ORACLE

The Lord of the Strings

«Как я провёл лето»

Aleksey Shipilëv
aleksey.shipilev@oracle.com, @shipilev

The 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.

Введение

Введение: ожидания

Этот доклад в основном про опыт разработки двух сложных JDK/JVM фич:

- **Compact Strings**: сжимаем в однобайтовые символы
- **Indify String Concat**: перетаскиваем конкатенацию в рантайм

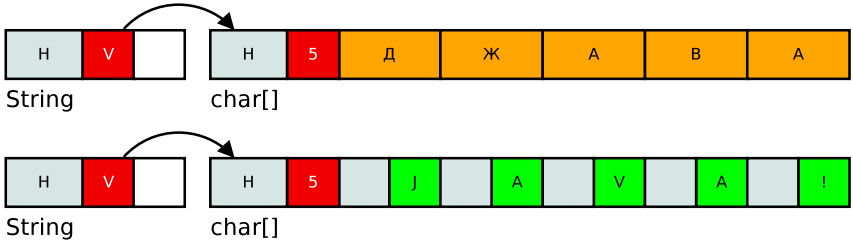
Про каждую фичу рассказываем в трёх подчастях:

1. **Инсайт**: основная идея и радостные повизгивания
2. **Ангст**: сколько седых ангелов танцует на конце JDK-девелопмент-иглы
3. **Катарсис**: что же очищает всё это страдание

Или, «почему эти идиоты не могут выпустить фичу за месяц, а не за год»

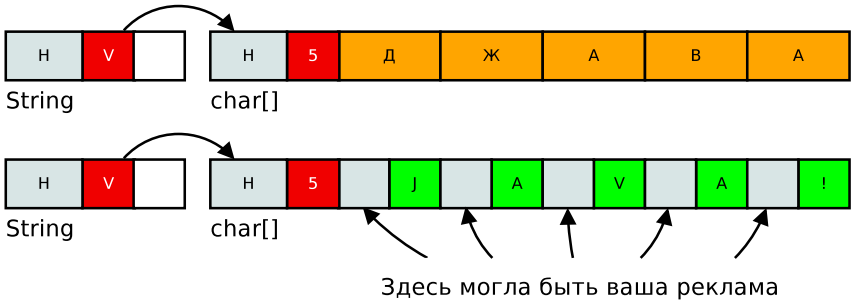
Compact Strings

Compact Strings: как устроен String



- Два объекта: `String` и `char[]`
- Нигде ничего лишнего нет?

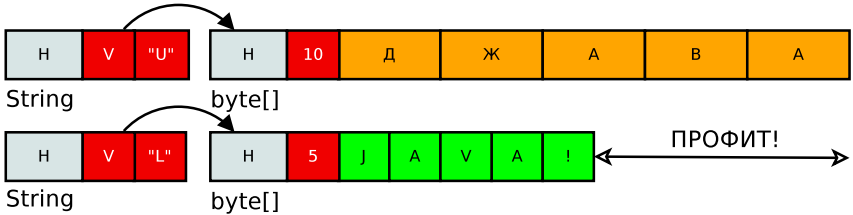
Compact Strings: как устроен String



- Большая часть байтов в `char[]` – нули
- Потому что большая часть стрингов укладывается в Latin1

Compact Strings: было круто, если...

...String был устроен вот так:



- Храним не `char[]`, а плотный `byte[]`
- Сам `String` знает, какой «кодер» использовать для `byte[]`

Compact Strings: о критериях успеха

- Футпринт большей части `String`-ов должен улучшиться
  - При этом можно нафантазировать какие угодно схемы
  - Но надо, чтобы улучшилось *в реальных ситуациях*

Compact Strings: о критериях успеха

- Футпринт большей части `String`-ов должен улучшиться
  - При этом можно нафантазировать какие угодно схемы
  - Но надо, чтобы улучшилось *в реальных ситуациях*
- Нельзя деградировать производительность `String`
  - Можно провалить на маленькую аддитивную константу (если это прямо кровь из носу необходимо)
  - Нельзя ухудшать асимптотическую сложность

Compact Strings: о критериях успеха

- Футпринт большей части `String`-ов должен улучшиться
  - При этом можно нафантазировать какие угодно схемы
  - Но надо, чтобы улучшилось *в реальных ситуациях*
- Нельзя деградировать производительность `String`
  - Можно провалить на маленькую аддитивную константу (если это прямо кровь из носу необходимо)
  - Нельзя ухудшать асимптотическую сложность

Пример: `charAt(int)` должен остаться $O(1)$, даже при случайном доступе.

Compact Strings: о критериях успеха

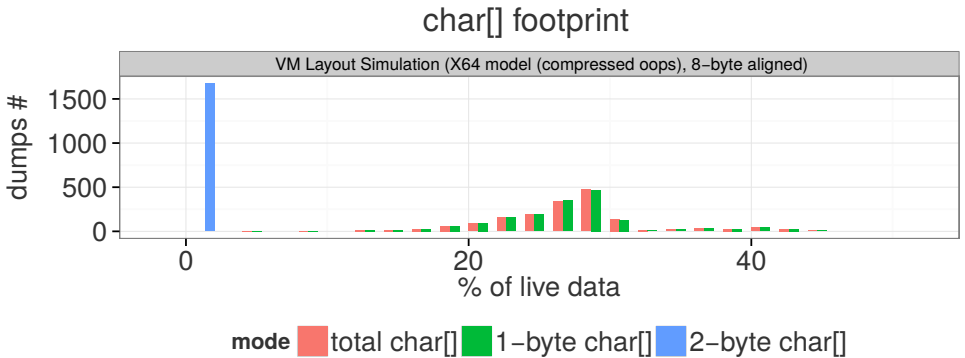
- Футпринт большей части `String`-ов должен улучшиться
  - При этом можно нафантазировать какие угодно схемы
  - Но надо, чтобы улучшилось *в реальных ситуациях*
- Нельзя деградировать производительность `String`
  - Можно провалить на маленькую аддитивную константу (если это прямо кровь из носу необходимо)
  - Нельзя ухудшать асимптотическую сложность

Пример: `charAt(int)` должен остаться $O(1)$, даже при случайном доступе.

Это дисквалифицирует любое кодирование с переменной длиной.
Да, включая UTF-8, увы.

Compact Strings: пререквизиты

В обычных приложениях куча строчек, причём бОльшая часть из них помещается в однобайтовые Latin1:



Compact Strings: В крупную клетку

- Строка сжимается при конструировании
  - Основная беда: конструктор `String(char[] value)`
  - Сразу жмём в 1-байтовое кодирование
  - Не получилось в 1-байт? Копируем как 2-байтовое
  - Иногда можно сэкономить, e.g. `substring` от сжатой строчки уже сжат

- Операции умеют работать как на сжатых, так и разжатых строках
  - Основный выигрыш: не делаем лишней возни
  - Некоторые операции вынуждены разжимать, e.g. `char[] getChars()`

Compact Strings: как разработать фичу в JDK

```
$ <checkout>
$ <configure>
$ sed -i -e "s/char[]/byte[]/g" \
    java/lang/String.java
$ make images
```

1. Запускаем тесты
2. Исправляем баги (пару if-ов здесь и там...)
3. ...
4. PROFIT!

Кто кодирует кодер?

CS, Кто кодирует кодер: ссылка на интерфейс?

Ссылка на реализацию кодера:

- Почти идиоматичный ООП
- Модно, стильно, молодёжно
- Можно накодить этих кодеров, как собак нерезаных

Проблемы?

```
class String {
  byte[] val;
  Coder coder;

  char charAt(int idx) {
    return coder.charAt(val, idx);
  }
}

interface Coder {
  char charAt(byte[] val, int idx);
}
```


Java

CS, Кто кодирует кодер: Object?

Замаскируем `byte[]` или `char[]`:

- «Маленькое» отступление от идиоматики
- Typecheck'и-то дешёвые, небось?

Проблемы?

```
class String {
  Object val;
  char charAt(int idx) {
    // FIXME: Range checks!
    if (val instanceof byte[]) {
      byte[] v = (byte[])val;
      return toChar(v[idx]);
    } else {
      // Totally safe!
      char[] v = (char[])val;
      return v[idx];
    }
  }
}
```


CS, Кто кодирует кодер: замесим в первый байт?

Спрячем в первый байт:

- Элитно, хакерно, классически!
- Можно сравнивать сразу массивы!

Проблемы?

```
class String {
  byte[] val;
  char charAt(int idx) {
    // FIXME: Range checks!
    byte coder = val[0];
    if (coder == LATIN1) {
      return toChar(val[idx + 1]);
    } else {
      return toChar(
         (val[idx*2] << 8)
               + val[idx*2+1]);
    }
  }
}
```


CS, Кто кодирует кодер: покрасим массив?

Сопрём битик из адреса, например:

- VM-щики оргазмируют от таких трюков

Проблемы?

```
class String {
  byte[] val;
  char charAt(int idx) {
    // FIXME: Range checks!
    byte coder = VM_MAGIC(val);
    if (coder == LATIN1) {
      return val[idx];
    } else {
      return getChar(val, idx);
    }
  }
}
```


CS, Кто кодирует кодер: не выпендриваемся

```
***** 32-bit VM: ************************************************
java.lang.String object internals:
 OFFSET  SIZE   TYPE DESCRIPTION                    VALUE
      0     8        (object header)                N/A
      8     4 char[] String.value                   N/A
     12     4    int String.hash                    N/A
Instance size: 16 bytes

***** 64-bit VM, compressed references enabled: *****************
java.lang.String object internals:
 OFFSET  SIZE   TYPE DESCRIPTION                    VALUE
      0    12        (object header)                N/A
     12     4 char[] String.value                   N/A
     16     4    int String.hash                    N/A
     20     4        (loss due to the next object alignment)
Instance size: 24 bytes
```


CS, Кто кодирует кодер: поле, просто поле

Бинарный кодер:

- 0 - Latin1, «все старшие байты – нули»
- 1 - UTF16, «хотя бы в одном символе старший байт – не нуль»

Даёт возможность трюкачить:

- Сравнивать строки без `val` вообще
- Кодер конкатенированной строчки: «OR» кодеров аргументов

```
class String {
  byte[] val;
  byte coder;
  char charAt(int idx) {
    // FIXME: Range checks!
    if (coder == LATIN1) {
      return val[idx];
    } else {
      return getChar(val, idx);
    }
  }
}
```

CS, Кто кодирует кодер: подробнее

Супер-жёсткий Magnum Opus про вызовы методов:
`http://shipilev.net/blog/2015/black-magic-method-dispatch/`

Подробнее про попытки выбрать место для кодера:
`http://cr.openjdk.java.net/~shade/density/double-selection.txt`

Стабильный нуль и прочие извращения

CS, Нуль: what kind of sourcery is this?

Вы когда-нибудь задумывались, почему вот это работает?

```
static final String S;

static {
  S = "Foo";
}

@Benchmark
int testLength() {
  return S.length();
}
```

```
testLength():
  # ... some mumbo-jumbo ...
  # THE ANSWER IS "3"!
  mov $0x3$, %eax
  # ... some mumbo-jumbo ...
  ret
```


Java

CS, Нуль: верю – не верю

```
static final MyClass finallie    = new MyClass();
static       MyClass nonFinallie = new MyClass();

class MyClass {
  final int trustMe = 42;
}

int read_finallie() {
  return finallie.trustMe; // folded to "42", or not?
}

int read_nonFinallie() {
  return nonFinallie.trustMe; // folded to "42", or not?
}
```

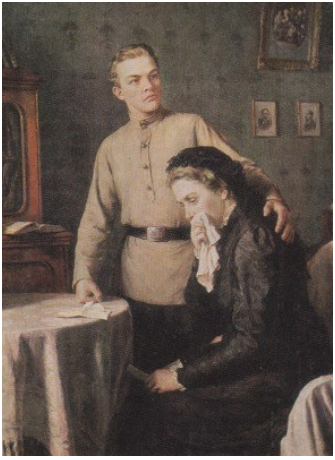

CS, Нуль: верю – не верю

```
static final MyClass finallie = new MyClass();

class MyClass {
  @Stable
  final int trustMe = 42;
}

int read_finallie() {
  return finallie.trustMe; // folded to "42", or not?
}
```


CS, Нуль: верю – не верю

```
class String {
  @Stable final byte coder;
  @Stable final byte[] value;
  int length() { return value.length >> coder);
}

int len_Latin1() {
  return "Foo".length(); // folded or not?
}

int len_UTF16() {
  return "Фуу".length(); // folded or not?
}
```


CS, Нуль: мы пойдём другим путём



CS, Нуль: мы пойдём другим путём

Хакнем компилятор, чтобы все поля в `String` считались константами!

Стоп-кран для фичи:
`-XX:-CompactStrings`

CS, Стоп-кран: прикинемся ветошью

```
class String {
  static final boolean COMPACT_STRINGS = ...;

  String(char[] value) {
     if (COMPACT_STRINGS) {
        byte[] v = tryCompress(value);
        if (v != null) {
           this.value = v; this.coder = LATIN1; return;
        }
     }
     this.value = copy(value); this.coder = UTF16;
  }
}
```


CS, Стоп-кран: работаем...

```
class String {
  static final boolean COMPACT_STRINGS = ...;

  char charAt(int idx) {
    if (coder == LATIN1) { // branch, nevertheless
      return val[idx];
    else
      return getChar(val, idx);
  }
}
```

Всё нормально, к нам будут приходить только с `(coder == UTF16)`?

CS, Стоп-кран: работаем...

```
class String {
  static final boolean COMPACT_STRINGS = ...;

  char charAt(int idx) {
    if ((coder == LATIN1) && COMPACT_STRINGS) {
      return val[idx];
    else
      return getChar(val, idx);
  }
}
```

Всё нормально, свернётся же?

Java

CS, Стоп-кран: приехали

```
class String {
  static final boolean COMPACT_STRINGS = ...;

  char charAt(int idx) {
    if (COMPACT_STRINGS && (coder == LATIN1)) {
      return val[idx];
    else
      return getChar(val, idx);
  }
}
```

Фиг там, только вот так можно делать.[1]

[1] https://bugs.openjdk.java.net/browse/JDK-8087309

CS, Стоп-кран: приехали, №2

```
class String {
  static final boolean COMPACT_STRINGS = ...;
  boolean isLatin1() { return COMPACT_STRINGS && (coder == LATIN1); }

  char charAt(int idx) {
    if (isLatin1()) {
      return val[idx];
    else
      return getChar(val, idx);
  }
}
```

Или так?

CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = Boolean.getBoolean("compactStrings");
}
```


CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = Boolean.getBoolean("compactStrings");
}
```

Циркулярные зависимости:

```
Error occurred during initialization of VM
java.nio.charset.IllegalCharsetNameException: UTF-8
at java.nio.charset.Charset.checkName(Charset.java:316)
    ...
at java.lang.StringCoding.decode(StringCoding.java:334)
at java.lang.String.<init>(String.java:592)
at java.lang.String.<init>(String.java:614)
at java.lang.System.initProperties(Native Method)
at java.lang.System.initializeSystemClass(System.java
```


CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = VMGate.isCompactStrings();
}
```


CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = VMGate.isCompactStrings();
}
```

Упс, а всё плохо:

```
# A fatal error has been detected by the JRE:
#
#  SIGSEGV (0xb) at pc=0x0000000000000000, pid=..., tid=...
#
# JRE version:  (9.0) (build ...)
# Java VM: OpenJDK 64-Bit Server VM ...
# Problematic frame:
# C  0x0000000000000000
```


Java

CS, Стоп-кран: работает!

```
class String {
  s. f. b. COMPACT_STRINGS = isCompactStrings();
  private static native boolean isCompactStrings();
}

JNIEXPORT jboolean JNICALL
Java_java_lang_String_isCompactStrings(JNIEnv *env, jclass cls) {
    return JVM_IsCompactStrings();
}

JNIEXPORT jboolean JNICALL JVM_IsCompactStrings(void) {
  JVMWrapper("JVM_IsCompactStrings");
  return CompactStrings;
}
```


CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = true;
}

void VM::super_secret_VM_method() {
  find_String_field_and_set_it_to(CompactStrings);
}
```


CS, Стоп-кран: работает?

```
class String {
  s. f. b. COMPACT_STRINGS = new Boolean(true).booleanValue();
}

void VM::super_secret_VM_method() {
  find_String_field_and_set_it_to(CompactStrings);
}
```


Дороги, кварталы, Java-овые массивы

CS, массивы: char[]

```
char[] tryWithChar(int size) {
  return new char[size];
}
```

На каком size пофейлит?

CS, массивы: char[]

```
char[] tryWithChar(int size) {
  return new char[size];
}
```

На каком size пофейлит?

```
tryWithChar(Integer.MAX_VALUE);
```

```
Exception in thread "main" java.lang.OutOfMemoryError:
Requested array size exceeds VM limit
at org.openjdk.BoundsTest.testWith(BoundsTest.java:30)
at org.openjdk.BoundsTest.main(BoundsTest.java:25)
```


CS, массивы: char[]

```
char[] tryWithChar(int size) {
  return new char[size];
}
```

На каком size пофейлит?

```
tryWithChar(Integer.MAX_VALUE - 2); // работает
```


CS, массивы: String

```
String tryWithString(int size) {
  return new String(new char[size]);
}
```

```
tryWithString(Integer.MAX_VALUE - 2):
```


CS, массивы: String

```
String tryWithString(int size) {
  return new String(new char[size]);
}
```

```
tryWithString(Integer.MAX_VALUE - 2):
```

```
java.lang.OutOfMemoryError:
UTF16 String size is 2147483645, should be less than 1073741807
at java.lang.StringUTF16.rangeCheck(StringUTF16.java:56)
at java.lang.StringUTF16.compress(StringUTF16.
```


getChar,
или как представить `byte[]` в виде `char[]`

CS, getChar: что уж проще

```
char getChar_UTF16(byte[] val, int idx) {
  idx *= 2;
  return toChar(val[idx]) + toChar(val[idx + 1]) << 8;
}
```


CS, getChar: что уж проще

```
char getChar_UTF16(byte[] val, int idx) {
  idx *= 2;
  return toChar(val[idx]) + toChar(val[idx + 1]) << 8;
}
```

Про эндианность не забыл, чудила?

CS, getChar: что уж проще

```
char getChar_UTF16(byte[] val, int idx) {
  idx *= 2;
  return toChar(val[idx]) + toChar(val[idx + 1]) << 8;
}
```

Про эндианность не забыл, чудила?

```
char getChar_UTF16(byte[] val, int idx) {
  idx *= 2;
  return toChar(val[idx + 1]) + toChar(val[idx]) << 8;
}
```


CS, getChar: побегаем с ножницами

```
// Пацаны, дух старой школы живёт только в:
static final Unsafe U = Unsafe.getUnsafe();
static final int base = U.ARRAY_CHAR_BASE_OFFSET;
static final int scale = U.ARRAY_CHAR_INDEX_SCALE;

char getChar_UTF16(byte[] val, int idx) {
  return U.getChar(val, base + scale*idx);
}
```


CS, getChar: побегаем с ножницами

```
// Пацаны, дух старой школы живёт только в:
static final Unsafe U = Unsafe.getUnsafe();
static final int base = U.ARRAY_CHAR_BASE_OFFSET;
static final int scale = U.ARRAY_CHAR_INDEX_SCALE;

char getChar_UTF16(byte[] val, int idx) {
  return U.getChar(val, base + scale*idx);
}
```

Хинт:

$$\lim_{idx \to MAX} getChar(val, idx) = \kappa\rho\alpha\kappa\circ\zeta\iota\alpha\beta\rho\iota$$

CS, getChar: побегаем с длинными ножницами

```
// Пацаны, дух старой школы живёт только в:
static final Unsafe U = Unsafe.getUnsafe();
static final long base = U.ARRAY_CHAR_BASE_OFFSET;
static final long scale = U.ARRAY_CHAR_INDEX_SCALE;

char getChar_UTF16(byte[] val, int idx) {
  return U.getChar(val, base + scale*idx);
}
```

[2] https://bugs.openjdk.java.net/browse/JDK-8074124

CS, getChar: побегаем с длинными ножницами

```
// Пацаны, дух старой школы живёт только в:
static final Unsafe U = Unsafe.getUnsafe();
static final long base = U.ARRAY_CHAR_BASE_OFFSET;
static final long scale = U.ARRAY_CHAR_INDEX_SCALE;

char getChar_UTF16(byte[] val, int idx) {
  return U.getChar(val, base + scale*idx);
}
```

Только 32-битным платформам трындец[2]:

```
UnsafeConvBench.plain                3835.953 ± 48.240 ns/op
UnsafeConvBench.unsafe_field_scale   7611.268 ± 72.331 ns/op
```

[2] https://bugs.openjdk.java.net/browse/JDK-8074124

Java

CS, getChar: в сухом остатке

Проблемы `(get|put)Char` решаются одной из трёх опций:

1. Автоматическим доказательством отсутствия overflow в выражениях
2. Бурением дырок в `Unsafe`

```
<T> T Unsafe.get_T_Indexed(Object,int)
```

3. Запиливанием интринзиков в C1 и C2:

```
char StringUTF16.getChar(byte[] val, int idx)
void StringUTF16.putChar(byte[] val, int idx, char c)
```


А ещё надо: ...сделать прорву работы

- Повторить всё для других методов `String`
- Повторить всё для других платформ
- Поправить кучу мест в VM
- Написать ещё интринзиков для Бога Интризиков
- Энкодерам/декодерам помочь новые формы `String` использовать
- Переписать `AbstractStringBuilder`'ы
- Поправить оптимизацию конкатенации строк

А ещё надо: ...сделать прорву работы

- Повторить всё для других методов `String`
- Повторить всё для других платформ
- Поправить кучу мест в VM
- Написать ещё интринзиков для Бога Интризиков
- Энкодерам/декодерам помочь новые формы `String` использовать
- Переписать `AbstractStringBuilder`'ы
- Поправить оптимизацию конкатенации строк

CS, Катарсис: LogLineBench

```
public class LogLineBench {
  @Param
  int size;

  String method = generateString(size);

  @Benchmark
  public String work() throws Exception {
      return "[" + System.nanoTime() + "] " +
             Thread.currentThread().getName() + ":" +
             "Calling an application method \"" + method +
             "\" without fear and prejudice.";
  }
}
```


Java

CS, Катарсис: производительность

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
`Baseline`	709 ± 3	721 ± 4	1338 ± 6	888	904	1680
− `Compact Strings`	714 ± 5	726 ± 4	1350 ± 6	888	904	1680
+ `Compact Strings`	403 ± 2	410 ± 3	721 ± 4	504	512	904

- В 1.76x-1.86x лучше пропускная способность, во столько же раз меньше мусора (не удивительно для GC-bound бенчмарка)
- Стоп-кран работает, на случай, если что-то пойдёт не так

Indify String Concat

Indify String Concat: конкатенацию видишь?

```
String m(String a, int b) {  return a + "(" + b + ")"; }
```


Indify String Concat: конкатенацию видишь?

```
String m(String a, int b) {  return a + "(" + b + ")"; }

 0: new           #2   // class java/lang/StringBuilder
 3: dup
 4: invokespecial #3   // Method java/lang/StringBuilder."":()V
 7: aload_1
 8: invokevirtual #4   // Method java/lang/StringBuilder.append:(Ljava/lang
11: ldc           #5   // String (
13: invokevirtual #4   // Method java/lang/StringBuilder.append:(Ljava/lang
16: iload_2
17: invokevirtual #6   // Method java/lang/StringBuilder.append:(I)Ljava/la
20: ldc           #7   // String )
22: invokevirtual #4   // Method java/lang/StringBuilder.append:(Ljava/lang
25: invokevirtual #8   // Method java/lang/StringBuilder.toString:()Ljava/l
28: areturn
```


Indify String Concat: OptimizeStringConcat

Компилятор не тупой, у него есть `-XX:+-OptimizeStringConcat`[3]

1. Посмотреть на IR-граф, который получился из перемалывания исходного байткода
2. Обнаружить там что-то похожее на нужную цепочку `StringBuilder::append`
3. Радостно взвизгнуть и переделать по-своему:
   - Посчитать длины аргументов и финальную длину
   - Где можно, подставить in-place Integer.getChars(...) и т.п.
   - Эффективно скопировать массивы
4. Профит!

[3] `http://hg.openjdk.java.net/jdk9/jdk9/hotspot/file/tip/src/share/vm/opto/stringopts.cpp`

Indify String Concat: Засада №1: «прозрачность»[4]

```
int x;

// Optimizeable:
new StringBuilder().append("Foo: ").append(x).toString();

// NOPE. NOPE. NOPE.
new StringBuilder().append("Foo: ").append(x++).toString();

// Optimizeable again:
x++;
new StringBuilder().append("Foo: ").append(x).toString();
```

[4] https://bugs.openjdk.java.net/browse/JDK-8043677

Indify String Concat: Засада №2: «универсальность»

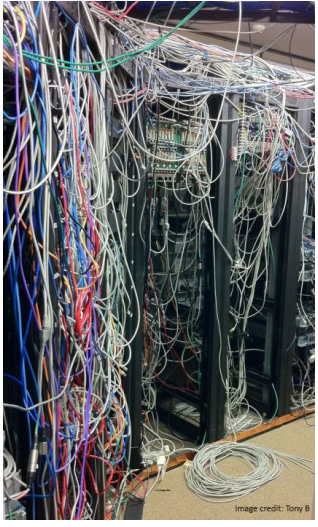
```
int i;
long l;
double d;

// Optimizeable:
new StringBuilder().append("Foo: ").append(i).toString();

// NOPE, LOL
new StringBuilder().append("Foo: ").append(l).toString();

// YOUR WISH, BROTHA, ROFL
new StringBuilder().append("Foo: ").append(d).toString();
```


Indify String Concat: Засада №3: «надёжность»[5]

```
String s1, s2;

// JDK 9 Baseline, optimizeable:
new StringBuilder(s1.length() + s2.length())
    .append(s1).append(s2).toString();

// JDK 9 Compact Strings... NOPE, FU, LOL
class String {
  int length() { return value.length >> coder; }
}

new StringBuilder(s1.length() + s2.length())
    .append(s1).append(s2).toString();
```

[5] https://bugs.openjdk.java.net/browse/JDK-8136469

Indify String Concat: мораль



Хочешь быстрого конката – надо контролировать байткод. А как его контролировать?

Чёрт его знает:

- Заставлять пользователей рекомпилировать с новым `javac`-ом?
- Легко ошибиться в паттерне так, что компилятор не сможет хорошо соптимизировать
- Меняешь паттерн – все остальные паттерны тоже нельзя регрессирова

Indify String Concat: позднее связывание

Выровнять импеданс между языком и байткодом
помогло бы отложенное связывание, да пусть хоть такое:

```
package java.lang;

class StringConcat {
  String concat(String first, String... moar) {
    // TODO: Actually implement this.
  }
}
```


Indify String Concat: позднее связывание

Выровнять импеданс между языком и байткодом
помогло бы отложенное связывание, да пусть хоть такое:

```
package java.lang;

class StringConcat {
  String concat(String first, String... moar) {
    // TODO: Actually implement this.
  }
}
```

А чёрт, объекты... А чёрт, примитивы...

Indify String Concat: позднее связывание

Выровнять импеданс между языком и байткодом
помогло бы отложенное связывание, да пусть хоть такое:

```
package java.lang;

class StringConcat {
  String concat(Object first, Object... moar) {
    // TODO: Actually implement this.
  }
}
```


Indify String Concat: позднее связывание

Выровнять импеданс между языком и байткодом
помогло бы отложенное связывание, да пусть хоть такое:

```
package java.lang;

class StringConcat {
  String concat(Object first, Object... moar) {
    // TODO: Actually implement this.
  }
}
```

А чёрт, боксинг. А чёрт, аллокация массива под vararg...

Java

Indify String Concat: магия invokedynamic

```
String m(String a, int b) {  return a + "(" + b + ")"; }
```


Indify String Concat: магия invokedynamic

```
String m(String a, int b) {  return a + "(" + b + ")"; }

 0: aload_1
 1: ldc           #2      // String (
 3: iload_2
 4: ldc           #3      // String )
 6: invokedynamic #4,  0  // InvokeDynamic #0:stringConcat:
                          //   (String,String,int,String)String;
11: areturn

BootstrapMethods:
   0: #19 invokestatic java/lang/invoke/StringConcatFactory.stringConcat..
```


Indify String Concat: магия invokedynamic, #2

```
String m(String a, int b) {  return a + "(" + b + ")"; }
```


Indify String Concat: магия invokedynamic, #2

```
String m(String a, int b) {  return a + "(" + b + ")"; }
```

```
0: aload_1
1: iload_2
2: invokedynamic #2,  0  // InvokeDynamic #0:stringConcat:
                         //                   (String,int)String;
7: areturn

BootstrapMethods:
  0: #15 invokestatic java/lang/invoke/StringConcatFactory.stringConcat..
                      (Lookup,String,MethodType,String,Object[])CallSite
    Method arguments:
      #16 OCOC
      #17 (
      #18 )
```

Что нам стоит дом построить:
компилируем весь конкат в invokedynamic...

ISC, Инит: бабах

```
Error occurred during initialization of VM
java.lang.StackOverflowError
at java.lang.Throwable.toString(Throwable.java:481)
at java.lang.BootstrapMethodError.<init>(BootstrapMethodError.java:77
at java.lang.Throwable.toString(Throwable.java:481)
...
at sun.nio.cs.StandardCharsets.lookup(StandardCharsets.java:1100)
at sun.nio.cs.StandardCharsets.charsetForName(StandardCharsets.java:1
```


ISC, Инит: бабах

```
Error occurred during initialization of VM
java.lang.StackOverflowError
at java.lang.Throwable.toString(Throwable.java:481)
at java.lang.BootstrapMethodError.<init>(BootstrapMethodError.java:77
at java.lang.Throwable.toString(Throwable.java:481)
...
at sun.nio.cs.StandardCharsets.lookup(StandardCharsets.java:1100)
at sun.nio.cs.StandardCharsets.charsetForName(StandardCharsets.java:1
```

Конкату нужен java.lang.invoke, который использует конкат, которому нужен java.lang.invoke, который использует конкат, которому нужен java.lang.invoke, который использует конкат, которому нужен java.lang.invoke, ...

ISC, Инит: необходимое зло?

До JEP 201 «Modular Source Code»[6]:

1. `java.lang.invoke` свободен использовать всякие JDK-овые классы
2. Всем JDK-овым классам нельзя использовать indy string concat

[6] `http://openjdk.java.net/jeps/201`

ISC, Инит: необходимое зло?

До JEP 201 «Modular Source Code»[6]:

1. `java.lang.invoke` свободен использовать всякие JDK-овые классы
2. Всем JDK-овым классам нельзя использовать indy string concat

После JEP 201:

1. `java.lang.invoke` свободен использовать классы из `java.base`
2. Всем классам из `java.base` нельзя использовать indy string concat
3. Спасибо, дорогой Jigsaw!

[6] `http://openjdk.java.net/jeps/201`

Но ведь изначальная линковка invokedynamic нам что-то стоит? Что там со стартапом?

ISC, Стартап: мерим HelloWorld

JDK 9 Baseline:

```
-XstringConcat:inline:  200.1 ± 1.5 ms
-XstringConcat:indy:    201.2 ± 2.0 ms
```

Всё классно! Никаких разрывов! Дешёвый `java.lang.invoke`!

[7] `https://bugs.openjdk.java.net/browse/JDK-8136854`

ISC, Стартап: мерим HelloWorld

JDK 9 Baseline:

```
-XstringConcat:inline:  200.1 ± 1.5 ms
-XstringConcat:indy:    201.2 ± 2.0 ms
```

Всё классно! Никаких разрывов! Дешёвый `java.lang.invoke`!
Погодите, но раньше же было 30 мс на стартап?

[7] `https://bugs.openjdk.java.net/browse/JDK-8136854`

ISC, Стартап: мерим HelloWorld

JDK 9 Baseline:

```
-XstringConcat:inline:  200.1 ± 1.5 ms
-XstringConcat:indy:    201.2 ± 2.0 ms
```

Всё классно! Никаких разрывов! Дешёвый `java.lang.invoke`!
Погодите, но раньше же было 30 мс на стартап?
Oh shi... баг в G1.[7]

[7] `https://bugs.openjdk.java.net/browse/JDK-8136854`

ISC, Стартап: мерим HelloWorld

JDK 9 Baseline:

```
-XstringConcat:inline:  200.1 ± 1.5 ms
-XstringConcat:indy:    201.2 ± 2.0 ms
```

Всё классно! Никаких разрывов! Дешёвый `java.lang.invoke`!
Погодите, но раньше же было 30 мс на стартап?
Oh shi... баг в G1.[7]

`-XX:+UseParallelGC`:

```
-XstringConcat:inline:  31.1 ± 1.5 ms
-XstringConcat:indy:    58.5 ± 2.0 ms
```

[7] https://bugs.openjdk.java.net/browse/JDK-8136854

ISC, Стартап: необходимое зло?

Первый пользователь invokedynamic платит за всех:[8]

- Indify String Concat
- Фабрики лямбд
- Nashorn
- Модульные класслоадеры из Jigsaw

Кто первый стартует, тот в «регрессии» и виноват!

[8] https://bugs.openjdk.java.net/browse/JDK-8086045

ISC, Катарсис: LogLineBench

```
public class LogLineBench {
  @Param
  int size;

  String method = generateString(size);

  @Benchmark
  public String work() throws Exception {
      return "[" + System.nanoTime() + "] " +
             Thread.currentThread().getName() + ":" +
             "Calling an application method \"" + method +
             "\" without fear and prejudice.";
  }
}
```


ISC, Катарсис: производительность

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
Baseline	709 ± 3	721 ± 4	1338 ± 6	888	904	1680
BC_SB	709 ± 4	716 ± 9	1338 ± 9	888	904	1680
BC_SB_SIZED	447 ± 3	478 ± 9	763 ± 5	560	600	960
MH_SB_SIZED	446 ± 3	472 ± 3	763 ± 5	560	592	960
BC_SB_SIZED_EXACT	268 ± 2	281 ± 1	427 ± 2	336	352	536
MH_INLINE_SIZED_EXACT	229 ± 2	241 ± 2	384 ± 3	288	304	488

- В 3.1x-3.5x лучше пропускная способность, во столько же раз меньше мусора
- BC_SB (чисто перенесённый из javac) вообще не регрессировал

Рандеву

Рандеву: Compact Strings + Indy String Concat

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
Baseline	709 ± 3	721 ± 4	1338 ± 6	888	904	1680
BC_SB	404 ± 4	410 ± 3	721 ± 4	504	512	904
BC_SB_SIZED	248 ± 2	263 ± 3	406 ± 2	312	328	512
BC_SB_SIZED_EXACT	275 ± 2	288 ± 3	428 ± 2	344	360	536
MH_SB_SIZED	249 ± 2	261 ± 1	409 ± 3	312	328	512
MH_INLINE_SIZED_EXACT	188 ± 4	189 ± 2	210 ± 2	168	176	264

- В 3.8x-6.4x лучше пропускная способность и меньше мусора
- Даже тупой BC_SB куда лучше из-за Compact Strings
- Супер-оптимизированные стратегии рвут вообще всех и всё

Рандеву: Compact Strings + Indy String Concat

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
Baseline	709 ± 3	721 ± 4	1338 ± 6	888	904	1680
BC_SB	404 ± 4	410 ± 3	721 ± 4	504	512	904
BC_SB_SIZED	248 ± 2	263 ± 3	406 ± 2	312	328	512
BC_SB_SIZED_EXACT	275 ± 2	288 ± 3	428 ± 2	344	360	536
MH_SB_SIZED	249 ± 2	261 ± 1	409 ± 3	312	328	512
MH_INLINE_SIZED_EXACT	188 ± 4	189 ± 2	210 ± 2	168	176	264

- В 3.8x-6.4x лучше пропускная способность и меньше мусора
- Даже тупой BC_SB куда лучше из-за Compact Strings
- Супер-оптимизированные стратегии рвут вообще всех и всё

Рандеву: Compact Strings + Indy String Concat

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
Baseline	709 ± 3	721 ± 4	1338 ± 6	888	904	1680
BC_SB	404 ± 4	410 ± 3	721 ± 4	504	512	904
BC_SB_SIZED	248 ± 2	263 ± 3	406 ± 2	312	328	512
BC_SB_SIZED_EXACT	275 ± 2	288 ± 3	428 ± 2	344	360	536
MH_SB_SIZED	249 ± 2	261 ± 1	409 ± 3	312	328	512
MH_INLINE_SIZED_EXACT	188 ± 4	189 ± 2	210 ± 2	168	176	264

- В 3.8x-6.4x лучше пропускная способность и меньше мусора
- Даже тупой BC_SB куда лучше из-за Compact Strings
- Супер-оптимизированные стратегии рвут вообще всех и всё

Рандеву: эпистаз

Возьмём `BC_SB_SIZED_EXACT`:

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
CS	403 ± 2	410 ± 3	721 ± 4	504	512	904
ISC	268 ± 2	281 ± 1	427 ± 2	336	352	536
ISC + CS	275 ± 2	288 ± 3	428 ± 2	344	360	536

Рандеву: эпистаз

Возьмём `BC_SB_SIZED_EXACT`:

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
CS	403 ± 2	410 ± 3	721 ± 4	504	512	904
ISC	268 ± 2	281 ± 1	427 ± 2	336	352	536
ISC + CS	275 ± 2	288 ± 3	428 ± 2	344	360	536

- Из-за Compact Strings в конкатенации появились ветвления на кодере

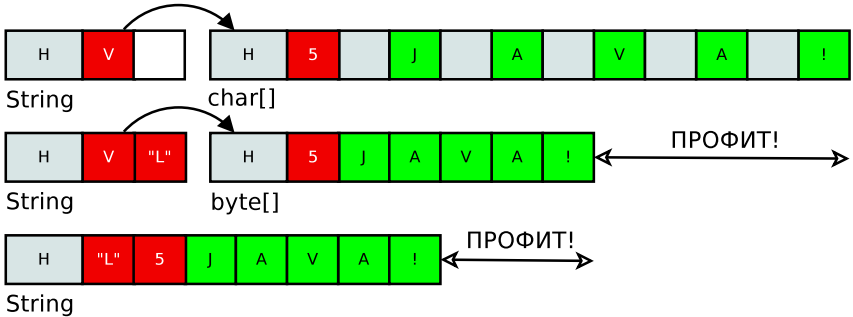
Рандеву: эпистаз

Возьмём `BC_SB_SIZED_EXACT`:

	throughput, ns/op			allocated, b/op		
	1	10	100	1	10	100
`CS`	403 ± 2	410 ± 3	721 ± 4	504	512	904
`ISC`	268 ± 2	281 ± 1	427 ± 2	336	352	536
`ISC + CS`	275 ± 2	288 ± 3	428 ± 2	344	360	536

- Из-за Compact Strings в конкатенации появились ветвления на кодере
- ...но выпавший из C2 код профилировать поздно!

Рандеву: String Fusion



- Осталось только выбросить сам `byte[]` и вклеить его в сам `String`
- String Fusion удобнее делать после Project Panama

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904
ISC (best)	384	488

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904
ISC (best)	384	488
CS + ISC (best)	210	264

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904
ISC (best)	384	488
CS + ISC (best)	210	264
«Garbage Free» boundary		264

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904
ISC (best)	384	488
CS + ISC (best)	210	264
«Garbage Free» boundary		264
CS + ISC (best) + SF	???	???

Рандеву: будущее

Mode	Throughput, ns/op	Allocation, b/op
Current JDK	1338	1680
CS	763	904
ISC (best)	384	488
CS + ISC (best)	210	264
«Garbage Free» boundary		264
CS + ISC (best) + SF	???	???
Absolute minimum	???	235

Заключение

Заключение: проекты

Compact Strings:
http://openjdk.java.net/jeps/254

Indify String Concat:
http://openjdk.java.net/jeps/8085796

Q/A

ОБЩЕСТВО

ФАС проверит, почему длина SMS на кириллице короче, чем на латинице

12:06 05.08.2011 (обновлено: **12:07 05.08.2011**) 21975 5 13

На данный момент длина одного SMS-сообщения ограничена 70 символами на кириллице, в то время как сообщение на латинице может быть более чем в два раза длиннее - 160 символов. Таким образом, потребители вынуждены переплачивать за SMS, набранное кириллическими символами, считает ФАС.

http://ria.ru/society/20110805/412291550.html