The bytecode mumbo-jumbo

#perfmatters

@rrafols

#droidconIT

droidcon Italy // Torino

Agenda

- Disclaimer
- Who am I?
- Our friend the java compiler
- Language additions & things to consider
- Tooling

Disclaimer

This presentation contains bytecode

Who am I?

Mobile Engineering Manager

AXA Group Solutions Barcelona

Twitter: @rrafols

raimon.rafols@gmail.com

http://blog.rafols.org

We are hiring android developers!
Come and join us in Barcelona!

Our friend the java compiler

*.java → [javac] → *.class

*.class → [dx] → dex file

Change is coming!

Jack & Jill

I sense much fear in you.

*.java → [jack] → dex file

*.jar & *.aar → [jill] → jack library file

No java tooling!!

I sense much fear in you.

Javac vs other compilers

Compilers

Produces optimised code for target platform

Javac

Doesn't optimise anything

Javac

Doesn't know on which architecture will the code be executed

For the same reason

Java bytecode is stack based

Easy to interpret

But not the most optimal solution (regarding performance)

Quick example

Stack based integer addition

$$j = j + i$$

Java bytecode

```
iload_3
iload_2
iadd
istore_2
```

Java VM (JVM)

Only the JVM knows on which architecture is running

Java VM (JVM)

All optimisations are left to be done by the JVM

Maybe takes this concept a bit too far...

Imagine this simple C code

```
#include <stdio.h>
int main() {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + 6 + a;

  printf("%d\n", b);
}
```

GCC compiler

```
#include <stdio.h>
int main() {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + 6 + a;

  printf("%d\n", b);
}
```

```
…
movl $31, %esi
call _printf
…
```

* Using gcc & -O2 compiler option

javac

```
public static void main(String args[]) {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + 6 + a;

  System.out.println(b);
}
```

```
0: bipush        10
2: istore_1
3: bipush        21
5: iload_1
6: iadd
7: istore_2
...
```

Let's do a small change

```
#include <stdio.h>
int main() {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + a + 6;

  printf("%d\n", b);
}
```

GCC compiler

```
#include <stdio.h>
int main() {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + a + 6;

  printf("%d\n", b);
}
```

```
…
movl $31, %esi
call _printf
…
```

* Using gcc & -O2 compiler option

javac

```
public static void main(String args[]) {
  int a = 10;
  int b = 1 + 2 + 3 + 4 + 5 + a + 6;

  System.out.println(b);
}
```

```
0: bipush        10
2: istore_1
3: bipush        15
5: iload_1
6: iadd
7: bipush        6
9: iadd
10: istore_2

...
```

Let's do another quick change..

```
public static void main(String args[]) {
  int a = 10;
  int b = a + 1 + 2 + 3 + 4 + 5 + 6;

  System.out.println(b);
}
```

javac

```
public static void main(String args[]) {
  int a = 10;
  int b = a + 1 + 2 + 3 + 4 + 5 + 6;

  System.out.println(b);
}
```

```
0: bipush         10
2: istore_1
3: iload_1
4: iconst_1
5: iadd
6: iconst_2
7: iadd
8: iconst_3
9: iadd
10: iconst_4
11: iadd
12: iconst_5
13: iadd
14: bipush         6
16: iadd
17: istore_2
```

Java 8 to the rescue...

```
raimon$ javac -version
javac 1.8.0_45
```

javac

```
public static void main(String args[]) {
  int a = 10;
  int b = a + 1 + 2 + 3 + 4 + 5 + 6;

  System.out.println(b);
}
```

```
0: bipush          10
2: istore_1
3: iload_1
4: iconst_1
5: iadd
6: iconst_2
7: iadd
8: iconst_3
9: iadd
10: iconst_4
11: iadd
12: iconst_5
13: iadd
14: bipush          6
16: iadd
17: istore_2
```

Jack to the rescue...

jack

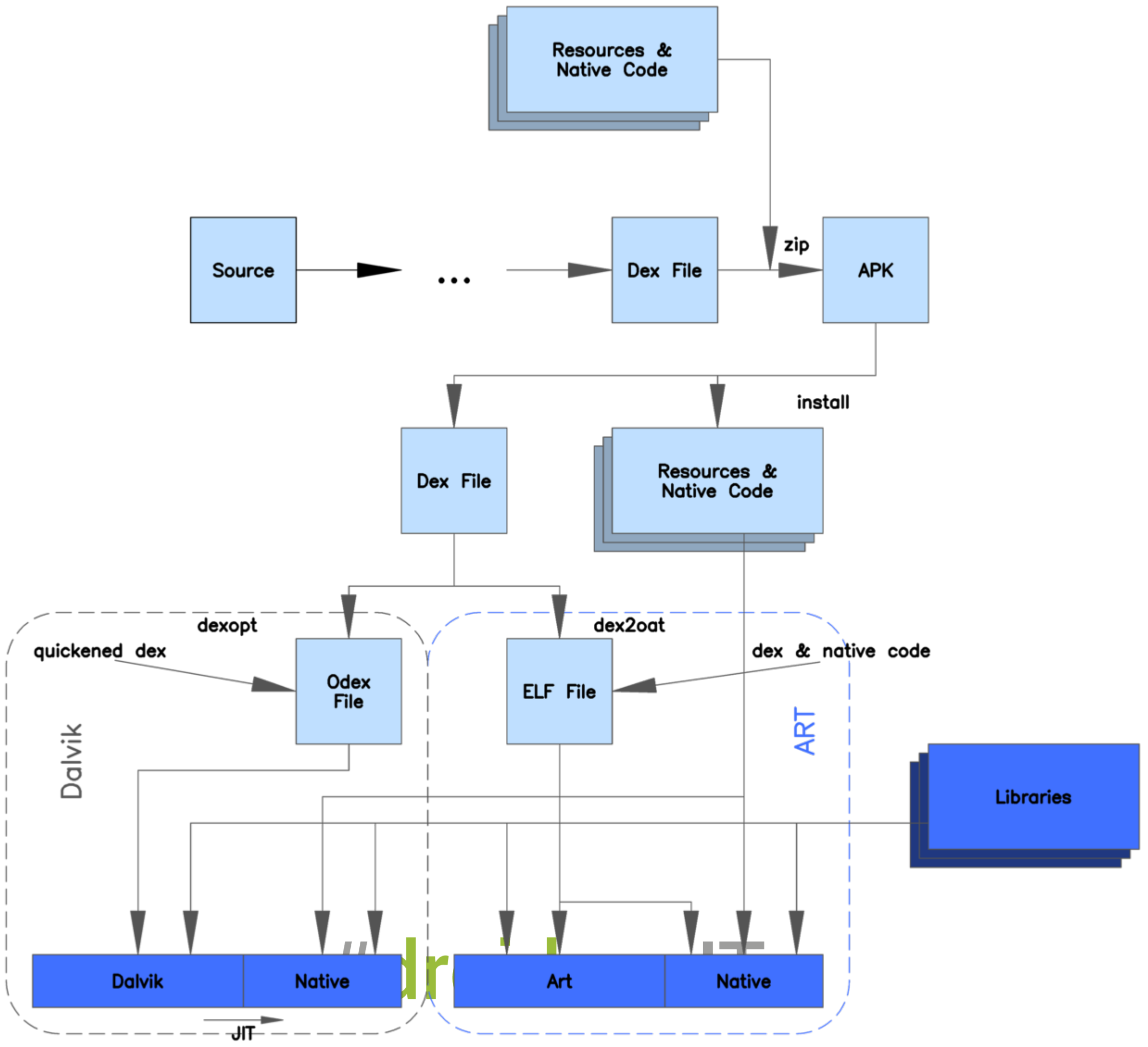
```
public static void main(String args[]) {
  int a = 10;
  int b = a + 1 + 2 + 3 + 4 + 5 + 6;

  System.out.println(b);
}
```

```
...
0: const/16 v0, #int 31
2: sget-object v1,
 Ljava/lang/System;
4: invoke-virtual {v1, v0}
7: return-void
...
```

Dalvik VM / ART

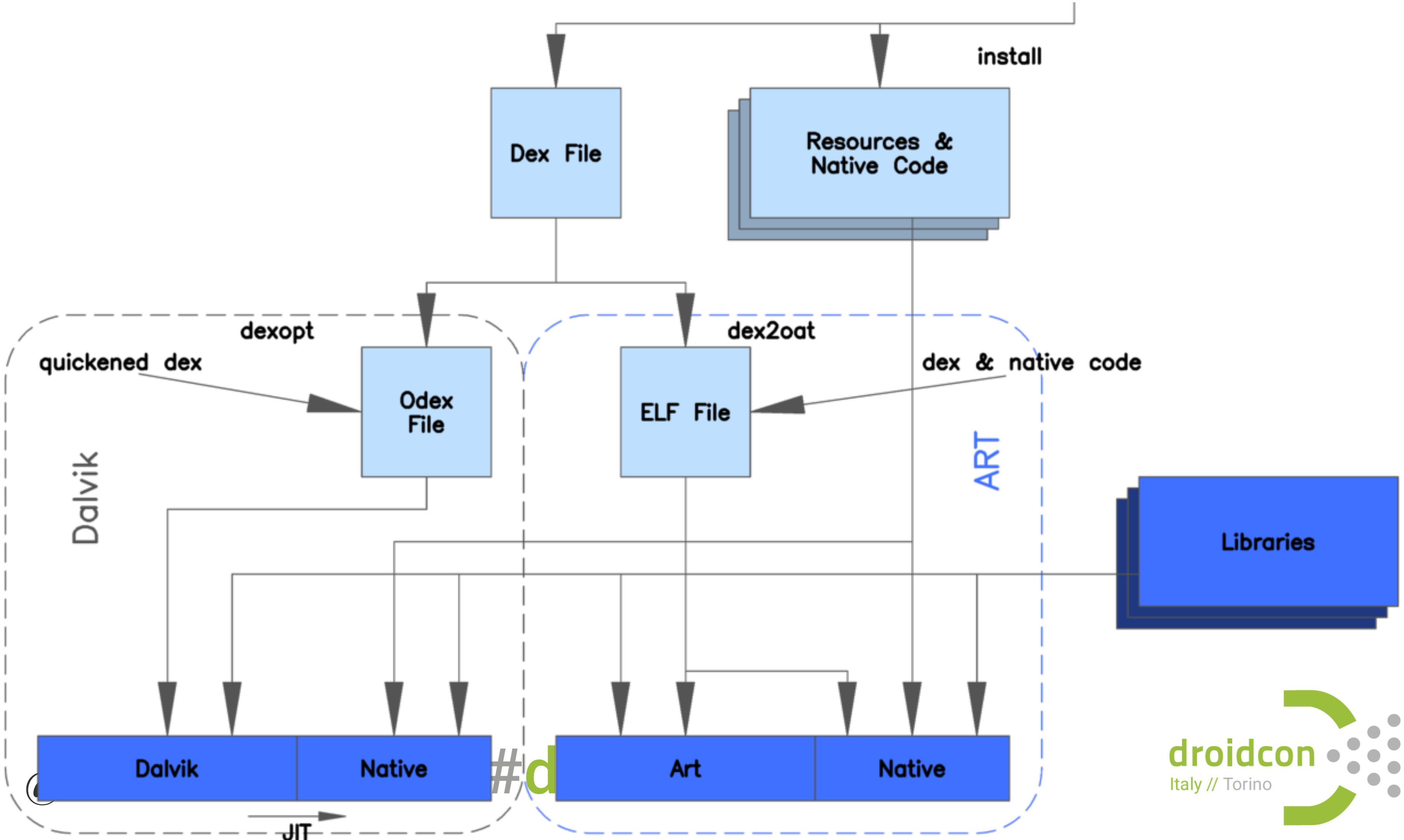
What about other “JVM”?

Dalvik VM / ART



@rrafols

droidcon Italy // Torino



droidcon Italy // Torino

Language additions

Thinks to consider

The Java compiler adds some code under the hood.

Autoboxing

Transparent to the developer but compiler adds some 'extra' code

Autoboxing

```
long total = 0;
for(int i = 0; i < N; i++) {
  total += i;
}
```

```
4: lconst_0
5: lstore_3
6: iconst_0
7: istore 5
9: iload 5
11: ldc #6;
13: if_icmpge 28
16: lload_3
17: iload 5
19: i2l
20: ladd
21: lstore_3
22: iinc 5,1
25: goto 9
```

Autoboxing

```
Long total = 0;
for(Integer i = 0; i < N; i++) {
  total += i;
}
```

```
9: iconst_0
10: invokestatic #4; //Method java/lang/Integer.valueOf
13: astore 4
15: aload 4
17: invokevirtual #5; //Method java/lang/Integer.intVal
20: ldc #6; //int 10000000
22: if_icmpge 65
25: aload_3
26: invokevirtual #7; //Method java/lang/Long.longValue
29: aload 4
31: invokevirtual #5; //Method java/lang/Integer.intVal
34: i2l
35: ladd
36: invokestatic #3; //Method java/lang/Long.valueOf:(J
39: astore_3
40: aload 4
42: astore 5
44: aload 4
46: invokevirtual #5; //Method java/lang/Integer.intVal
49: iconst_1
50: iadd
51: invokestatic #4; //Method java/lang/Integer.valueOf
54: dup
55: astore 4
57: astore 6
59: aload 5
61: pop
62: goto 15
```

Autoboxing

- This is what that code is actually doing:

```
Long total = 0;
for(Integer i = Integer.valueOf(0);
            i.intValue() < N;
            i = Integer.valueOf(i.intValue() + 1)) {

  total = Long.valueOf(total.longValue() + (long)i.intValue())
}
```

Autoboxing

Jack does not help in this situation

dex file contains same code

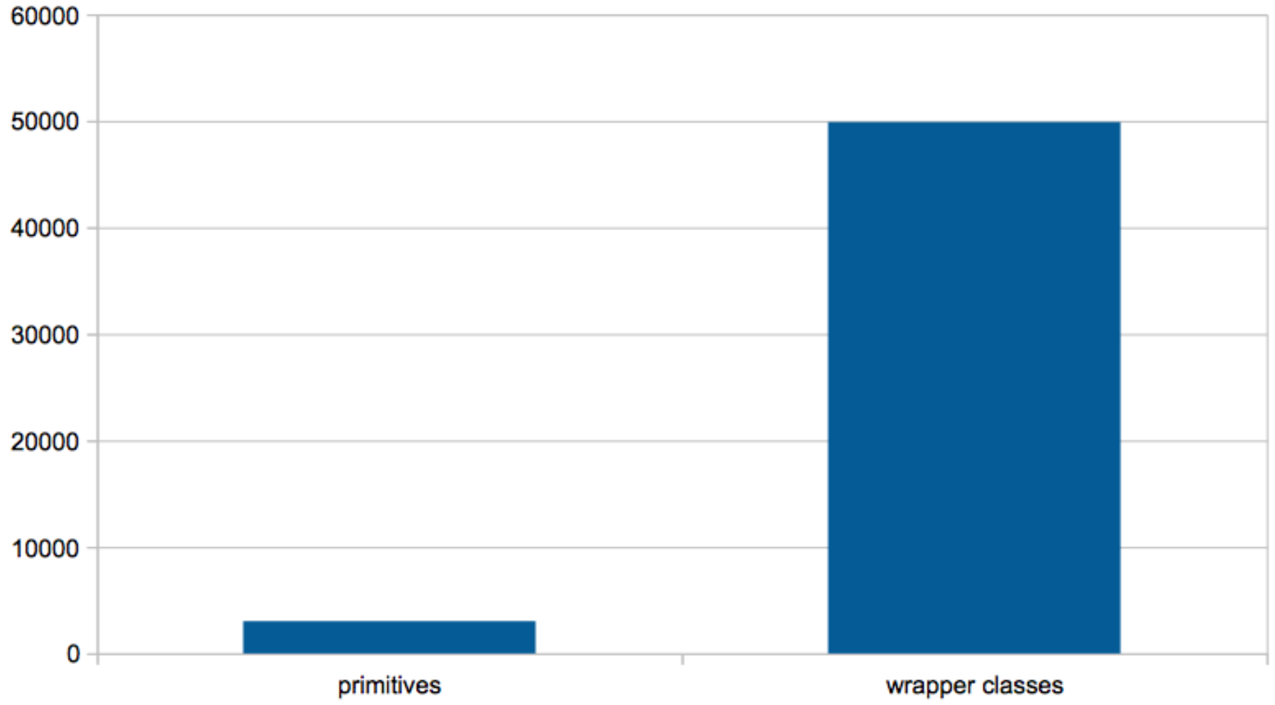
Autoboxing

Let's run that loop

10.000.000.000 times

(on my desktop computer)

Autoboxing


60000
50000
40000
30000
20000
10000
0
primitives
wrapper classes

Autoboxing

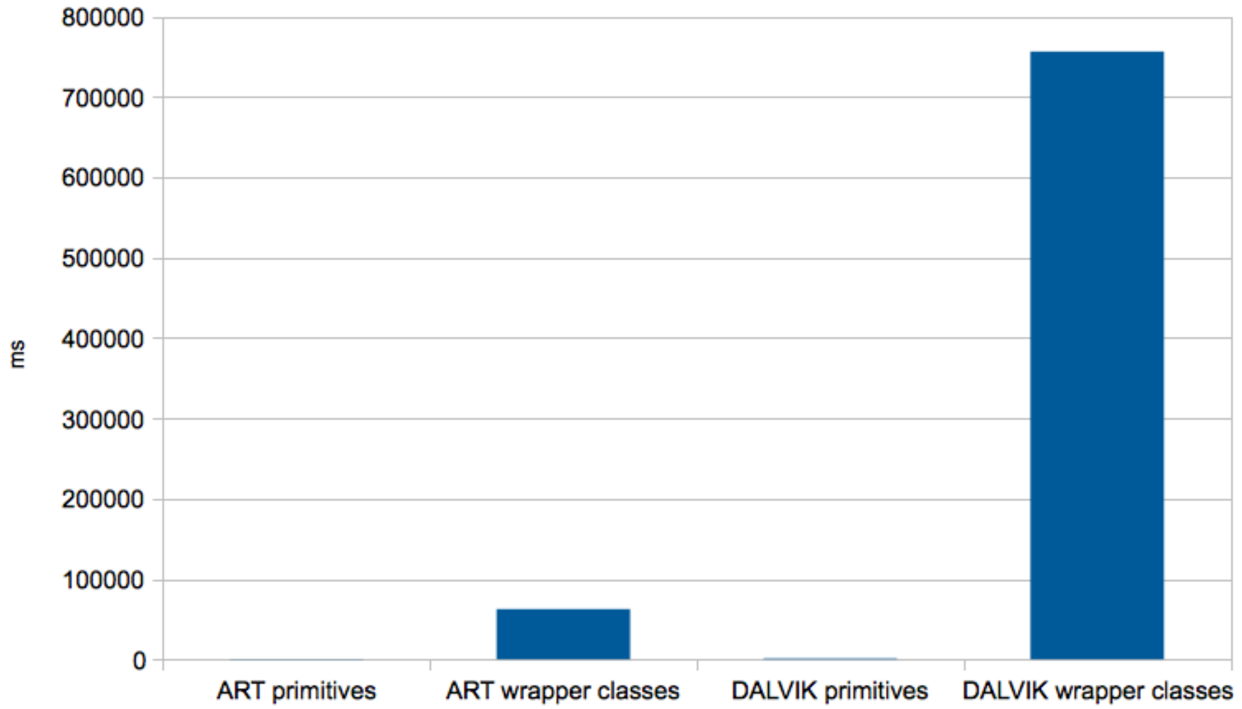
Let's run that loop 100.000.000

Times on two Nexus 5

KitKat & Lollipop

Dalvik VM & ART

Autoboxing



800000
700000
600000
500000
400000
300000
200000
100000
0
ms
ART primitives
ART wrapper classes
DALVIK primitives
DALVIK wrapper classes

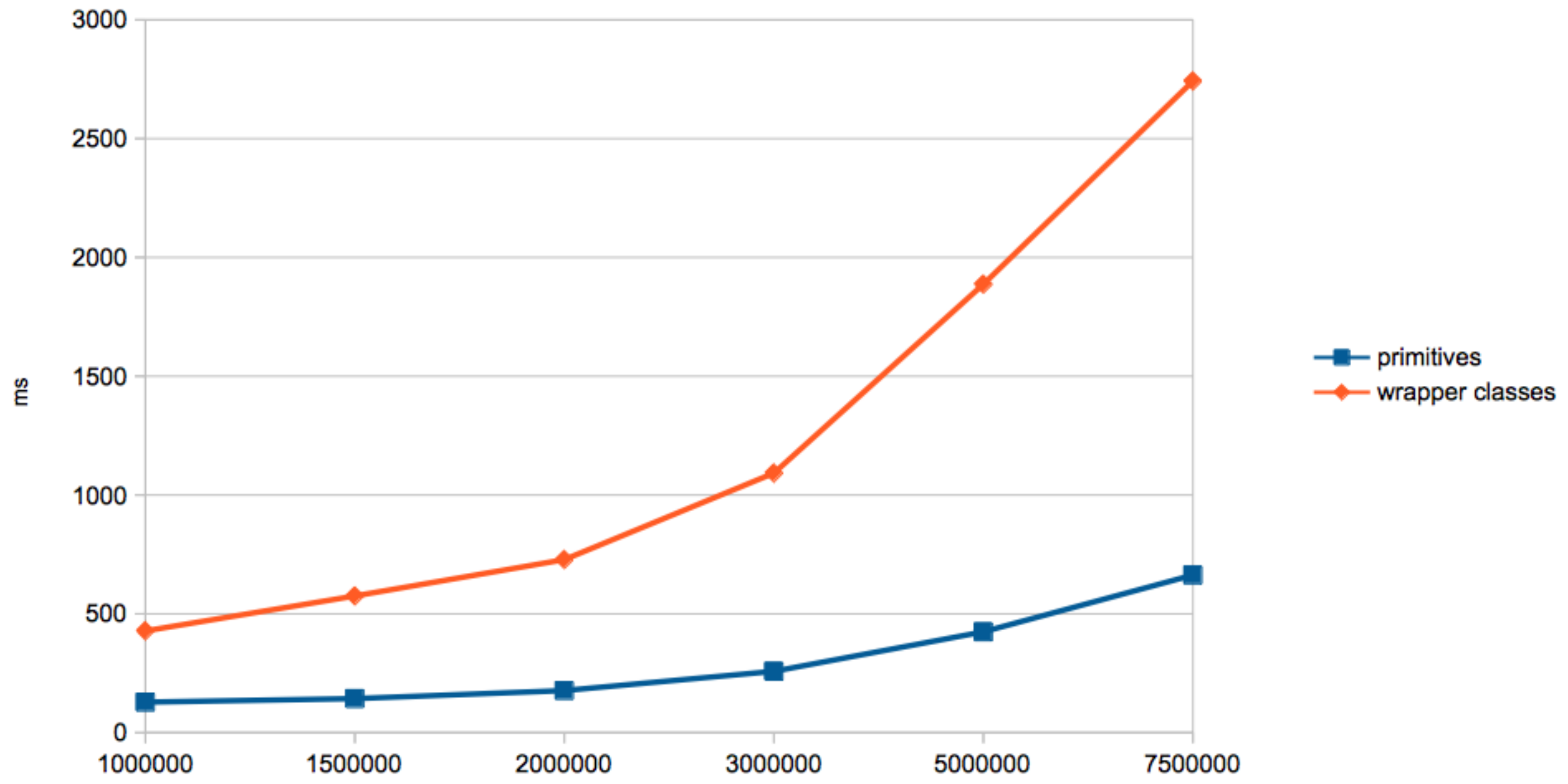
Sorting

The easy way

Let's sort some numbers

Arrays.sort(...)

Difference between sorting primitive types & objects


ms
3000
2500
2000
1500
1000
500
0
1000000
1500000
2000000
3000000
5000000
7500000
primitives
wrapper classes

Sorting objects is a stable sort

Default java algorithm: TimSort
(derived from MergeSort)

Sorting primitives doesn't require to be stable sort

Default java algorithm:
Dual-Pivot quicksort

Sorting

Use primitive types as much as possible

Loops

What's going on behind the scenes

Loops - List

```
ArrayList<Integer> list = new …
static long loopStandardList() {
    long result = 0;
    for(int i = 0; i < list.size(); i++) {
        result += list.get(i);
    }
    return result;
}
```

Loops - List (Java bytecode)

```
7: lload_0
8: getstatic      #26       // Field list:Ljava/util/ArrayList;
11: iload_2
12: invokevirtual #54       // Method java/util/ArrayList.get:(I)Ljava/lang/Object;
15: checkcast     #38       // class java/lang/Integer
18: invokevirtual #58       // Method java/lang/Integer.intValue:()I
21: i2l
22: ladd
23: lstore_0
24: iinc          2, 1
27: iload_2
28: getstatic     #26       // Field list:Ljava/util/ArrayList;
31: invokevirtual #61       // Method java/util/ArrayList.size:()I
34: if_icmplt     7
```

Loops - foreach

```
ArrayList<Integer> list = new …
static long loopForeachList() {
    long result = 0;
    for(int v : list) {
        result += v;
    }
    return result;
}
```

Loops - foreach (Java bytecode)

```
12: aload_3
13: invokeinterface #70,  1   // InterfaceMethod java/util/Iterator.next:()
18: checkcast      #38        // class java/lang/Integer
21: invokevirtual #58         // Method java/lang/Integer.intValue:()I
24: istore_2
25: lload_0
26: iload_2
27: i2l
28: ladd
29: lstore_0
30: aload_3
31: invokeinterface #76,  1   // InterfaceMethod java/util/Iterator.hasNext:()Z
36: ifne           12
```

Loops - Array

```
static int[] array = new ...
static long loopStandardArray() {
    long result = 0;
    for(int i = 0; i < array.length; i++) {
        result += array[i];
    }
    return result;
}
```

Loops - Array (Java bytecode)

```
7: lload_0
8: getstatic      #28                 // Field array:[I
11: iload_2
12: iaload
13: i2l
14: ladd
15: lstore_0
16: iinc           2, 1
19: iload_2
20: getstatic      #28                 // Field array:[I
23: arraylength
24: if_icmplt      7
```

Loops - size cached

```
static int[] array = new ...
static long loopStandardArraySizeStored() {
    long result = 0;   int length = array.length;
    for(int i = 0; i < length; i++) {
        result += array[i];
    }
    return result;
}
```

Loops - size stored (Java bytecode)

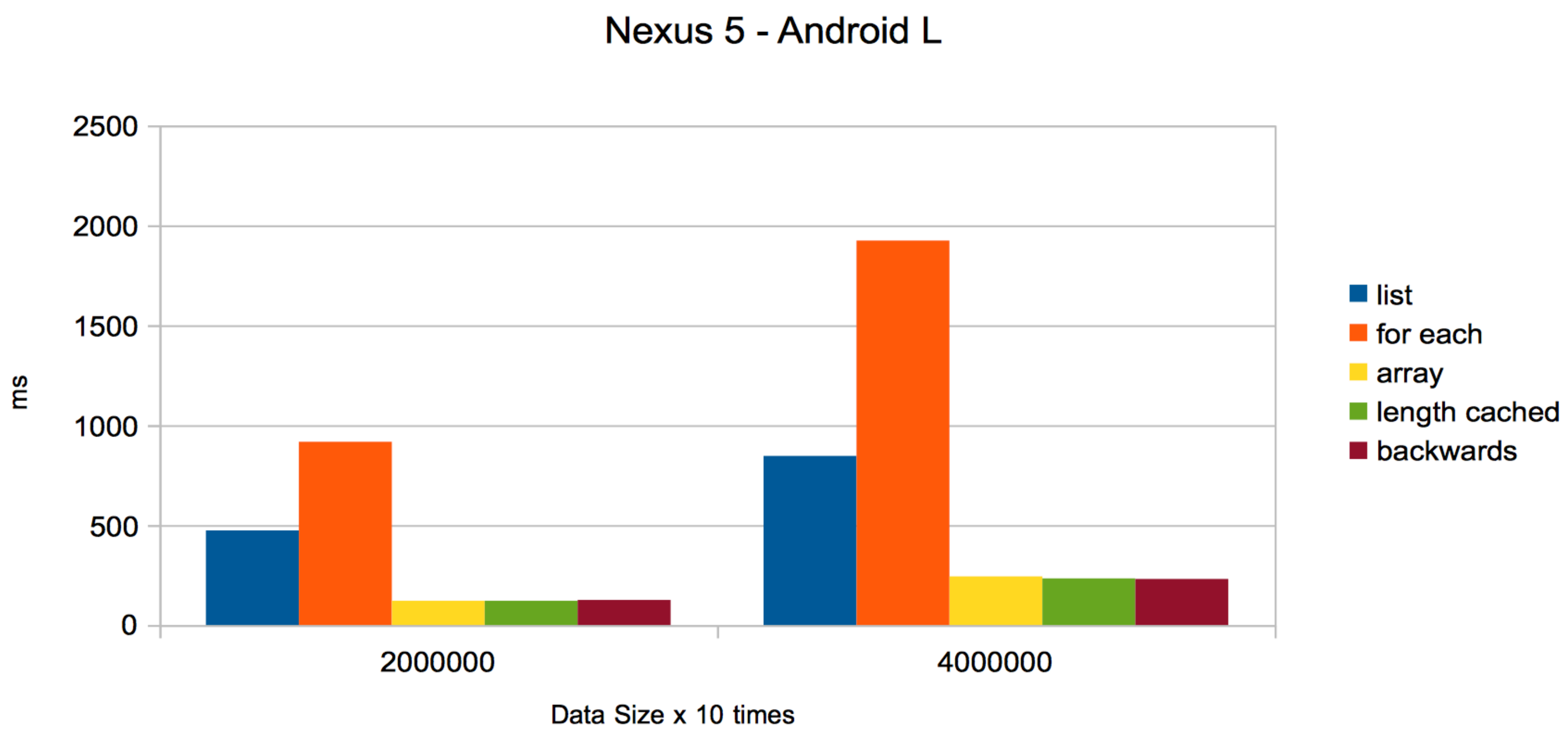
```
12: lload_0
13: getstatic      #28                   // Field array:[I
16: iload_3
17: iaload
18: i2l
19: ladd
20: lstore_0
21: iinc           3, 1
24: iload_3
25: iload_2
26: if_icmplt      12
```

Loops - backwards

```
static int[] array = new ...
static long loopStandardArrayBackwards() {
  long result = 0;
  for(int i = array.length - 1; i >= 0; i--) {
      result += array[i];
  }
  return result;
}
```

Loops - backwards (Java bytecode)

```
12: lload_0
13: getstatic     #28                 // Field array:[I
16: iload_2
17: iaload
18: i2l
19: ladd
20: lstore_0
21: iinc          2, -1
24: iload_2
25: ifge          12
```


Nexus 5 - Android L
ms
2500
2000
1500
1000
500
0
2000000
4000000
Data Size x 10 times
list
for each
array
length cached
backwards

Loops

Avoid foreach constructions if performance is a requirement

Calling a method

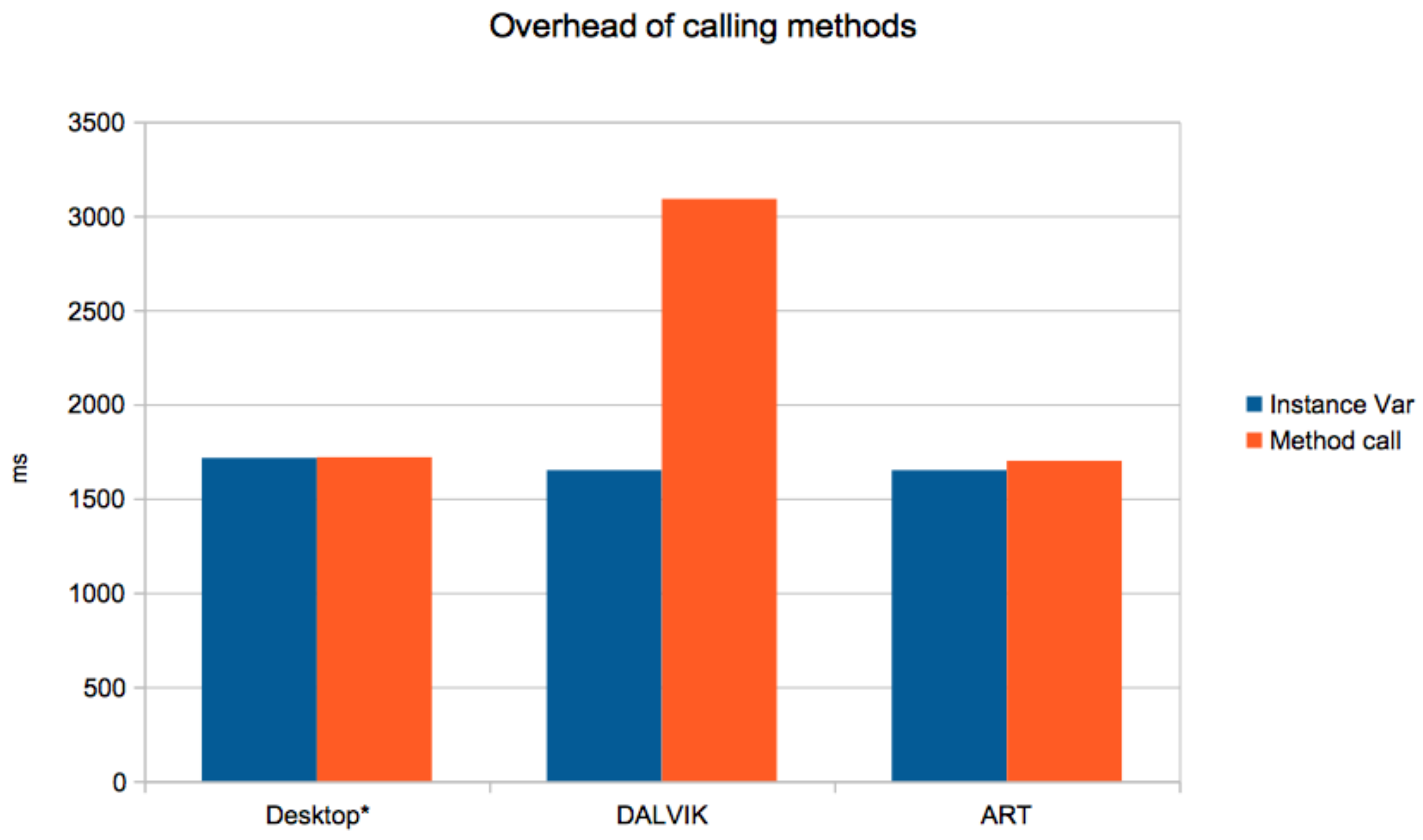
Is there an overhead?

Calling a method overhead

```
for(int i = 0; i < N; i++) {
    setVal(getVal() + 1);
}
```

VS

```
for(int i = 0; i < N; i++) {
    val = val + 1;
}
```


Overhead of calling methods
3500
3000
2500
2000
1500
1000
500
0
ms
Desktop*
DALVIK
ART
Instance Var
Method call

String concatenation

The evil + sign

String concatenation

```
String str = "";
for(int i = 0;  i  < ITERATIONS; i++) {
  str += ANY_OTHER_STRING;
}
```

String concatenation

```
8: new             #26            // class java/lang/StringBuilder
11: dup
12: aload_1
13: invokestatic   #28            // Method java/lang/String.valueOf:(Ljava/lang/Object;)Ljava/lang/
  String;
16: invokespecial  #34            // Method java/lang/StringBuilder."<init>":(Ljava/lang/String;)V
19: ldc            #11            // String ANY_OTHER_STRING
21: invokevirtual  #37            // Method java/lang/StringBuilder.append:(Ljava/lang/String;)
24: invokevirtual  #41            // Method java/lang/StringBuilder.toString:()Ljava/lang/String;
27: astore_1
28: iinc           2, 1
31: iload_2
32: bipush         ITERATIONS
34: if_icmplt      8
```

String concatenation

```
String str = "";
for(int i = 0;  i  < ITERATIONS; i++) {
    StringBuilder sb = new StringBuilder(String.valueOf(str));
    sb.append(ANY_OTHER_STRING);
    str = sb.toString();
}
```

String concatenation
alternatives

String.concat()

- Concat cost is O(N) + O(M)
- Concat returns a new String Object.

```
String str = "";
for(int i = 0;  i  < ITERATIONS; i++) {
    str = str.concat(ANY_OTHER_STRING);
}
```

StringBuilder

- StringBuffer.append cost is O(M) amortized time (M length of appended String)
- Avoids creation of new objects.

```
StringBuilder sb = new StringBuilder()
    for(int i = 0;  i  < ITERATIONS; i++) {
        sb.append(ANY_OTHER_STRING);
    }
    str = sb.toString();
```

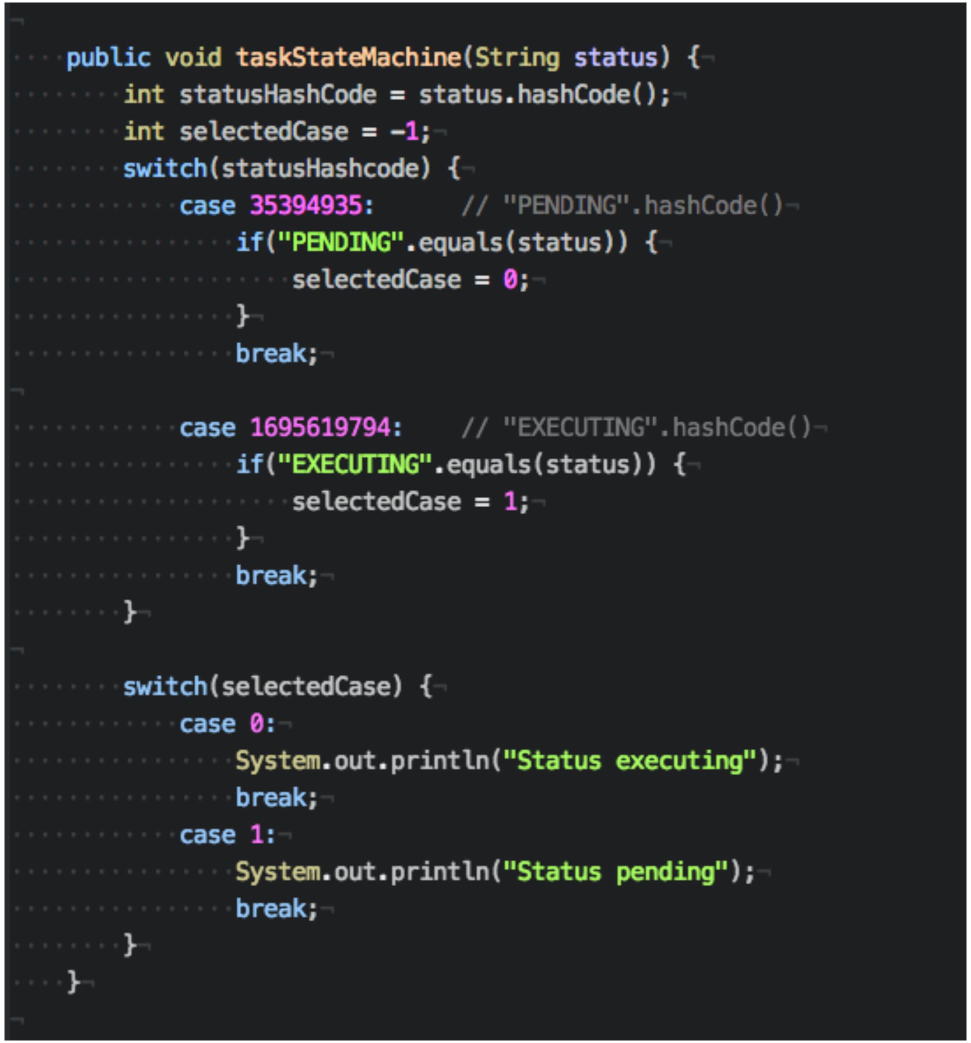
String concatenation

Use StringBuilder (properly) as much as possible. StringBuffer is the thread safe implementation.

Strings in case statements

```
public void taskStateMachine(String status) {
        switch(status) {
            case "PENDING":
                System.out.println("Status pending");
                break;

            case "EXECUTING":
                System.out.println("Status executing");
                break;
        }
}
```

```
Code:
   0: aload_1
   1: astore_2
   2: iconst_m1
   3: istore_3
   4: aload_2
   5: invokevirtual #2                  // Method java/lang/String.hashCode:()I
   8: lookupswitch  { // 2
        35394935: 36
      1695619794: 50
         default: 61
    }
  36: aload_2
  37: ldc           #3                  // String PENDING
  39: invokevirtual #4                  // Method java/lang/String.equals:(Ljava/lang/Object;)Z
  42: ifeq          61
  45: iconst_0
  46: istore_3
  47: goto          61
  50: aload_2
  51: ldc           #5                  // String EXECUTING
  53: invokevirtual #4                  // Method java/lang/String.equals:(Ljava/lang/Object;)Z
  56: ifeq          61
  59: iconst_1
  60: istore_3
  61: iload_3
  62: lookupswitch  { // 2
               0: 88
               1: 99
         default: 107
    }
  88: getstatic     #6                  // Field java/lang/System.out:Ljava/io/PrintStream;
  91: ldc           #7                  // String Status pending
  93: invokevirtual #8                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
  96: goto          107
  99: getstatic     #6                  // Field java/lang/System.out:Ljava/io/PrintStream;
 102: ldc           #9                  // String Status executing
 104: invokevirtual #8                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
 107: return
```



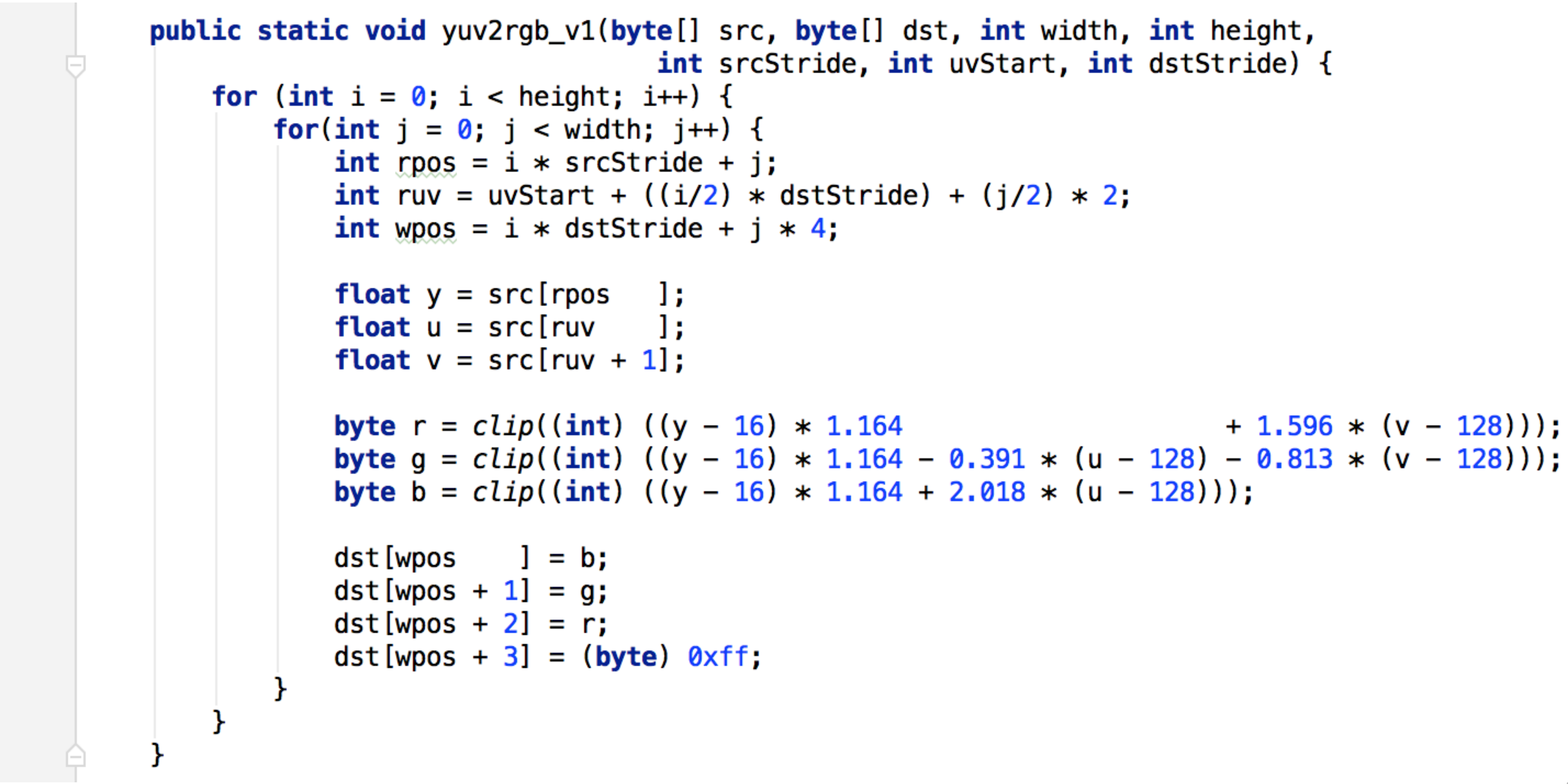
```
    public void taskStateMachine(String status) {
        int statusHashCode = status.hashCode();
        int selectedCase = -1;
        switch(statusHashcode) {
            case 35394935:      // "PENDING".hashCode()
                if("PENDING".equals(status)) {
                    selectedCase = 0;
                }
                break;

            case 1695619794:    // "EXECUTING".hashCode()
                if("EXECUTING".equals(status)) {
                    selectedCase = 1;
                }
                break;
        }

        switch(selectedCase) {
            case 0:
                System.out.println("Status executing");
                break;
            case 1:
                System.out.println("Status pending");
                break;
        }
    }
```

@rrafols

droidcon
Italy // Torino

Complex example

yuv2rgb



```java
public static void yuv2rgb_v1(byte[] src, byte[] dst, int width, int height,
                              int srcStride, int uvStart, int dstStride) {
    for (int i = 0; i < height; i++) {
        for(int j = 0; j < width; j++) {
            int rpos = i * srcStride + j;
            int ruv = uvStart + ((i/2) * dstStride) + (j/2) * 2;
            int wpos = i * dstStride + j * 4;

            float y = src[rpos    ];
            float u = src[ruv     ];
            float v = src[ruv + 1];

            byte r = clip((int) ((y - 16) * 1.164                   + 1.596 * (v - 128)));
            byte g = clip((int) ((y - 16) * 1.164 - 0.391 * (u - 128) - 0.813 * (v - 128)));
            byte b = clip((int) ((y - 16) * 1.164 + 2.018 * (u - 128)));

            dst[wpos    ] = b;
            dst[wpos + 1] = g;
            dst[wpos + 2] = r;
            dst[wpos + 3] = (byte) 0xff;
        }
    }
}
```

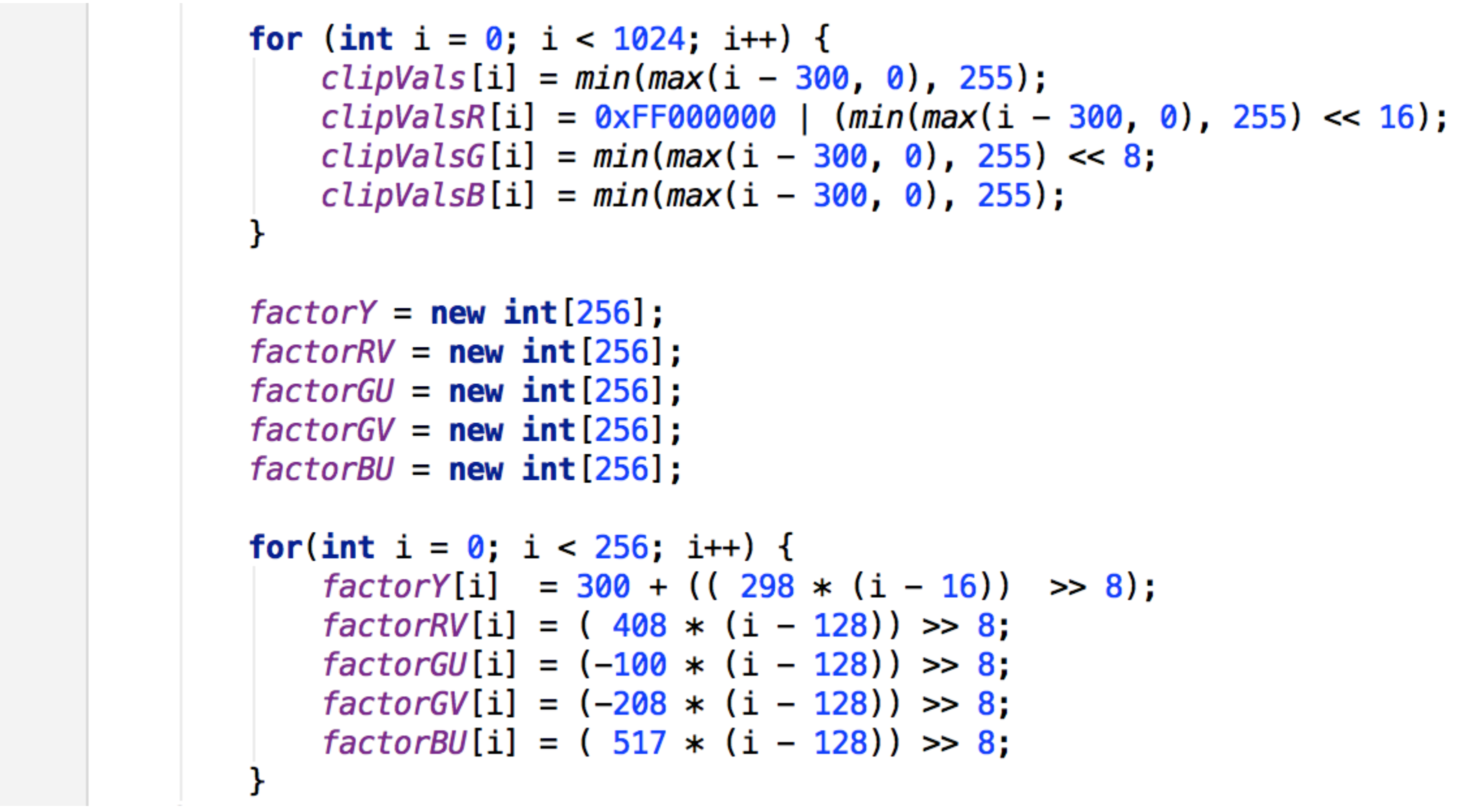
@rrafols

#droidconIT

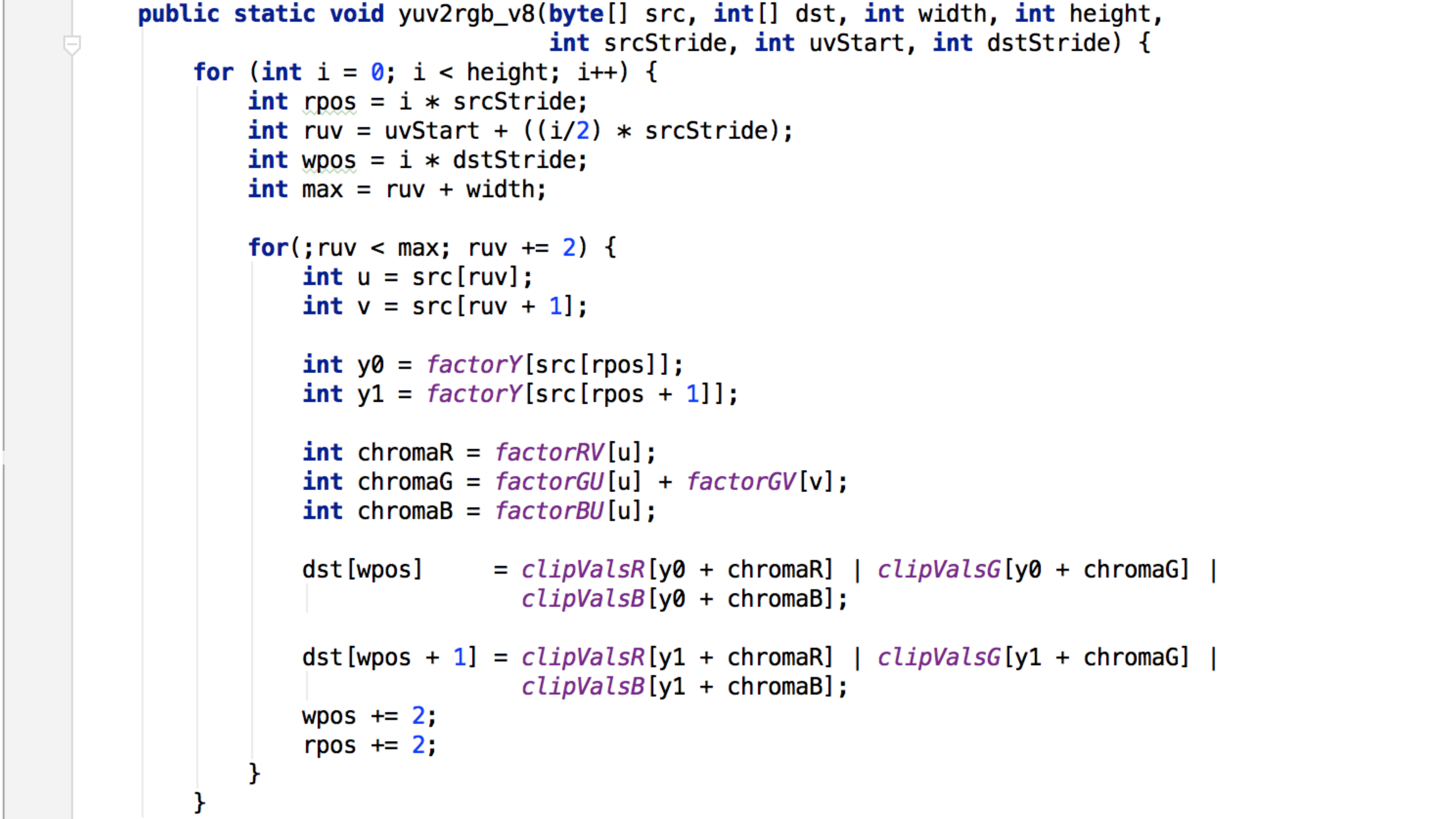
droidcon
Italy // Torino

Slightly optimized version

precalc tables, 2 pixels per loop



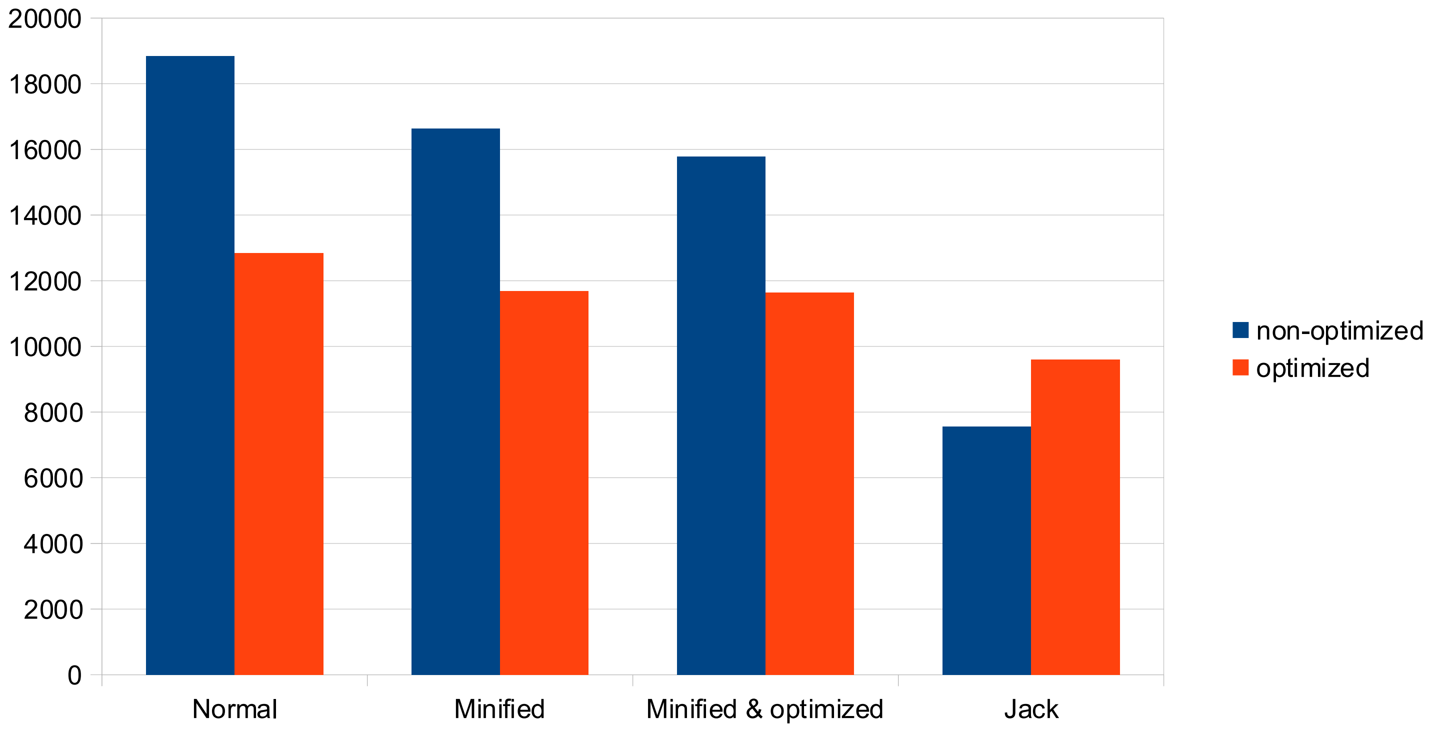
```
for (int i = 0; i < 1024; i++) {
    clipVals[i] = min(max(i - 300, 0), 255);
    clipValsR[i] = 0xFF000000 | (min(max(i - 300, 0), 255) << 16);
    clipValsG[i] = min(max(i - 300, 0), 255) << 8;
    clipValsB[i] = min(max(i - 300, 0), 255);
}

factorY = new int[256];
factorRV = new int[256];
factorGU = new int[256];
factorGV = new int[256];
factorBU = new int[256];

for(int i = 0; i < 256; i++) {
    factorY[i]  = 300 + (( 298 * (i - 16))  >> 8);
    factorRV[i] = ( 408 * (i - 128)) >> 8;
    factorGU[i] = (-100 * (i - 128)) >> 8;
    factorGV[i] = (-208 * (i - 128)) >> 8;
    factorBU[i] = ( 517 * (i - 128)) >> 8;
}
```



```java
public static void yuv2rgb_v8(byte[] src, int[] dst, int width, int height,
                              int srcStride, int uvStart, int dstStride) {
    for (int i = 0; i < height; i++) {
        int rpos = i * srcStride;
        int ruv = uvStart + ((i/2) * srcStride);
        int wpos = i * dstStride;
        int max = ruv + width;

        for(;ruv < max; ruv += 2) {
            int u = src[ruv];
            int v = src[ruv + 1];

            int y0 = factorY[src[rpos]];
            int y1 = factorY[src[rpos + 1]];

            int chromaR = factorRV[u];
            int chromaG = factorGU[u] + factorGV[v];
            int chromaB = factorBU[u];

            dst[wpos]     = clipValsR[y0 + chromaR] | clipValsG[y0 + chromaG] |
                            clipValsB[y0 + chromaB];

            dst[wpos + 1] = clipValsR[y1 + chromaR] | clipValsG[y1 + chromaG] |
                            clipValsB[y1 + chromaB];
            wpos += 2;
            rpos += 2;
        }
    }
```

Lets compare:

Normal, minified, minified with optimizations & jack


20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Normal
Minified
Minified & optimized
Jack
🟦 non-optimized
🟥 optimized

Tooling

Tooling - Disassembler

Java

- javap -c <classfile>

Android:

- Dexdump -d <dexfile>

Tooling – Disassembler - ART

```
adb pull /data/dalvik-cache/arm/data@app@<package>-1@base.apk@classes.dex
```

```
gobjdump -D <file>
```

Tooling – Disassembler - ART

```
adb shell oatdump --oat-file=/data/dalvik-cache/arm/data@app@<package>-1@base.apk@classes.dex
```

Performance measurements

Avoid doing multiple tests in one run

JIT might be evil!

Do not trust the compiler!

@rrafols

http://blog.rafols.org

droidcon Italy // Torino

#droidconIT