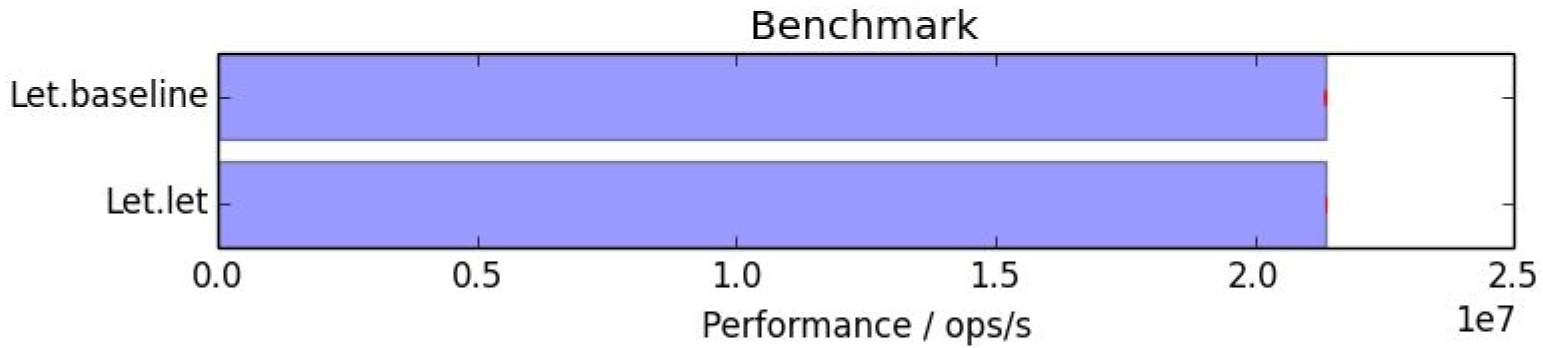
```
public final baseline(LcostOfKotlin/let/LetState;)LcostOfKotlin/let/LetState;
  L0
   LINENUMBER 12 L0
   ALOAD 1
   ASTORE 2
  L1
   LINENUMBER 13 L1
   ALOAD 2
   ARETURN
```

```
public final let(LcostOfKotlin/let/LetState;)LcostOfKotlin/let/LetState;
  L0
   LINENUMBER 17 L0
   ALOAD 1
   ASTORE 2
  L1
   ALOAD 2
   ASTORE 3
  L2
   LINENUMBER 18 L2
   ALOAD 3
  L3
  L4
   LINENUMBER 17 L4
   NOP
  L5
   LINENUMBER 19 L5
   ARETURN
```

```
public inline fun <T, R> T.let(block: (T) -> R): R = block(this)
```

Let

```
fun baseline(state: LetState): LetState {
    val v = state
    return v
}

fun let(state: LetState) = state.let {
    it
}
```

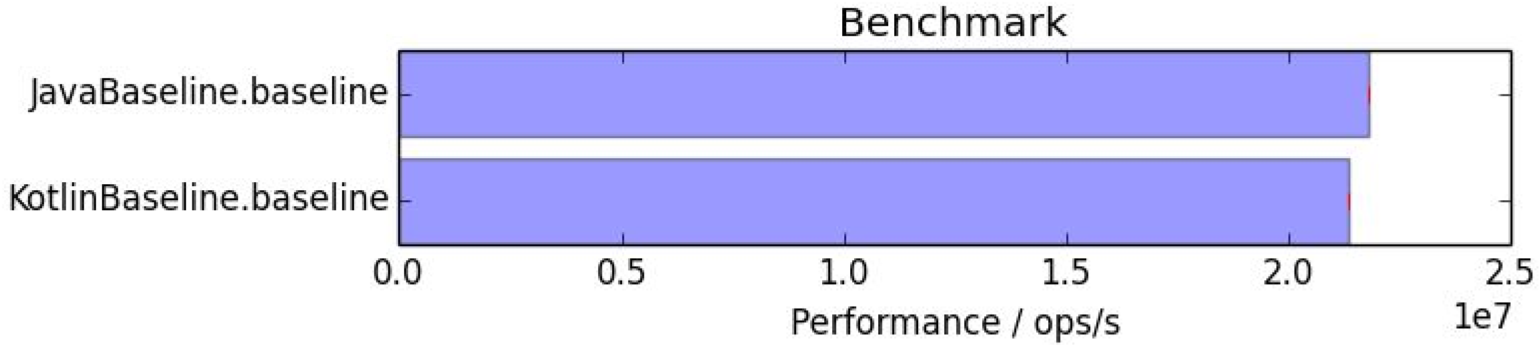


Null Safety

Null Safety

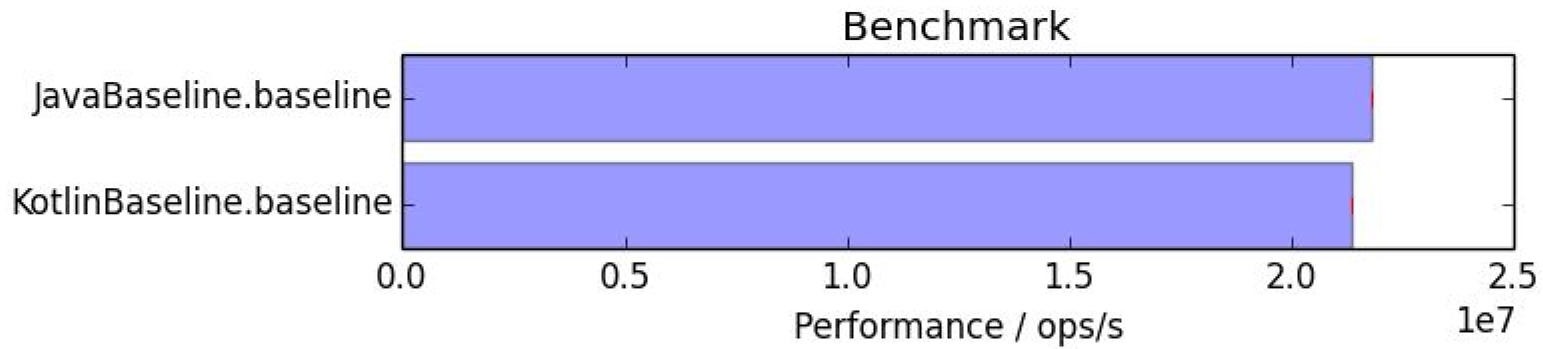
```java
public class JavaBaseline {
    @Benchmark
    public EmptyState baseline(EmptyState state) {
        return state;
    }
}
```

```kotlin
open class KotlinBaseline {
    @Benchmark
    fun baseline(state: EmptyState): EmptyState {
        return state
    }
}
```



Null Safety

```
@Test
fun `Kotlin null check has some cost`() {
   assertThat(
       JavaBaseline::baseline,
       probablyFasterThan(
           KotlinBaseline::baseline,
           byMoreThan = 0.03,
           butNotMoreThan = 0.04))
}
```



Null Safety

```
@Benchmark
public EmptyState baseline(EmptyState state) { return state; }
```

```
public baseline(LcostOfKotlin/baselines/EmptyState;)LcostOfKotlin/baselines/EmptyState;
@Lorg/openjdk/jmh/annotations/Benchmark;()
 L0
  ALOAD 1
  ARETURN
```

```
@Benchmark
fun baseline(state: EmptyState) = state
```

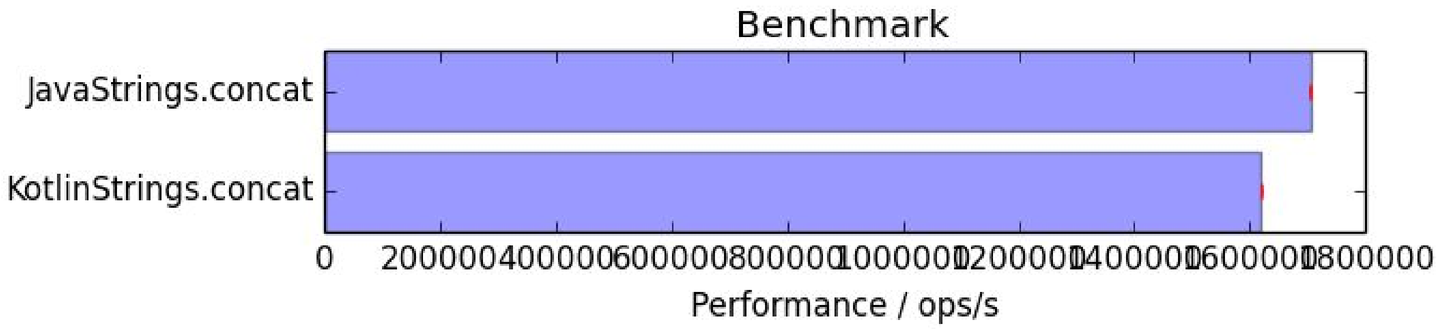
```
public final baseline(LcostOfKotlin/baselines/EmptyState;)LcostOfKotlin/baselines/EmptyState;
@Lorg/openjdk/jmh/annotations/Benchmark;()
@Lorg/jetbrains/annotations/NotNull;() // invisible
  @Lorg/jetbrains/annotations/NotNull;() // invisible, parameter 0
 L0
  ALOAD 1
  LDC "state"
  INVOKESTATIC kotlin/jvm/internal/Intrinsics.checkParameterIsNotNull
(Ljava/lang/Object;Ljava/lang/String;)V
 L1
  ALOAD 1
  ARETURN
```

String Interpolation

String Interpolation

```
@State(Scope.Benchmark)
public class StringState {
    public String greeting = "hello";
    public String subject = "world";
}

@Benchmark
public String concat(StringState state) {
    return state.greeting + " " + state.subject;
}

@Benchmark
fun concat(state: StringState): String {
    return "${state.greeting} ${state.subject}"
}
```

```
@Test
fun `Java is quicker but not by much`() {
    assertThat(JavaStrings::concat,
        probablyFasterThan(KotlinStrings::concat,
            byMoreThan = 0.05,
            butNotMoreThan = 0.08))
}
```

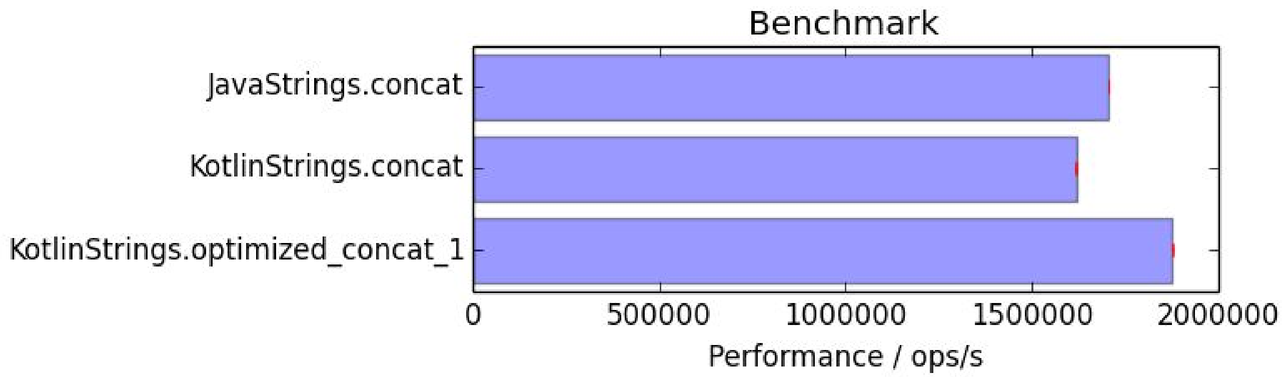


String Interpolation

```
@Benchmark
fun concat(state: StringState): String {
   return "${state.greeting} ${state.subject}"
}
```

```
fun desugared_concat(state: StringState): String? {
   return StringBuilder()
        .append("")
        .append(state.greeting)
        .append(' ')
        .append(state.subject)
        .toString()
}
```

```
fun optimized_concat_1(state: StringState): String? {
   return StringBuilder()
        .append(state.greeting)
        .append(' ')
        .append(state.subject)
        .toString()
}
```

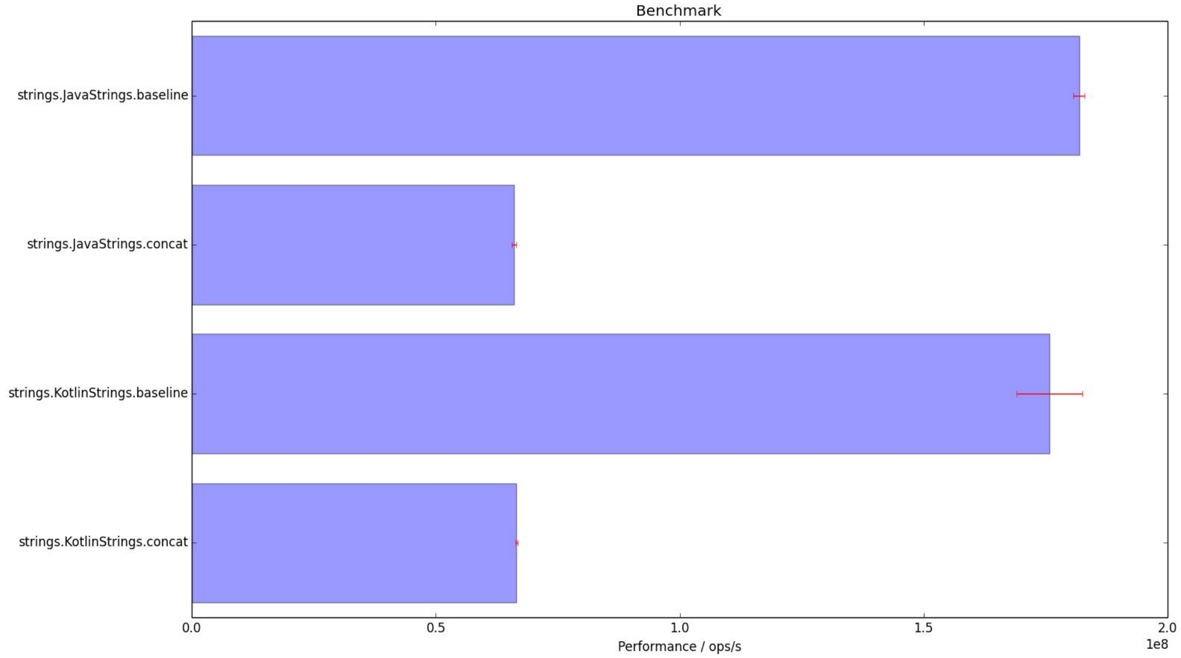


String Interpolation

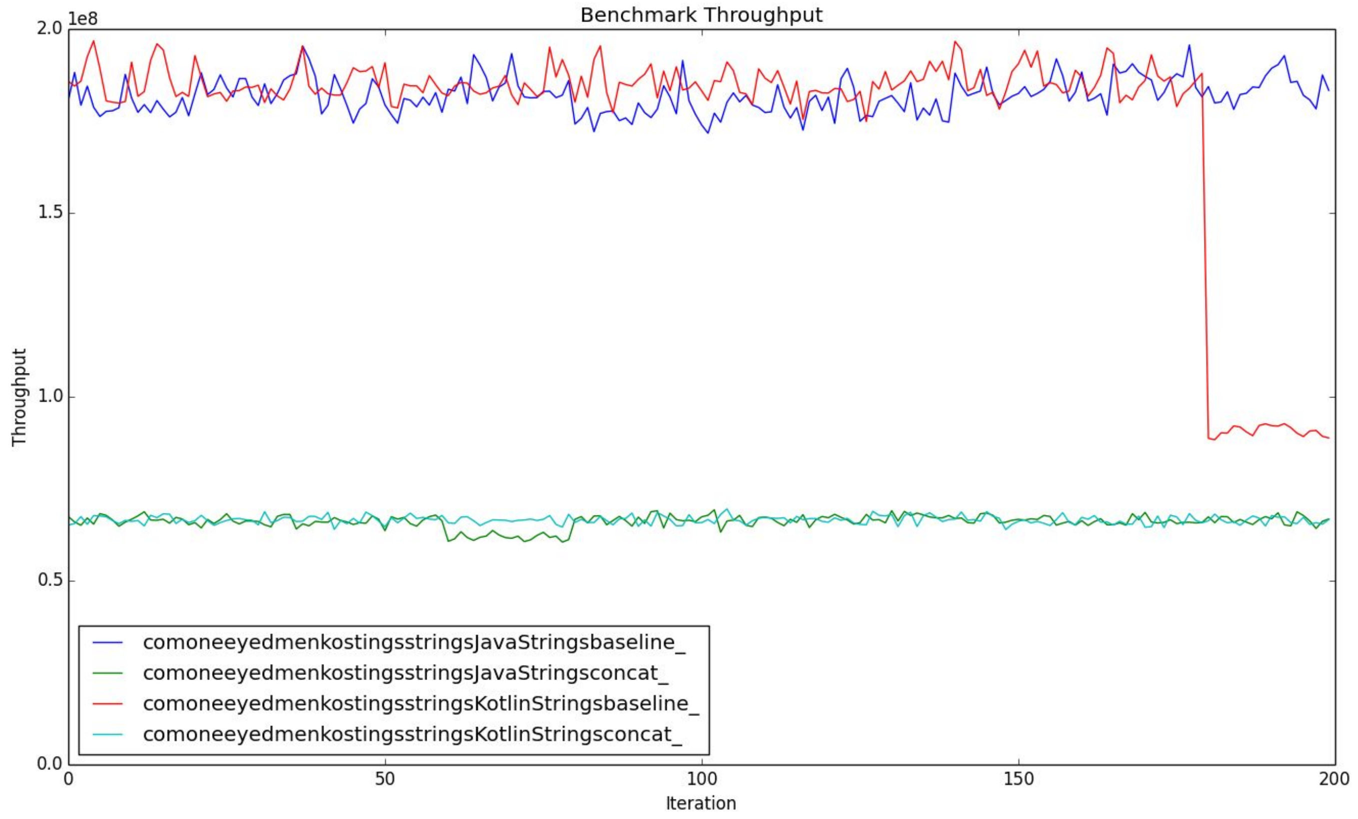
```kotlin
fun `the compiler optimizes this to a constant`() = "${"hello"} ${"world"}"

fun `and even this`() = "${"${"hello" + " " + "world"}"}"



private val hello = "hello"
private val world = "world"

fun `but not this`() = "$hello $world"
```

Running Benchmarks

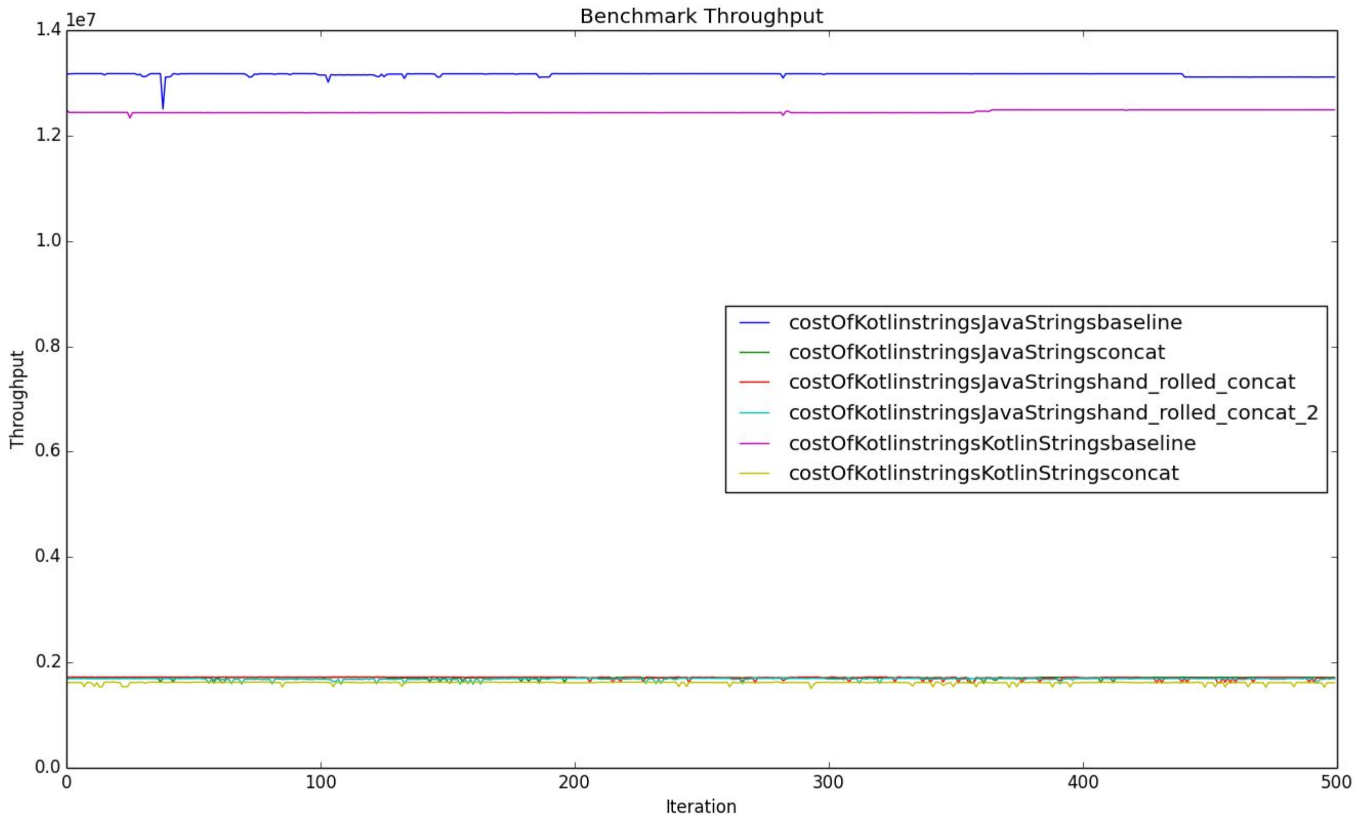
JMH Benchmark Run on MacOS



JMH Benchmark Run on MacOS



JMH Benchmark Run on Raspberry Pi



Properties

Properties

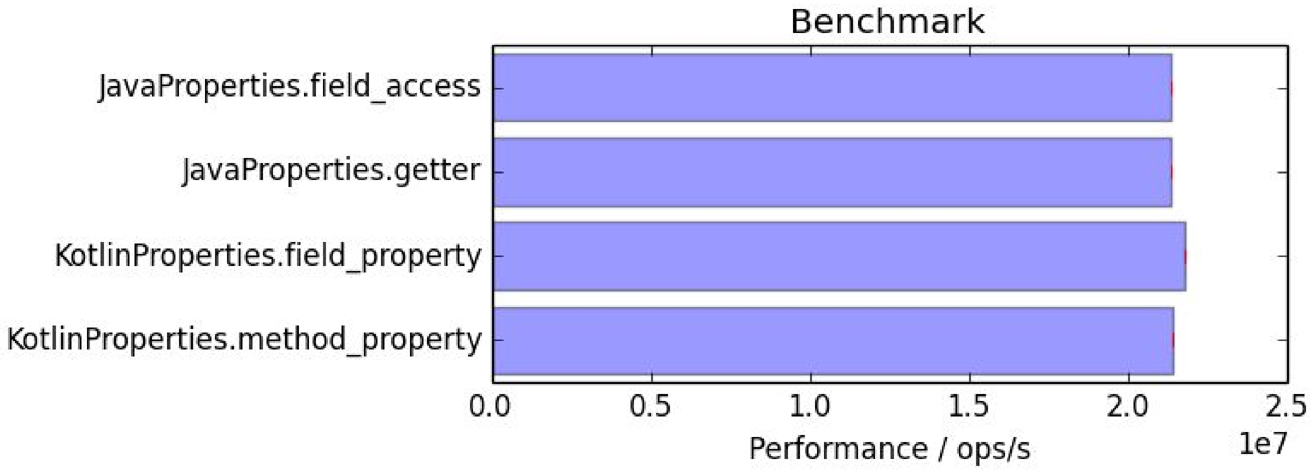
```java
@State(Scope.Benchmark)
public class JavaState {

   public String field = "hello";

   public String getField() {
       return field;
   }
}

@Benchmark
public String field_access(JavaState state) {
   return state.field;
}

@Benchmark
public String getter(JavaState state) {
   return state.getField();
}
```

```kotlin
@State(Scope.Benchmark)
open class KotlinState {

   val fieldProperty = "hello"

   val methodProperty get() = "hello"
}

@Benchmark
fun field_property(state: KotlinState): String {
    return state.fieldProperty
}

@Benchmark
fun method_property(state: KotlinState): String
{
    return state.methodProperty
}
```

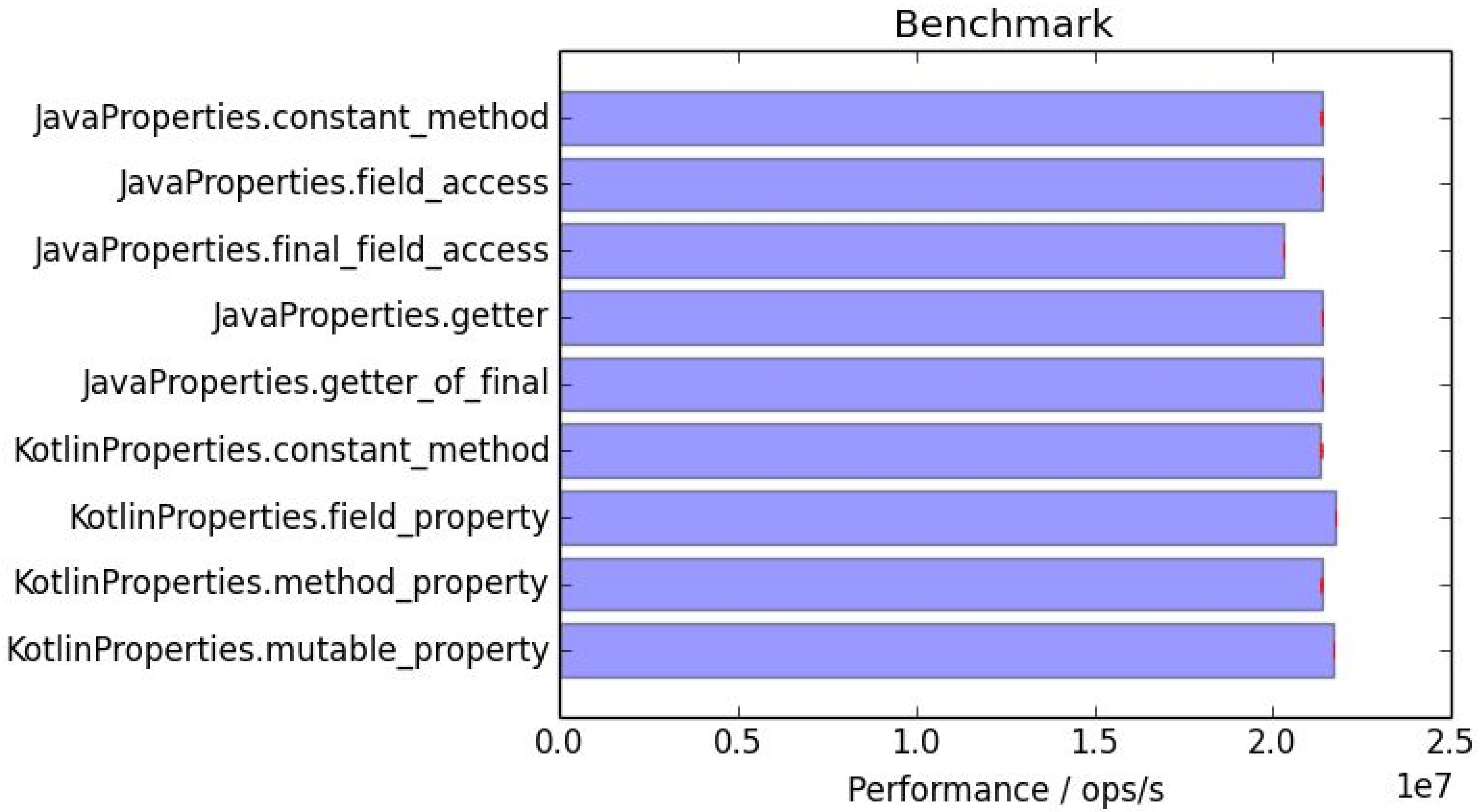
Properties

```java
@State(Scope.Benchmark)
public class JavaState {

   public String field = "hello";

   public String getField() {
       return field;
   }
}
```

```kotlin
@State(Scope.Benchmark)
open class KotlinState {

   val fieldProperty = "hello"

   val methodProperty get() = "hello"
}
```



Properties



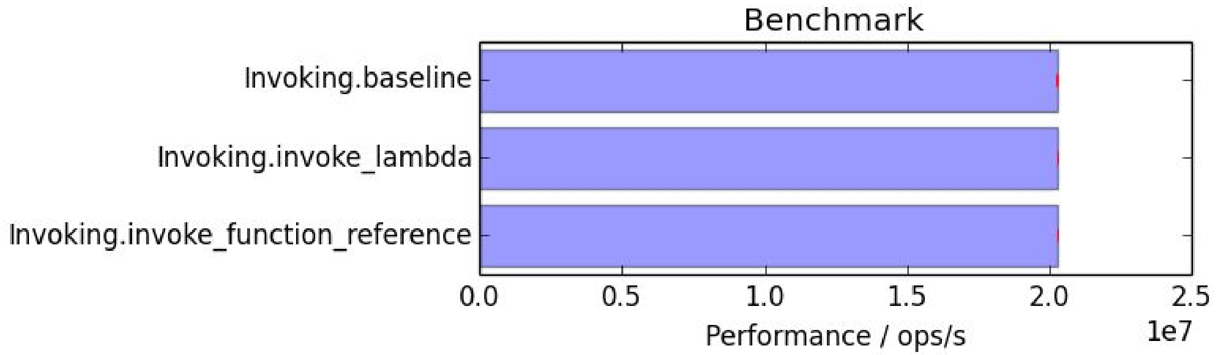
Benchmark
JavaProperties.constant_method
JavaProperties.field_access
JavaProperties.final_field_access
JavaProperties.getter
JavaProperties.getter_of_final
KotlinProperties.constant_method
KotlinProperties.field_property
KotlinProperties.method_property
KotlinProperties.mutable_property
0.0
0.5
1.0
1.5
2.0
2.5
Performance / ops/s
1e7

First-class Functions

First-class Functions

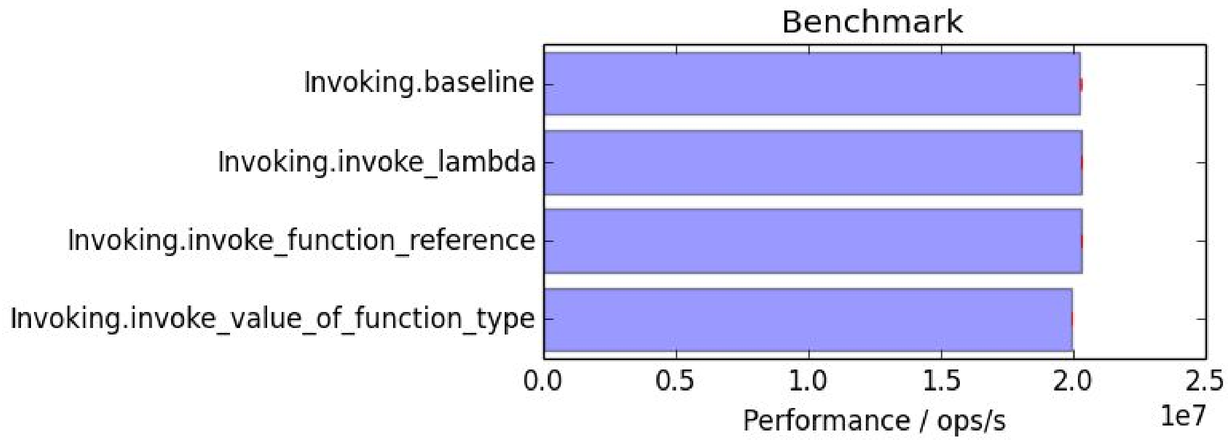
```
@Benchmark
fun baseline(state: MiscState) : String {
    return identity(state.aString)
}

@Benchmark
fun invoke_lambda(state: MiscState) : String {
    return invokeWith(state.aString) { identity(it) }
}

@Benchmark
fun invoke_function_reference(state: MiscState) : String {
    return invokeWith(state.aString, ::identity)
}
```

```
fun identity(s: String) = s

inline fun <T> invokeWith(t: T, f: (T) -> T) = f(t)
```



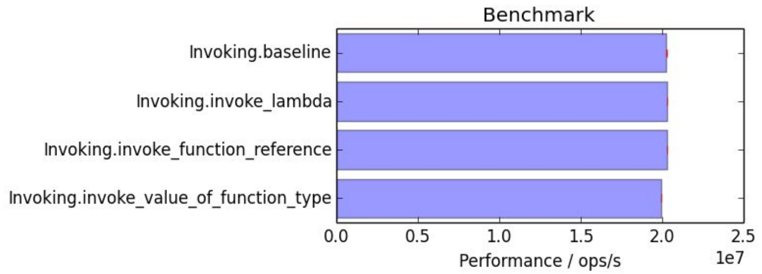
First-class Functions

```
@Benchmark
fun invoke_function_reference(state: MiscState) : String {
   return invokeWith(state.aString, ::identity)
}

val identityAsValue: (String) -> String = ::identity

@Benchmark
fun invoke_value_of_function_type(state: MiscState) : String {
   return invokeWith(state.aString, identityAsValue)
}
```



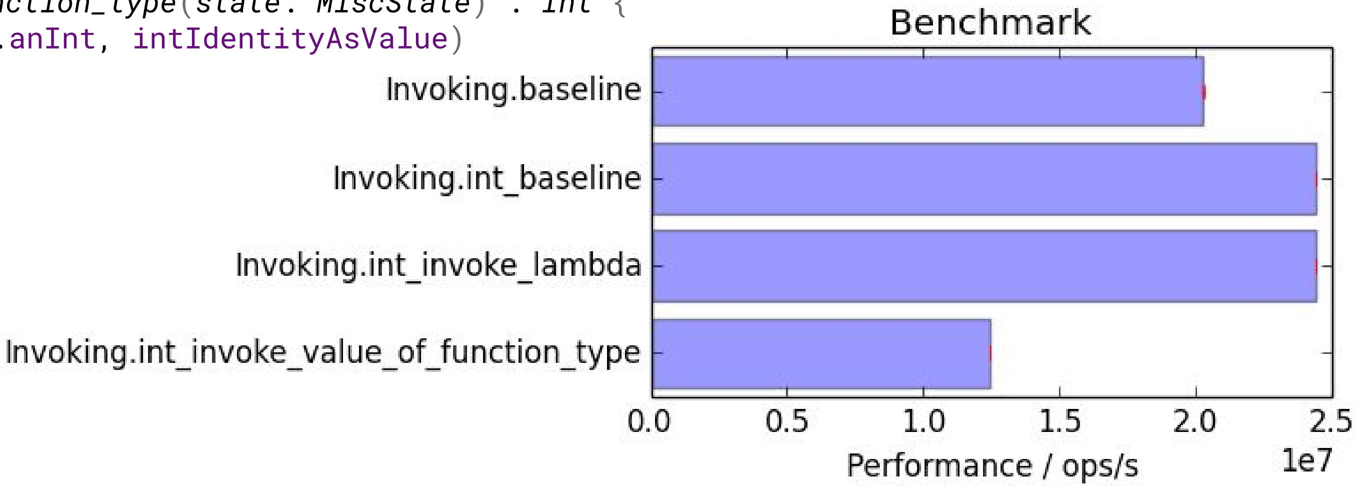
First-class Functions

```
public final invoke_value_of_function_type(LcostOfKotlin/invoking/MiscState;)Ljava/lang/String;
@Lorg/openjdk/jmh/annotations/Benchmark;()
@Lorg/jetbrains/annotations/NotNull;() // invisible
  @Lorg/jetbrains/annotations/NotNull;() // invisible, parameter 0
 L0
  ALOAD 1
  LDC "state"
  INVOKESTATIC kotlin/jvm/internal/Intrinsics.checkParameterIsNotNull (Ljava/lang/Object;Ljava/lang/String;)V
 L1
  LINENUMBER 30 L1
  ALOAD 1
  INVOKEVIRTUAL costOfKotlin/invoking/MiscState.getAString ()Ljava/lang/String;
  ASTORE 2
  INVOKESTATIC costOfKotlin/invoking/InvokingKt.getIdentityAsValue ()Lkotlin/jvm/functions/Function1;
  ASTORE 3
 L2
  LINENUMBER 60 L2
  ALOAD 3
  ALOAD 2
  INVOKEINTERFACE kotlin/jvm/functions/Function1.invoke (Ljava/lang/Object;)Ljava/lang/Object;
 L3
  CHECKCAST java/lang/String
  ARETURN
```



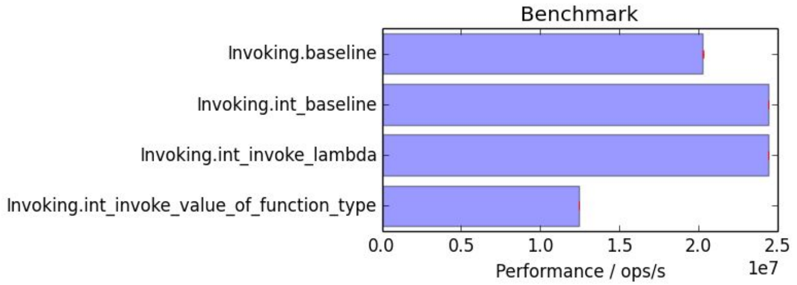
First-class Functions

```
@Benchmark
fun int_baseline(state: MiscState) : Int {
    return intIdentity(state.anInt)
}

@Benchmark
fun int_invoke_lambda(state: MiscState) : Int {
    return invokeWith(state.anInt) { intIdentity(it) }
}

@Benchmark
fun int_invoke_value_of_function_type(state: MiscState) : Int {
    return invokeWith(state.anInt, intIdentityAsValue)
}
```



First-class Functions

```
public final int_invoke_value_of_function_type(LcostOfKotlin/invoking/MiscState;)I
@Lorg/openjdk/jmh/annotations/Benchmark;()
  @Lorg/jetbrains/annotations/NotNull;() // invisible, parameter 0
 L0
  ALOAD 1
  LDC "state"
  INVOKESTATIC kotlin/jvm/internal/Intrinsics.checkParameterIsNotNull (Ljava/lang/Object;Ljava/lang/String;)V
 L1
  LINENUMBER 45 L1
  ALOAD 1
  INVOKEVIRTUAL costOfKotlin/invoking/MiscState.getAnInt ()I
  INVOKESTATIC java/lang/Integer.valueOf (I)Ljava/lang/Integer;
  ASTORE 2
  INVOKESTATIC costOfKotlin/invoking/InvokingKt.getIntIdentityAsValue ()Lkotlin/jvm/functions/Function1;
  ASTORE 3
 L2
  LINENUMBER 62 L2
  ALOAD 3
  ALOAD 2
  INVOKEINTERFACE kotlin/jvm/functions/Function1.invoke (Ljava/lang/Object;)Ljava/lang/Object;
 L3
  CHECKCAST java/lang/Number
  INVOKEVIRTUAL java/lang/Number.intValue ()I
  IRETURN
```



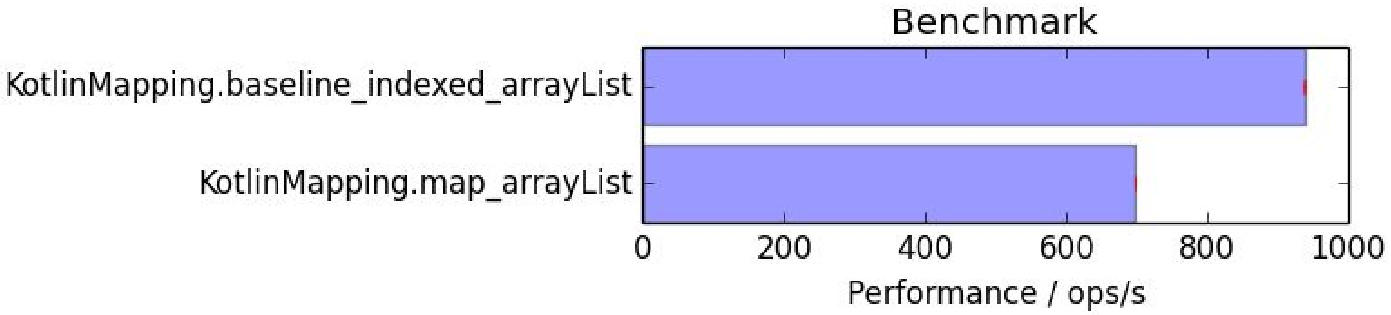
Mapping

Mapping

```
@Benchmark
fun map_arrayList(listState: ListState) =
   listState.arrayListOfStrings.map { it }
```

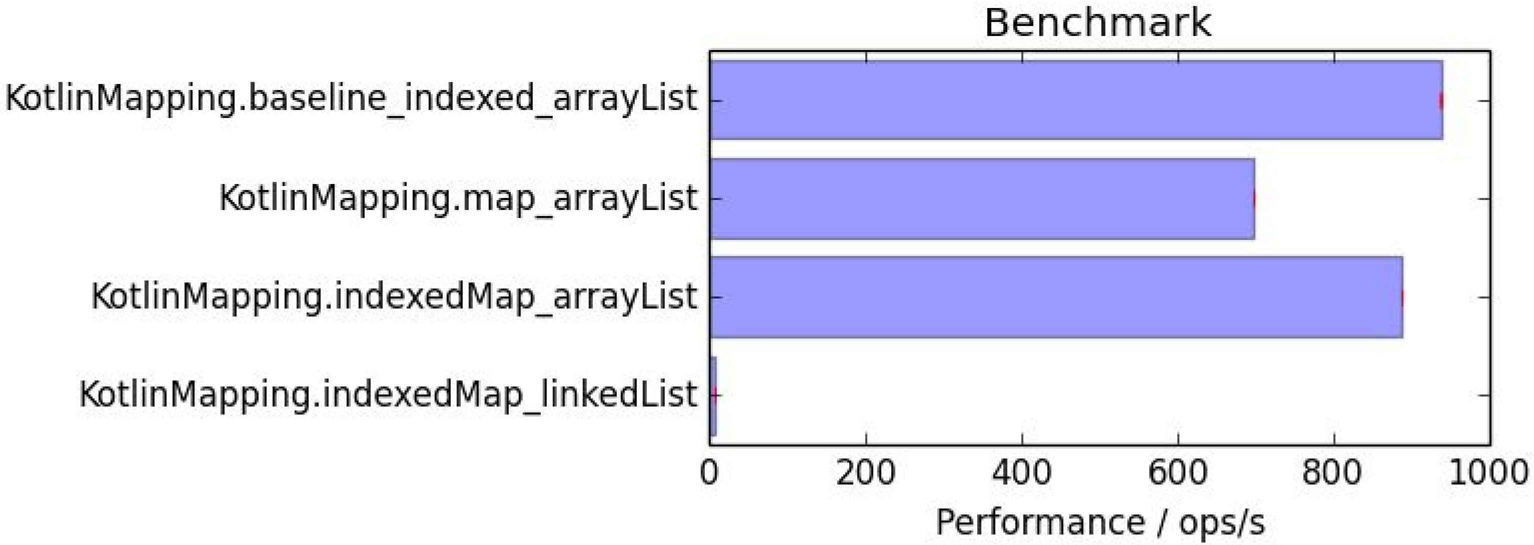
```
@Benchmark
fun baseline_indexed_arrayList(listState: ListState)
 : List<String> {
   val list = listState.arrayListOfStrings
   val result = ArrayList<String>(list.size)
   for (i in 0 until list.size) {
       result.add(list[i])
   }
   return result
}
```

```
public inline fun <T, R, C : MutableCollection<in R>> Iterable<T>.mapTo(destination: C, transform: (T)
-> R): C {
   for (item in this)
       destination.add(transform(item))
   return destination
}
```



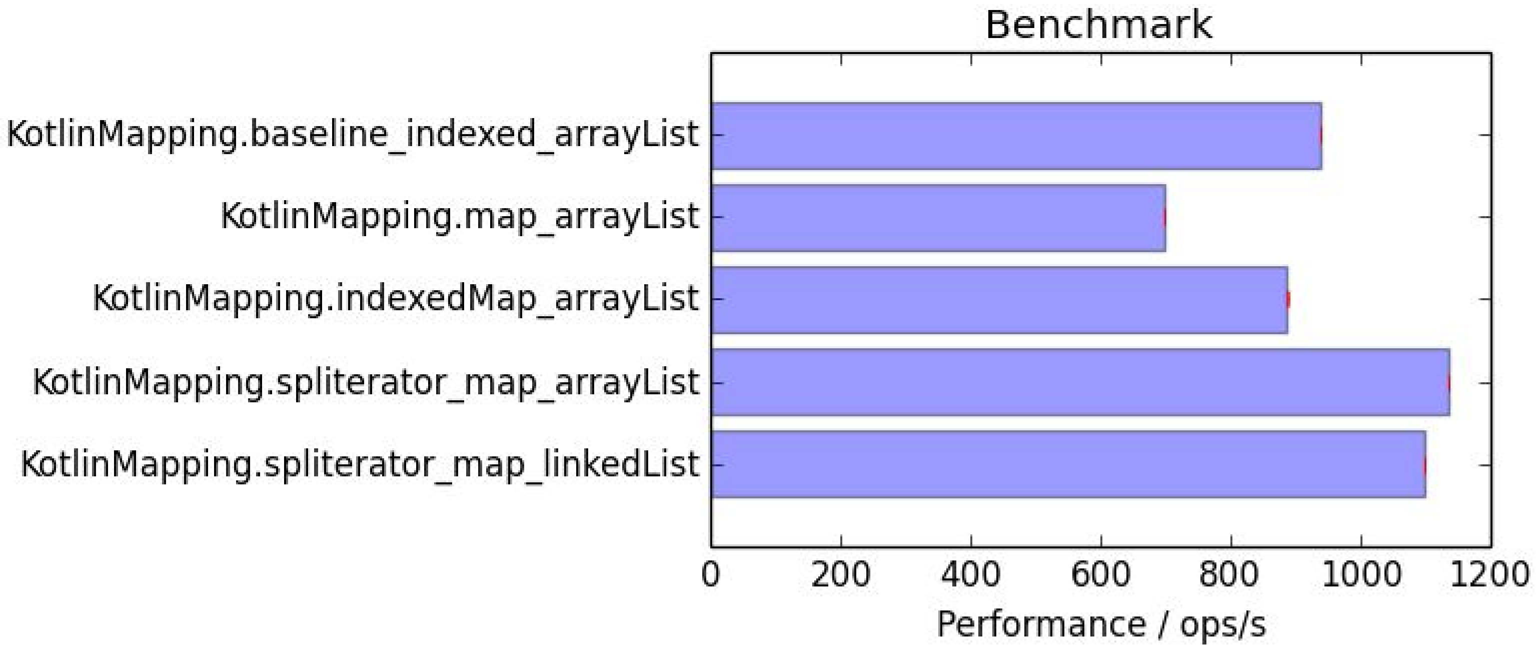
Mapping

```kotlin
inline fun <T, R> List<T>.indexedMap(transform: (T) -> R): List<R> {
   val result = ArrayList<R>(this.size)
   for (i in 0 until size) {
       result.add(transform(this.get(i)))
   }
   return result
}
```

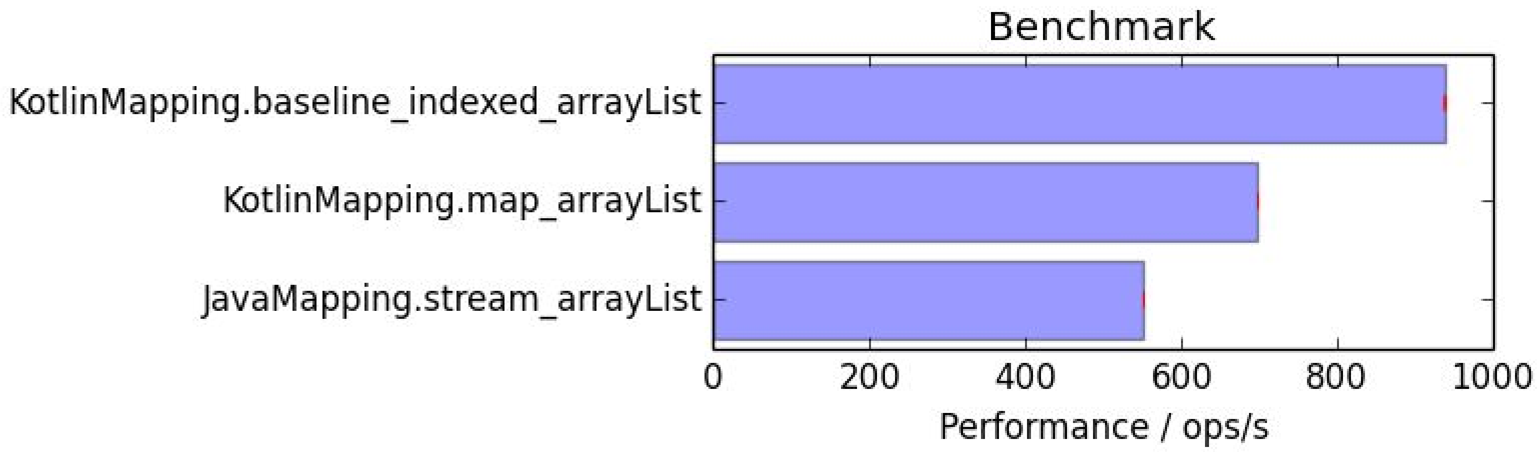


Mapping

```
inline fun <T, R> List<T>.spliteratorMap(crossinline transform: (T) -> R) : List<R>{
   val result = ArrayList<R>(this.size)
   spliterator().forEachRemaining() { result.add(transform(it)) }
   return result
}
```



Mapping

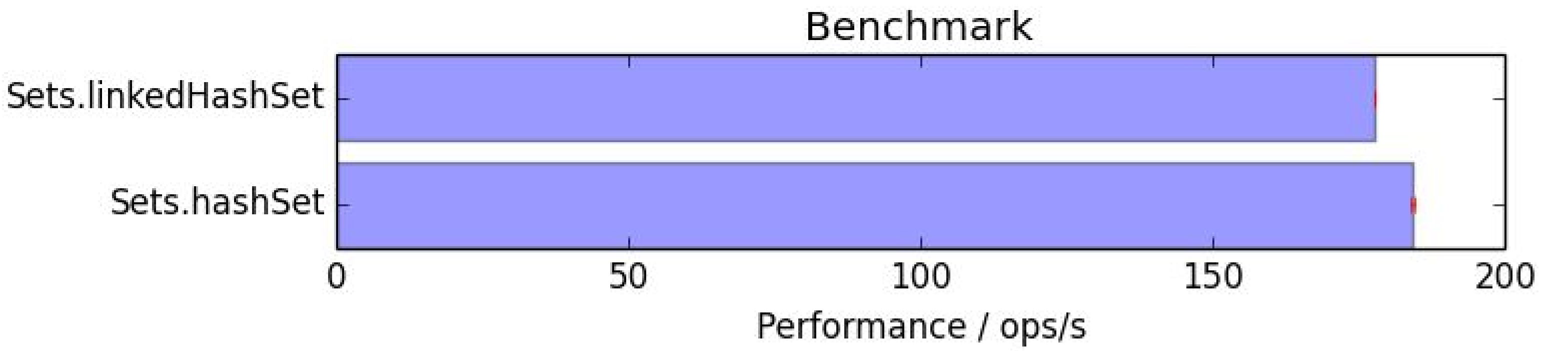

Benchmark
KotlinMapping.baseline_indexed_arrayList
KotlinMapping.map_arrayList
JavaMapping.stream_arrayList
0
200
400
600
800
1000
Performance / ops/s

Default Collections

Default Collections

```
@Benchmark
fun linkedHashSet(state: ObjectsState) =
    state.objects.toSet().map {
        it
    }
```

```
@Benchmark
fun hashSet(state: ObjectsState) =
    HashSet(state.objects).map {
        it
    }
```

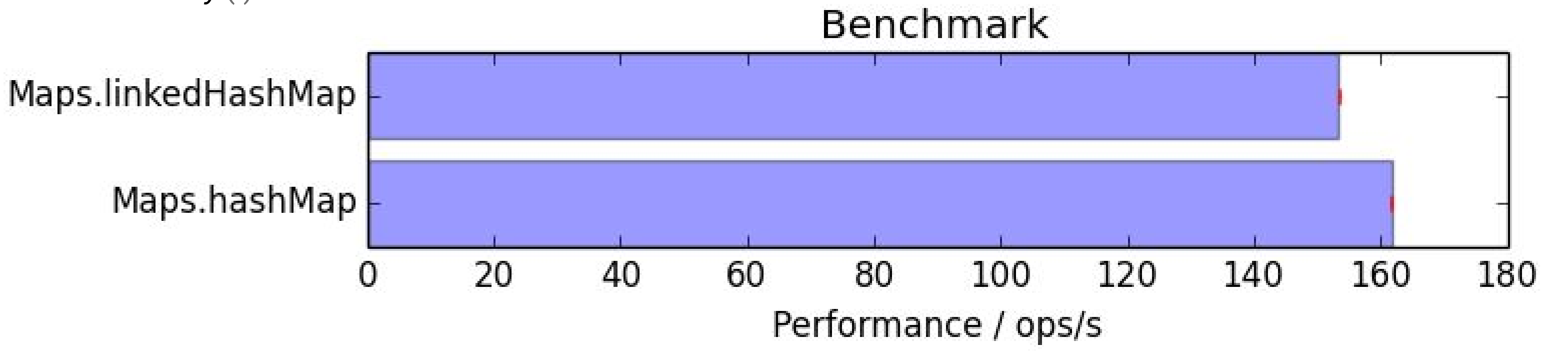


Default Collections

```
@Benchmark
fun linkedHashMap(state: ObjectsState): List<String> {
   return LinkedHashMap<String, String>(state.objects.size)
       .filledWith(state.objects)
       .everyValueTheHardWay()
}
```

```
private fun Map<String, String>.everyValueTheHardWay() =
    keys.map {
       this[it]!!
    }
```

```
@Benchmark
fun hashMap(state: ObjectsState): List<String> {
   return HashMap<String, String>(state.objects.size)
       .filledWith(state.objects)
       .everyValueTheHardWay()
}
```



Other Costs

Other Costs

- Compilation speed
- Code size

```
  @Lkotlin/Metadata;(mv={1, 1, 7}, bv={1, 0, 2}, k=1,
d1={"\u0000\u001a\n\u0002\u0018\u0002\n\u0002\u0010\u0000\n\u0002\u0008\u0002\n\u0002\u0010\u0
008\n\u0000\n\u0002\u0018\u0002\n\u0002\u0008\u0007\u0008\u0016\u0018\u00002\u00020\u0001B\u
0005\u00a2\u0006\u0002\u0010\u0002J\u0010\u0010\u0003\u001a\u00020\u00042\u0006\u0010\u0005\
u001a\u00020\u0006H\u0007J\u0010\u0010\u0007\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u0
0020\u0006H\u0007J\u0010\u0010\u0008\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u00020\u00
06H\u0007J\u0010\u0010\u0009\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u00020\u0006H\u00
07J\u0010\u0010\n\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u00020\u0006H\u0007J\u0010\u0
010\u000b\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u00020\u0006H\u0007J\u0010\u0010\u000
c\u001a\u00020\u00042\u0006\u0010\u0005\u001a\u00020\u0006H\u0007\u00a8\u0006\r"},
d2={"LcostOfKotlin/primitives/KotlinPrimitives;", "", "()V", "_1_baseline", "", "state",
"LcostOfKotlin/primitives/IntState;", "_2_sum", "_3_sum_nullable_bang_bang",
"_4_sum_elvis_never_null", "_5_sum_elvis_always_null", "_6_sum_elvis_50_50_nullable",
"_7_sum_elvis_90_10_nullable", "production sources for module kostings"})
```

Takeaways

Takeways

- Everything I examined is reassuringly OK
- Kotlin appears to favour safety and predictability over raw performance
- Inlining and HotSpot often mean that it doesn’t have to choose
    - but beware of primitives
    - and cold code
- You can make performance improvements
- I hope that you now know how to assess whether they work

Thank you!

- John Nolan for his statistical help
- You, for still being here

Christophe Beyls
    https://medium.com/@BladeCoder/exploring-kotlins-hidden-costs-part-1-fbb9935d9b62
Renato Athaydes
    https://sites.google.com/a/athaydes.com/renato-athaydes/posts/kotlinshiddencosts-benchmarks
Java Microbenchmark Harness
    http://openjdk.java.net/projects/code-tools/jmh/

This presentation
    https://docs.google.com/presentation/d/1wYX8RvspzQVxoGTlyagNqEyFGKIaTObSSz8CjoUpMks

Duncan McGregor
@duncanmcg

http://oneeyedmen.com
github.com/dmcg/kostings