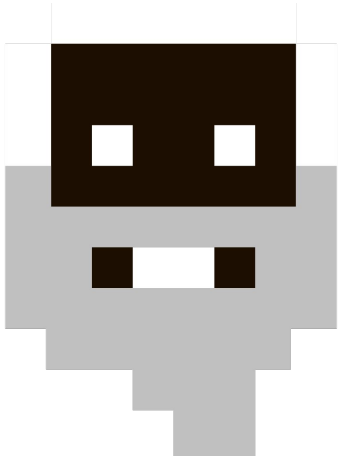
Один день из жизни JVM инженера



Иван, здравствуйте!
Сейчас мы ищем сильных java разработчиков в нашу команду.

Позвольте обратиться к Вам с предложением рассмотреть вакансию Senior Java Developer открытую в нашей компании.

Иван, здравствуйте!
Сейчас мы ищем сильных java разработчиков в нашу команду.

Позвольте обратиться к Вам с предложением рассмотреть вакансию Senior Java Developer открытую в нашей компании.

Помоги, пожалуйста, с написанием тестового задания на Java?

Пройди квиз по Java, получи носки от партнеров!

Иван, здравствуйте!
Сейчас мы ищем сильных java разработчиков в нашу команду.

Позвольте обратиться к Вам с предложением рассмотреть вакансию Senior Java Developer открытую в нашей компании.

Помоги, пожалуйста, с написанием тестового задания на Java?

Пройди квиз по Java, получи носки от партнеров!

Простите, но я не Java разработчик. Я разработчик Java.


Java мир


Java разработчик
(джун)
Java мир


Java мир
Java разработчик
(джун)
JVM


Java мир
Java разработчик
(джун)
JVM
JIT
GC
C++
0xFF
JVM-инженеры

Иван Углянский

JVM engineer at Excelsior

JVM engineer at Excelsior@Huawei

@dbg_nsk

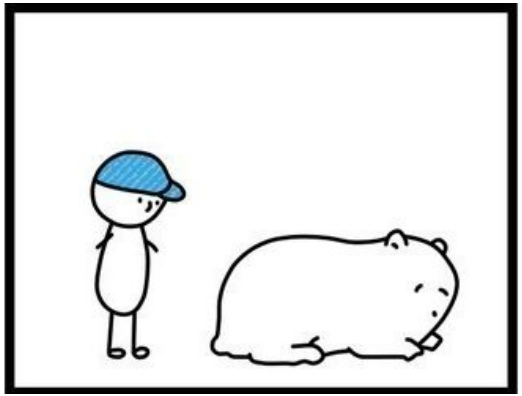
@ugliansky

Что в докладе

1. Кто такие JVM-инженеры, и зачем они нужны?
2. Как именно разрабатываются фичи в JVM?
3. Байки из жизни

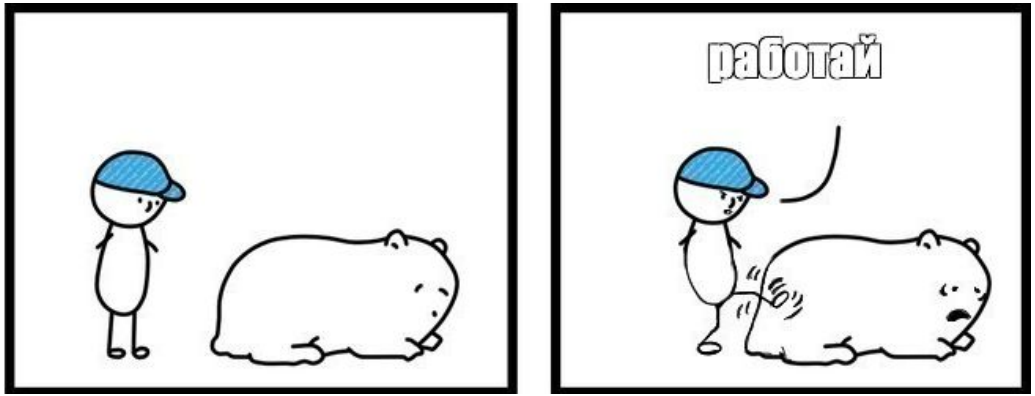
Что в докладе

1. Кто такие JVM-инженеры, и зачем они нужны?
2. Как именно разрабатываются фичи в JVM?
3. Байки из жизни
4. Почему так долго все делаете??



Что в докладе

1. Кто такие JVM-инженеры, и зачем они нужны?
2. Как именно разрабатываются фичи в JVM?
3. Байки из жизни
4. Почему так долго все делаете??



Как работает Java?

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        Object obj = new Object();
        System.out.println(obj.hashCode());
    }
}
```

test.java

Как работает Java?

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        Object obj = new Object();
        System.out.println(obj.hashCode());
    }
}
```

test.java

javac

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

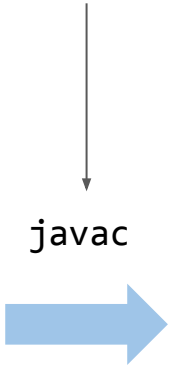
test.class

Как работает Java?

frontend-разработка
(никакого JavaScript-а!)

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        Object obj = new Object();
        System.out.println(obj.hashCode());
    }
}
```

test.java




```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush        10
5:  if_icmpge     32
8:  new           #2
11: dup
12: invokespecial #1
15: astore_2
16: getstatic     #3
19: aload_2
20: invokevirtual #4
23: invokevirtual #5
26: iinc          1, 1
29: goto          2
32: return
```

test.class

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Счастье

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM


```
...
add     rsp, byte -24
mov     eax, dword [rsp-0C00H]
lea     rax, [rel L002]
mov     qword [rsp], rax
lea     rdi, [rsp+8H]
xor     eax, eax
stosq
stosq
xor     ebx, ebx
jmp     short L012 ; v
add     ebx, byte 1
lea     rdi, [rsp+14H]
xor     eax, eax
stosd
lea     rbp, [rel _0java_lang_Object]
mov     qword [rsp+8H], rbp
mov     dword [rsp+10H], -2126479360
lea     rcx, [rsp+8H]
mov     rsi, qword [rel
_4java_lang_System_out]
call    hashCode
mov     edx, eax
mov     rax, qword [rsi]
mov     rcx, rsi
mov     rax, qword [rax+1D0H]
call    rax
...
```

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Интерпретация

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Интерпретация

JIT-компиляция

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Интерпретация

JIT-компиляция

AOT-компиляция

Как работает Java?

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

JVM

Интерпретация

JIT-компиляция

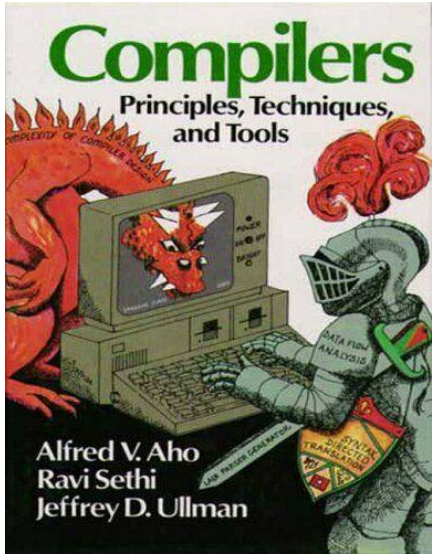
AOT-компиляция

И рантайм!

Дела в компиляторе:

- Глобально - породить машинный код:
  - ✓ под нужную платформу
  - ✓ корректный*
  - ✓ эффективный

Дела в компиляторе:



- Глобально - породить машинный код:
  - ✓ под нужную платформу
  - ✓ корректный*
  - ✓ эффективный

- Более конкретно:
  - ✓ Parsing => Lowering => Code generation

  - ✓ DCE, Inline, Loop unrolling, Global value numbering, Scalar replacement, Register allocation, Bounds-checking optimizations, and more and more

Дела в рантайме:

Дела в рантайме:

- Глобально — помочь полученному машинному коду работать (эффективно)

Дела в рантайме:

- Глобально — помочь полученному машинному коду работать (эффективно)

- Более конкретно:
  - ✓ Аллокация памяти и сборка мусора
  - ✓ Поддержка `tiered compilation`: от интерпретатора к более оптимизированному коду (и обратно!)

Дела в рантайме:

- Глобально — помочь полученному машинному коду работать (эффективно)

- Более конкретно:
  - ✓ Аллокация памяти и сборка мусора
  - ✓ Поддержка `tiered compilation`: от интерпретатора к более оптимизированному коду (и обратно!)
  - ✓ Реализация высокоуровневых фич языков: исключения, синхронизация, `threading`, рефлексия, интероп и т.д.

Пара экзотических примеров!

Неявные исключения

```
struct st {
    int a, b;
};

int main() {
    struct st * ptr = NULL;
    ptr->a = 42;
}
```

Неявные исключения

```
struct st {
    int a, b;
};

int main() {
    struct st * ptr = NULL;
    ptr->a = 42;
}
```


```
Segmentation fault
(core dumped)
```

точнее UB

Неявные исключения

```
class Test {
    int a;
}

public static void main(String[] args) {
    Test t = null;
    t.a = 42;
}
```

```
Exception in thread "main" java.lang.NullPointerException:
Cannot assign field "a" because "t" is null
        at Main.main(Main.java:9)
```

Неявные исключения

```
class Test {
    int a;
}

public static void main(String[] args) {
    Test t = null;
    t.a = 42;
}
```

```
Exception in thread "main" java.lang.NullPointerException:
Cannot assign field "a" because "t" is null
        at Main.main(Main.java:9)
```

Как добиться такого поведения?

Неявные исключения

```
public static void foo(Test t) {

        t.a = 42;



}

foo(null);
```

Неявные исключения

```
public static void foo(Test t) {
    if (t != null) { // [COMPILER-GENERATED]
        t.a = 42;
    } else {         // [COMPILER-GENERATED]
        throw new NullPointerException();
    }
}

foo(null);
```

Неявные исключения

```
public static void foo(Test t) {
    if (t != null) { // [COMPILER-GENERATED]
        t.a = 42;
    } else {         // [COMPILER-GENERATED]
        throw new NullPointerException();
    }
}

foo(null);
```

Каждый доступ к полю, каждый вызов функции обрамлять проверкой?

Но это же очень дорого!

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

Каждый доступ к полю, каждый вызов функции обрамлять проверкой?

Но это же очень дорого!

А как на самом деле?

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

Каждый доступ к полю, каждый вызов функции обрамлять проверкой?

Но это же очень дорого!

А как на самом деле?

```
-XX:+UnlockDiagnosticVMOptions
-XX:+PrintAssembly
```

```
mov DWORD PTR [rsi+0xc], 0x2a  ;*putfield a {reexecute=0 rethrow=0 return_oop=0}
                               ; - Main::foo@34 (line 13)
```

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

Каждый доступ к полю, каждый вызов функции обрамлять проверкой?

Но это же очень дорого!

А как на самом деле?

```
-XX:+UnlockDiagnosticVMOptions
-XX:+PrintAssembly
```

```
mov DWORD PTR [rsi+0xc], 0x2a  ;*putfield a {reexecute=0 rethrow=0 return_oop=0}
                               ; - Main::foo@34 (line 13)
                               ; implicit exception: dispatches to 0x0000028cc
```

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

а пусть
взрывается!

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

а пусть
взрывается!

SEGFAULT

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

а пусть
взрывается!

SEGFAULT

Segfault
перехватывается
рантаймом!

Неявные исключения

```
public static void foo(Test t) {
    t.a = 42;
}

foo(null);
```

а пусть взрывается!

SEGFAULT

Segfault перехватывается рантаймом!

В этой точке может быть NPE?

да

```
throw new NullPointerException()
```

нет

core dump

Неявные исключения

- Бесплатные, когда исключение не случается

Неявные исключения

- Бесплатные, когда исключение не случается
- Дорогие, когда случаются, но это редко! (а если будет часто случаться - деоптимизируем)

Неявные исключения

- Бесплатные, когда исключение не случается
- Дорогие, когда случаются, но это редко! (а если будет часто случаться - деоптимизируем)
- Чтобы сработали, рантайм фильтрует сегфолты (когда весь мир рушится!), полезные превращает в исключения
- Похожим образом могут обрабатываться деление на ноль, SOE, `safe-points` и даже `clinit checks`


говорят, new то
ненастоящий!

new то ненастоящий!

- Выделять объекты в памяти ⇒ нагружать GC
- Поэтому JVM максимально старается этого избежать

new то ненастоящий!

- Выделять объекты в памяти ⇒ нагружать GC
- Поэтому JVM максимально старается этого избежать
- Пользуются ли объектом только локально, или же он утекает в кучу? На это ответит анализ утеканий или `Escape Analysis`

new то ненастоящий!

```
class Test {
   int a, b;
}


public void foo() {
   Test obj = new Test();
   obj.a = 42;
   obj.b = 10;

   System.out.println(obj.a + obj.b);
}
```

new то ненастоящий!

```
class Test {
   int a, b;
}


public void foo() {
   Test obj = new Test();
   obj.a = 42;
   obj.b = 10;

   System.out.println(obj.a + obj.b);
}
```

obj существует только в этой области. Ну зачем его аллоцировать в хипе?

new то ненастоящий!

```
class Test {
   int a, b;
}


public void foo() {
   Test obj = new Test();
   obj.a = 42;
   obj.b = 10;

   System.out.println(obj.a + obj.b);
}
```

В таком случае можно применить оптимизацию Scalar Replacement

new то ненастоящий!

```
class Test {
   int a, b;
}


public void foo() {
   // REMOVED BY COMPILER
   // Test obj = new Test();

   int a = 42;
   int b = 10;

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

В таком случае можно применить оптимизацию Scalar Replacement

1. Объект распадается на атомы (работаем с его полями, как с обычными локалами)
2. Заголовка нет, аллокации тем более 👍

new то ненастоящий!

```
class Test {
   int a, b;
}


public void bar(Test arg) {
   arg.b = 10;
   System.out.println(arg.a + arg.b);
}

public void foo() {
   Test obj = new Test();
   obj.a = 42;
   bar(obj);
}
```

new то ненастоящий!

```
class Test {
   int a, b;
}


public void bar(Test arg) {
   arg.b = 10;
   System.out.println(arg.a + arg.b);
}

public void foo() {
   Test obj = new Test();
   obj.a = 42;
   bar(obj);
}
```

Допустим bar не проинлайнился, тогда так просто не взорвешь!

Становится дороже передавать внутренности объекта в функции, чем просто ссылку на этот объект.

new то ненастоящий!

```
class Test {
   int a, b;
}


public void bar(Test arg) {
   arg.b = 10;
   System.out.println(arg.a + arg.b);
}

public void foo() {
   Test obj = new Test();
   obj.a = 42;
   bar(obj);
}
```

Допустим bar не проинлайнился, тогда так просто не взорвешь!

Становится дороже передавать внутренности объекта в функции, чем просто ссылку на этот объект.

Но при этом new все еще лишний, утеканий ведь нет!

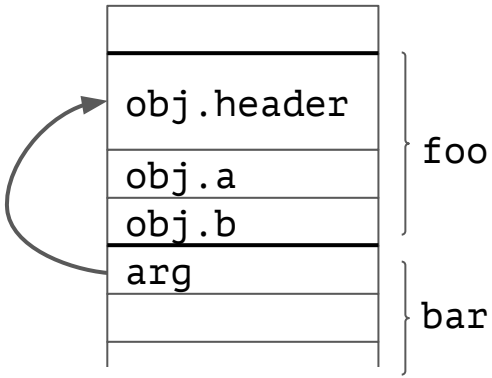
new то ненастоящий!

```
class Test {
   int a, b;
}


public void bar(Test arg) {
   arg.b = 10;
   System.out.println(arg.a + arg.b);
}

public void foo() {
   Test obj = new [on stack] Test();
   obj.a = 42;
   bar(obj);
}
```

Тогда можно заменить new на аллокацию на стеке.



```
class Test {
    int a, b;
}

static Test t;

public void foo(boolean shouldEscape) {
    Test obj = new Test();
    obj.a = 42;
    obj.b = 10;

    System.out.println(obj.a + obj.b);

    if (shouldEscape) {
        t = obj;
    }
}
```

```
class Test {
    int a, b;
}

static Test t;

public void foo(boolean shouldEscape) {
    Test obj = new Test();
    obj.a = 42;
    obj.b = 10;

    System.out.println(obj.a + obj.b);

    if (shouldEscape) {
        t = obj;
    }
}
```

С точки зрения любого анализа escape здесь случается (пусть и не всегда)

```
class Test {
    int a, b;
}

static Test t;

public void foo(boolean shouldEscape) {
    Test obj = new Test();
    obj.a = 42;
    obj.b = 10;

    System.out.println(obj.a + obj.b);

    if (shouldEscape) {
        t = obj;
    }
}
```

С точки зрения любого анализа escape здесь случается (пусть и не всегда)

Но есть более мощный анализ: partial escape analysis, который позволяет сделать так

```
class Test {
    int a, b;
}

static Test t;

public void foo(boolean shouldEscape) {
    int a = 42;
    int b = 10;

    System.out.println(a + b);

    if (shouldEscape) {
        Test obj = new Test();
        obj.a = a;
        obj.b = b;
        t = obj;
    }
}
```

С точки зрения любого анализа escape здесь случается (пусть и не всегда)

Но есть более мощный анализ: partial escape analysis, который позволяет сделать так

```
class Test {
    int a, b;
}

static Test t;

public void foo(boolean shouldEscape) {
    int a = 42;
    int b = 10;

    System.out.println(a + b);

    if (shouldEscape) {
        Test obj = new Test();
        obj.a = a;
        obj.b = b;
        t = obj;
    }
}
```

С точки зрения любого анализа escape здесь случается (пусть и не всегда)

Но есть более мощный анализ: partial escape analysis, который позволяет сделать так

Эвакуация объекта в heap только там, где это правда нужно. Зачастую это холодный путь.

Реализовано в компиляторе Graal

new то ненастоящий!

- Многие new - тоже бесплатные, можно не боятся их ставить в своем коде, JVM постарается их убрать
- Scalar replacement - в HotSpot (и везде), Partial EA в Graal компиляторе
- Stack allocation - в IBM OpenJ9, в HotSpot есть только прототип и proposal:

  github.com/microsoft/openjdk-proposals/blob/main/stack_allocation/Stack_Allocation_JEP.md


Оптимизации производительности и потребления памяти
КОМПИЛЯТОР
РАНТАЙМ

Много примеров разных небольших задач в JVM

https://shipilev.net/jvm/anatomy-quarks

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

C++?

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

C++? Но почему именно на нем?

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

C++? Но почему именно на нем?

Производительность и связь с низким уровнем

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

C++? Но почему именно на нем?

Производительность и связь с низким уровнем
Но все не так однозначно!

На чем писать компилятор?

Задача компилятора - породить машинный код:

- ✓ под нужную платформу
- ✓ корректный
- ✓ эффективный

На чем писать компилятор?

Задача компилятора - породить машинный код:

- ✓ под нужную платформу
- ✓ корректный
- ✓ эффективный

Берем байты на вход (исходную программу),
генерировать байты на выходе (машинный код)!

Нет никакой связи с "низким уровнем"!

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

C++? Но почему именно на нем?

Производительность и ~~связь с низким уровнем~~
Но все не так однозначно!

На чем писать компилятор?

Задача компилятора - породить машинный код:

- ✓ под нужную платформу
- ✓ корректный
- ✓ эффективный

Насколько быстро это нужно сделать?

На чем писать компилятор?

Задача компилятора - породить машинный код:

- ✓ под нужную платформу
- ✓ корректный
- ✓ эффективный

Насколько быстро это нужно сделать?

1. Для AOT компиляторов - не так важно

На чем писать компилятор?

Задача компилятора - породить машинный код:

- ✓ под нужную платформу
- ✓ корректный
- ✓ эффективный

Насколько быстро это нужно сделать?

1. Для AOT компиляторов - не так важно
2. Для JIT компиляторов важнее, платим за производительность компилятора стартапом приложения, но...

На чем писать компилятор?

Компилятор	Режим работы	На чем написан

Как работает Java?

frontend-разработка
(никакого JavaScript-а!)

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        Object obj = new Object();
        System.out.println(obj.hashCode());
    }
}
```

test.java

javac

```
0:  iconst_0
1:  istore_1
2:  iload_1
3:  bipush          10
5:  if_icmpge       32
8:  new             #2
11: dup
12: invokespecial   #1
15: astore_2
16: getstatic       #3
19: aload_2
20: invokevirtual   #4
23: invokevirtual   #5
26: iinc            1, 1
29: goto            2
32: return
```

test.class

Получившийся байт-код должен быть:

1. Корректным
2. Универсальным

На чем писать компилятор?

Компилятор	Режим работы	На чем написан
`javac`	`AOT*`	`Java`

На чем писать компилятор?

Компилятор	Режим работы	На чем написан
javac	AOT*	Java
HotSpot C1/C2	JIT	C++
IBM OpenJ9 compiler	JIT & AOT	C++

На чем писать компилятор?

Компилятор	Режим работы	На чем написан
javac	AOT*	Java
HotSpot C1/C2	JIT	C++
IBM OpenJ9 compiler	JIT & AOT	C++
Graal	JIT & AOT	Java (!)

На чем писать компилятор?

Компилятор	Режим работы	На чем написан
javac	AOT*	Java
HotSpot C1/C2	JIT	C++
IBM OpenJ9 compiler	JIT & AOT	C++
Graal	JIT & AOT	Java (!)
Excelsior JET compiler	AOT/JIT	Scala/Java (!!)

На чем писать рантайм?

На чем писать рантайм?

Производительность и связь с низким уровнем

Вот в рантайме и то и другое нужно, как воздух

На чем писать рантайм?

Производительность и связь с низким уровнем

Вот в рантайме и то и другое нужно, как воздух

А еще: нужна предсказуемость/непрерывность исполнения и полный контроль над ним

На чем писать рантайм?

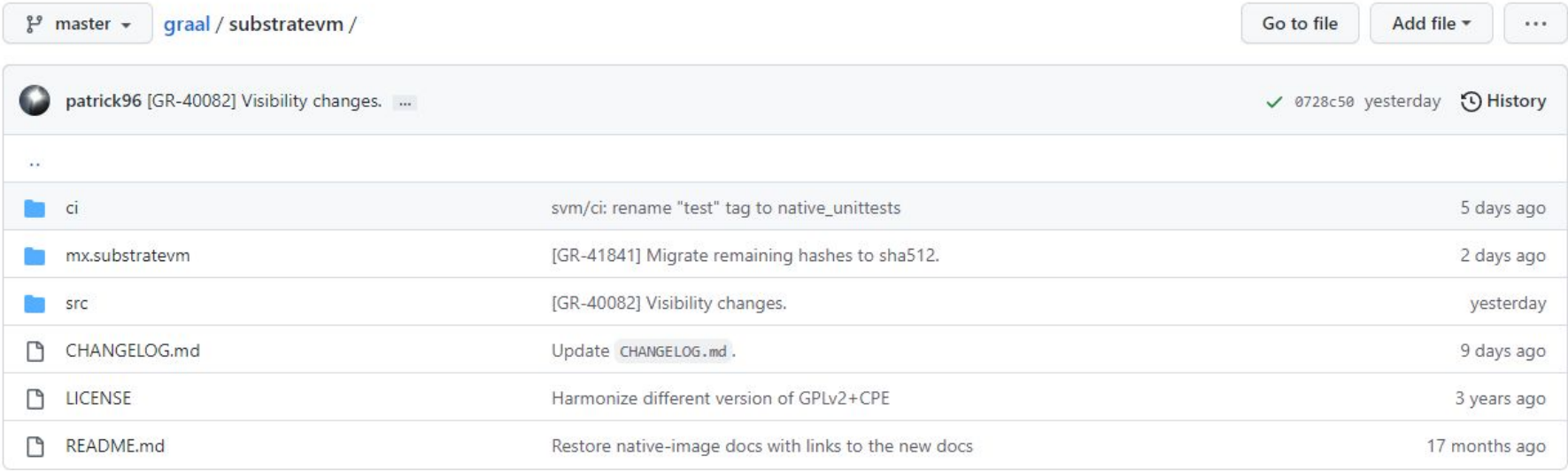
Производительность и связь с низким уровнем

Вот в рантайме и то и другое нужно, как воздух

А еще: нужна предсказуемость/непрерывность исполнения и полный контроль над ним

Значит все-таки C++?

На чем писать рантайм?



На чем писать рантайм?

Repository Structure

This source repository is the main repository for GraalVM and includes the following components:

Directory	Description
`.devcontainer/`	Configuration files for GitHub dev containers.
`.github/`	Configuration files for GitHub issues, workflows, ...
`compiler/`	Graal compiler, a modern, versatile compiler written in Java.
`espresso/`	Espresso, a meta-circular Java bytecode interpreter for the GraalVM.
`java-benchmarks/`	Java benchmarks.
`regex/`	TRegex, a regular expression engine for other GraalVM languages.
`sdk/`	GraalVM SDK, long-term supported APIs of GraalVM.
`substratevm/`	Framework for ahead-of-time (AOT) compilation with Native Image.

На самом деле - полноценный рантайм для Native Image

На чем написан?

На чем писать рантайм?

master ▾ graal / substratevm / src / com.oracle.svm.core / src / com / oracle / svm / core / heap / Go to file Add file ▾ ···

jovanstevanovic JFR StackTrace and JFR Method repositories. Dump samples in form of J... ··· ✕ 417734e on Sep 27 History

..

AbstractMemoryMXBean.java	Update import statements	3 months ago
AllocationFeature.java	Use @AutomaticallyRegisteredFeature and @AutomaticallyRegisteredImage...	2 months ago
BarrierSetProvider.java	Minor refactorings.	3 months ago
ClassHistogramVisitor.java	Replace Heap.getInitializedClasses with Heap.getLoadedClasses.	17 months ago
CodeReferenceMapDecoder.java	JFR StackTrace and JFR Method repositories. Dump samples in form of J...	2 months ago
CodeReferenceMapEncoder.java	Loom support for Native Image.	2 years ago
ExcludeFromReferenceMap.java	Move annotations to different package	3 months ago
FillerObject.java	Use filler objects in aligned chunks to prevent gaps in a range cover...	2 years ago
GC.java	Add AggressiveShrinkCollectionPolicy for libgraal.	4 months ago
GCCause.java	Use @AutomaticallyRegisteredFeature and @AutomaticallyRegisteredImage...	2 months ago
Heap.java	Merge remote-tracking branch 'origin/master' into cwi/GR-38909-substi...	2 months ago

На чем писать рантайм?

```
2 usages
@Uninterruptible(reason = "Avoid races with other threads that also try to trigger a GC")
@RestrictHeapAccess(access = RestrictHeapAccess.Access.NO_ALLOCATION,
        reason = "Must not allocate in the implementation of garbage collection.")
boolean collectWithoutAllocating(GCCause cause, boolean forceFullGC) {
    VMError.guarantee(!hasNeverCollectPolicy());

    int size = SizeOf.get(CollectionVMOperationData.class);
    CollectionVMOperationData data = StackValue.get(size);
    UnmanagedMemoryUtil.fill((Pointer) data, WordFactory.unsigned(size), (byte) 0);
    data.setNativeVMOperation(collectOperation);
    data.setCauseId(cause.getId());
    data.setRequestingEpoch(getCollectionEpoch());
    data.setRequestingNanoTime(System.nanoTime());
    data.setForceFullGC(forceFullGC);
    enqueueCollectOperation(data);
    return data.getOutOfMemory();
}
```

На чем писать рантайм?

The Substrate VM is ...

... an **embeddable** VM

for, and written in, a **subset of Java**

optimized to **execute Truffle** languages

ahead-of-time compiled using Graal

integrating with **native development tools.**

https://www.oracle.com/technetwork/java/jvmls2016-wimmer-3125555.pdf

На чем писать рантайм?

На чистой Java (или любом другом обычном managed языке) писать рантаймы не получится, но…

…можно делать свои специализированные полу-managed языки, которые вам подойдут

Пример: SubstrateVM/SystemJava

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

Некоторые современные JVM разрабатываются не на C++, а на managed языках (Java, Scala, их вариации для рантайма)

Как разрабатывать JVM?

На чем писать рантаймы и компиляторы?

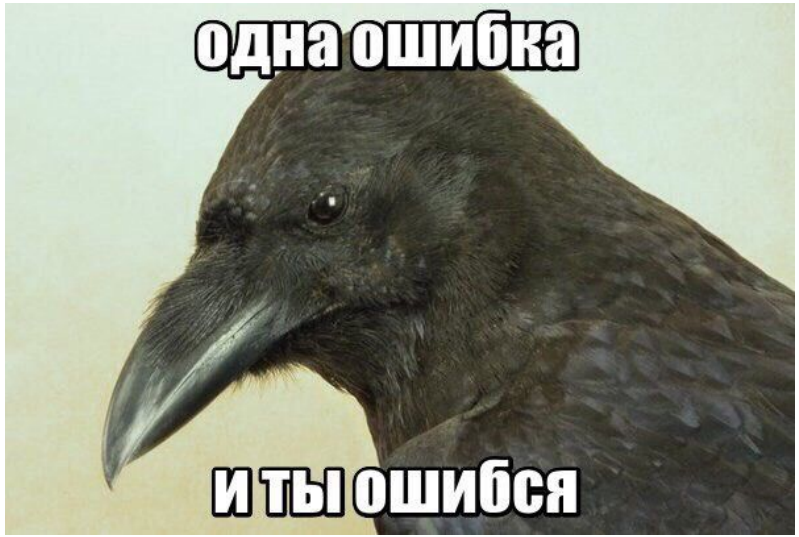
Некоторые современные JVM разрабатываются не на C++, а на managed языках (Java, Scala, их вариации для рантайма)

Это значит: разработка в IDEA, юнит-тесты и более низкий порог входа в разработку

Отлаживай так, словно никто не видит

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC



Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Проявления:

- Собираются живые объекты ⇒

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Проявления:

- Собираются живые объекты ⇒

  ✓ развалы JVM
  (не должно происходить никогда!)

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Проявления:

- Собираются живые объекты ⇒

  ✓ развалы JVM
  (не должно происходить никогда!)

  ✓ некорректное поведение!

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Проявления:

- Собираются живые объекты ⇒
  - ✓ развалы JVM
    (не должно происходить никогда!)
  - ✓ некорректное поведение!
- Не собираются мертвые объекты ⇒
  - ✓ утечки памяти (формально не баг)

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

Да, но...

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

Да, но...

- Проблемы проявляются раз в N запусков (где N может быть ~10000)

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

Да, но...

- Проблемы проявляются раз в N запусков (где N может быть ~10000)
- В отладчике не проявляется (тайминги не те)

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

Да, но...

- Проблемы проявляются раз в N запусков (где N может быть ~10000)
- В отладчике не проявляется (тайминги не те)
- Логи не помогут - диска не хватит

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

А как отлаживать то тогда???

Отлаживай так, словно никто не видит

Допустим, мы ошиблись при разработке GC

Ну, подумаешь! Отладим.

А как отлаживать то тогда???

- Ассерты по всему коду рантайма
- Ловушки и терпеливое ожидание
- Формальная верификация!

Эффект бабочки

Эффект бабочки

Случай из жизни:

1. У некоторых пользователей проявляется странный развал в недрах `msvcr100.dll`

Эффект бабочки

Случай из жизни:

1. У некоторых пользователей проявляется странный развал в недрах `msvcr100.dll`

2. На нашей стороне, конечно, не воспроизводится

Эффект бабочки

Случай из жизни:

1. У некоторых пользователей проявляется странный развал в недрах `msvcr100.dll`
2. На нашей стороне, конечно, не воспроизводится
3. Разваливается не только наша JVM!
4. У всех есть одно общее: `Windows 10 Creators Update`, а кроме того...

Эффект бабочки

Эффект бабочки

Разгадка:

1. У всех пользователей на рабочем столе была "Папка Бога"

Эффект бабочки

Разгадка:

1. У всех пользователей на рабочем столе была "Папка Бога"

2. В `Windows 10 Creators Update` изменилось поведение системного вызова `IShellFolder::GetDisplayNameOf(...)`

Эффект бабочки

Разгадка:

1. У всех пользователей на рабочем столе была "Папка Бога"
2. В `Windows 10 Creators Update` изменилось поведение системного вызова `IShellFolder::GetDisplayNameOf(...)`
3. JVM использовала нативные методы на C, которые этот сискол дергают, не проверяя результат
4. Божественная папка вызывала случайные развалы

Выводы про отладку

- Баги в JVM дороги и коварны
- Отладка — мощный детективный challenge
- На поведение сильно влияют O/S и железо (`memtest` в помощь)

Выводы про отладку

- Баги в JVM дороги и коварны
- Отладка — мощный детективный challenge
- На поведение сильно влияют O/S и железо (`memtest` в помощь)

- Для системного программирования нужны языки, гарантирующие безопасность
- Парадоксальным образом managed языки иногда подходят лучше, чем C++

Клиенты - это Java программисты

Клиенты - это Java программисты

В целом - это очень круто!

Клиенты - это Java программисты

В целом - это очень круто!

Пока клиенты не начинают использовать `sun.misc.Unsafe`

История из жизни

История из жизни

```
ERROR - com.esotericsoftware.kryo.KryoException:
Unknown offset Serialization trace:
        ...
        at com.esotericsoftware.kryo.serializers.UnsafeCacheFields.getField
        at com.esotericsoftware.kryo.serializers.ObjectField.write
        at com.esotericsoftware.kryo.serializers.FieldSerializer.write
        at com.esotericsoftware.kryo.Kryo.writeClassAndObject
```

История из жизни

```
ERROR - com.esotericsoftware.kryo.KryoException:
Unknown offset Serialization trace:
        ...
        at com.esotericsoftware.kryo.serializers.UnsafeCacheFields.getField
        at com.esotericsoftware.kryo.serializers.ObjectField.write
        at com.esotericsoftware.kryo.serializers.FieldSerializer.write
        at com.esotericsoftware.kryo.Kryo.writeClassAndObject
```

Проявляется (иногда) у пользователей Kryo

На Hotspot не проявляется 😔

История из жизни

```
offset = unsafe().objectFieldOffset(f);

...

public Object getField(Object object)
            throws IllegalArgumentException, IllegalAccessException {

    if (offset >= 0) {
        return unsafe().getObject(object, offset);
    } else
        throw new KryoException("Unknown offset");

}
```

История из жизни

звучит логично!

```
offset = unsafe().objectFieldOffset(f);

...

public Object getField(Object object)
            throws IllegalArgumentException, IllegalAccessException {

    if (offset >= 0) {
        return unsafe().getObject(object, offset);
    } else
        throw new KryoException("Unknown offset");

}
```

```java
/**
* Reports the location of a given field in the storage
* allocation of its class.  Do not expect to perform any
* sort of arithmetic on this offset; it is just a cookie
* which is passed to the unsafe heap memory accessors.
* …
*
*/
@ForceInline
public long objectFieldOffset(Field f) { ... }
```

sun.misc.Unsafe.java

```
/**
* Reports the location of a given field in the storage
* allocation of its class.  Do not expect to perform any
* sort of arithmetic on this offset; it is just a cookie
* which is passed to the unsafe heap memory accessors.
* ...
*
*/
@ForceInline
public long objectFieldOffset(Field f) { ... }
```

sun.misc.Unsafe.java

История из жизни

Почему наши оффсеты были отрицательными?

История из жизни

Почему наши оффсеты были отрицательными?


```
(long) offset == 12
```

История из жизни

Почему наши оффсеты были отрицательными?


```
(long) offset == 12
```

Много свободного места в старших битах!

История из жизни

Почему наши оффсеты были отрицательными?


```
(long) offset == 12
```

Много свободного места в старших битах!

Будем использовать в своих целях (например, чтобы быстро распознать интерфейсы)

История из жизни

Почему наши оффсеты были отрицательными?


```
(long) offset ==
-6917529027641081844
```

Много свободного места в старших битах!

Будем использовать в своих целях (например, чтобы быстро распознать интерфейсы)

История из жизни

```
public Object getField(Object object)
            throws IllegalArgumentException, IllegalAccessException {

    if (offset >= 0) {
        return unsafe().getObject(object, offset);
    } else
        throw new KryoException("Unknown offset");

}
```

эта проверка проваливалась для офсетов интерфейсных полей!

История из жизни

```
public Object getField(Object object)
            throws IllegalArgumentException, IllegalAccessException {

    if (offset != Unsafe.INVALID_FIELD_OFFSET) {
        return unsafe().getObject(object, offset);
    } else
        throw new KryoException("Unknown offset");

}
```

История из жизни

```
public Object getField(Object object)
            throws IllegalArgumentException, IllegalAccessException {

    if (offset != Unsafe.INVALID_FIELD_OFFSET) {
        return unsafe().get
    } else
        throw new KryoExcep

}
```

Практические выводы

- Приватные API лучше не использовать!
- Если очень нужно, то быть готовым к развалам, изменению логики и плохой работе с кастомными JVM
- Проверяйте в сорцах своей JVM

А почему все так долго делают?

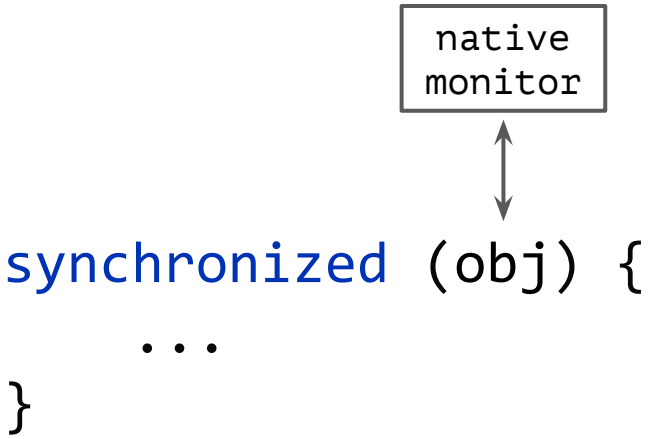
А почему все так долго делают?

- На любом Java объекте можно синхронизироваться

```
synchronized (obj) {
    ...
}
```

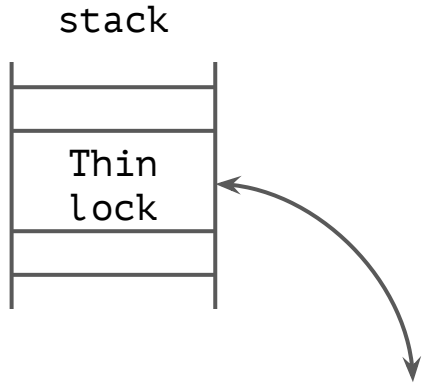
А почему все так долго делают?

- На любом Java объекте можно синхронизироваться
- Базовая реализация через нативные мониторы



А почему все так долго делают?

- На любом Java объекте можно синхронизироваться
- Базовая реализация через нативные мониторы
- Идея для оптимизации #1: обычно конкуренции за монитор нет

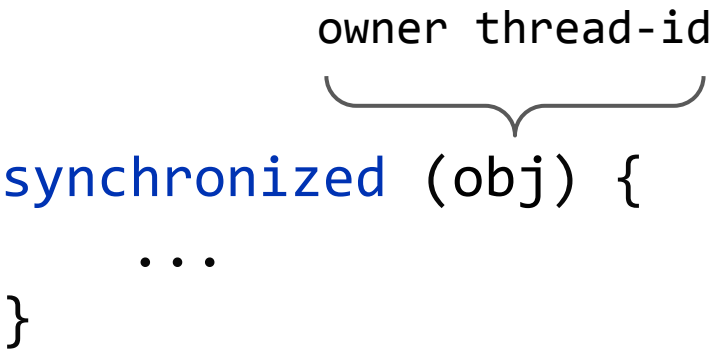



```
synchronized (obj) {
    ...
}
```

CAS-ы вместо нативных мониторов

А почему все так долго делают?

- На любом Java объекте можно синхронизироваться
- Базовая реализация через нативные мониторы
- Идея для оптимизации #1: обычно конкуренции за монитор нет
- Идея для оптимизации #2: зачастую вообще только один тред входит в `synchronized block`



Один CAS на первом входе, дальше просто сравнение

А почему все так долго делают?

- Оптимизация называется `Biased Locking`
- Реализована во многих JVM, включая HotSpot до Java 19
- Звучит так, что все круто и просто!
- Но есть нюанс..

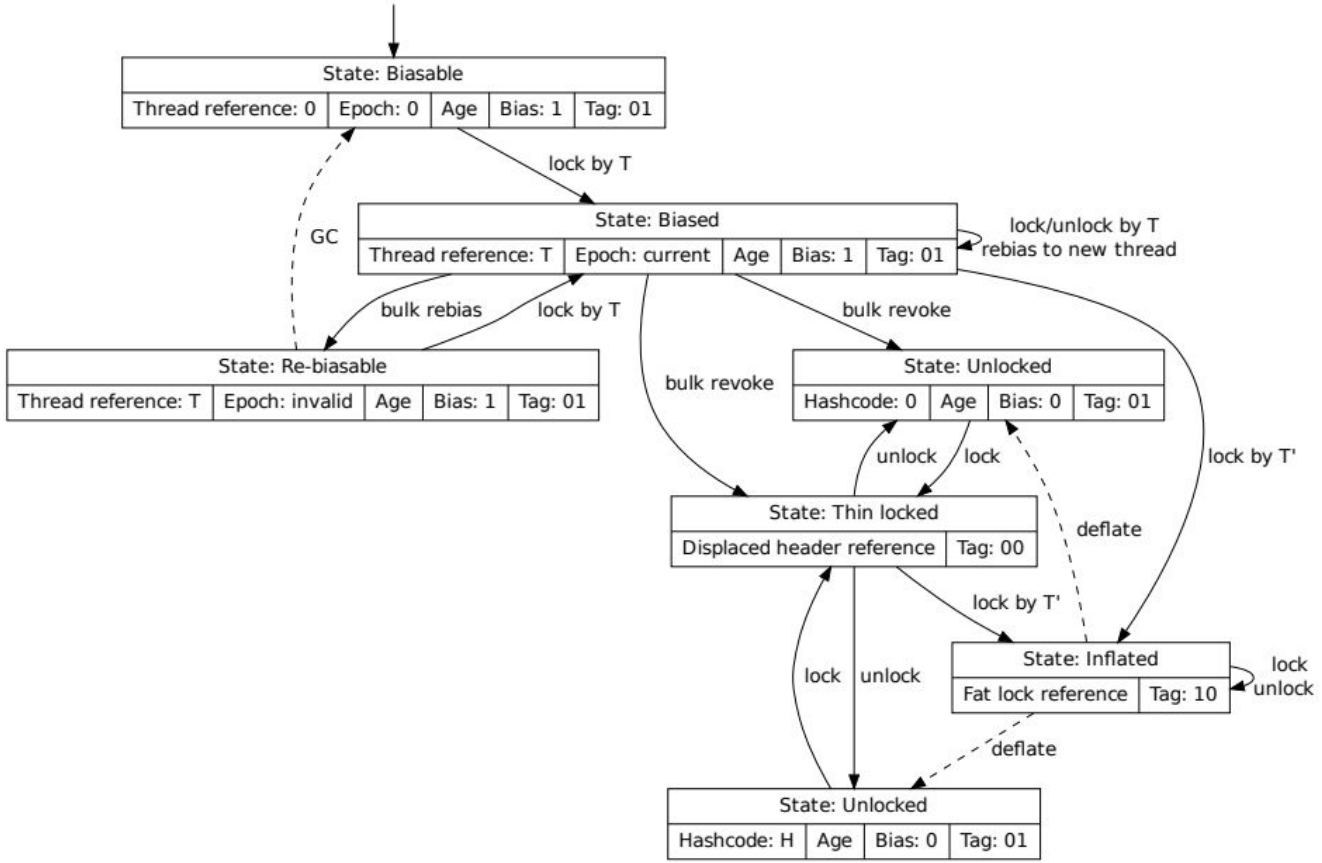


Figure 3.6. Simplified state graph of the mark word in HotSpot.

А почему все так долго делают?

- Оптимизация называется `Biased Locking`
- Реализована во многих JVM, включая HotSpot до Java 19
- Звучит так, что все круто и просто!
- Но есть нюанс: дикая сложность реализации в рантайме

А почему все так долго делают?

- Оптимизация называется `Biased Locking`
- Реализована во многих JVM, включая HotSpot до Java 19
- Звучит так, что все круто и просто!
- Но есть нюанс: дикая сложность реализации в рантайме
- А еще ухудшение производительности на некоторых примерах (Cassandra)

А почему все так долго делают?

- Оптимизация называется `Biased Locking`
- Реализована во многих JVM, включая HotSpot до Java 19
- `DEPRECATED in JDK 15`
- `REMOVED in JDK 19`

А почему все так долго делают?

- Оптимизация называется `Biased Locking`
- Реализована во многих JVM, включая HotSpot до Java 19
- `DEPRECATED in JDK 15`
- `REMOVED in JDK 19`

Теперь все довольны? Так ведь?

This code slows down considerably when biased locking is not available.(c)

```
boolean contentChanged = false;
BufferedInputStream oldContent = ...;
BufferedInputStream newContent = ...;

try {
    int newByte = newContent.read();
    int oldByte = oldContent.read();
    while (newByte != -1 && oldByte != -1 && newByte == oldByte) {
        newByte = newContent.read();
        oldByte = oldContent.read();
    }
    contentChanged = newByte != oldByte;
} catch (IOException e) {
    contentChanged = true;
}
...
```

https://mail.openjdk.org/pipermail/hotspot-dev/2022-October/065496.html

Последствия

```
public synchronized int read() throws IOException {
    if (pos >= count) {
        fill();
        if (pos >= count)
            return -1;
    }
    return getBufIfOpen()[pos++] & 0xff;
}
```

java.io.BufferedInputStream.java

https://mail.openjdk.org/pipermail/hotspot-dev/2022-October/065496.html

Последствия

https://gist.github.com/shipilev/71fd881f7cbf3f87028bf87385e84889

Последствия

```
private void readOut(Blackhole bh, InputStream s) throws IOException {
    int b;
    while ((b = s.read()) != -1) {
        bh.consume(b);
    }
}

@Benchmark
public void fis(Blackhole bh) throws IOException {
    try (InputStream s = new new FileInputStream(file)) {
        readOut(bh, s);
    }
}
```

https://gist.github.com/shipilev/71fd881f7cbf3f87028bf87385e84889

Последствия

```
private void readOut(Blackhole bh, InputStream s) throws IOException {
    int b;
    while ((b = s.read()) != -1) {
        bh.consume(b);
    }
}

@Benchmark
public void fis_bis(Blackhole bh) throws IOException {
    try (InputStream s = new BufferedInputStream(new FileInputStream(file))) {
        readOut(bh, s);
    }
}
```

https://gist.github.com/shipilev/71fd881f7cbf3f87028bf87385e84889

Последствия

```
# ---- JDK 17, +BiasedLocking
Buffers.fis                1000  avgt   10     319.211 ±    2.068  us/op
Buffers.fis_bis            1000  avgt   10       5.855 ±    0.027  us/op

# ----  JDK 19 (no biased locking support)
Buffers.fis                1000  avgt   10     310.152 ±    2.300  us/op
Buffers.fis_bis            1000  avgt   10      13.626 ±    0.032  us/op
```

https://gist.github.com/shipilev/71fd881f7cbf3f87028bf87385e84889

Последствия

- Вилка: слишком сложный код в рантайме, который трудно поддерживать или клиенты, у которых теперь тормозит код
- Хорошего решения нет, но удаление `biased locking` выходит боком (жалобы уже есть)
- В других JVM `biased locking` остался

Разработка JVM - это:

1. Challenge!
   Научный и
   инженерный

Разработка JVM - это:

1. Challenge!
   Научный и
   инженерный

2. Детектив

Разработка JVM - это:

1. Challenge!
   Научный и
   инженерный

2. Детектив

3. Общение с клиентами
   Java программистами

Можно ли стать ~~гномом~~ JVM инженером?

Можно ли стать ~~гномом~~ JVM инженером?

Конечно!

Стоит начать с чтения:

`shipilev.net/jvm/anatomy-quarks`

Можно ли стать ~~гномом~~ JVM инженером?

Конечно!

Стоит начать с чтения:

`shipilev.net/jvm/anatomy-quarks`

А потом сорцы:

`github.com/openjdk/jdk/tree/master/src/hotspot`
`github.com/oracle/graal`

831 Open ✓ 2,467 Closed | Author | Label | Projects | Milestones | Assignee | Sort

- Linux Core Dump with OOM `bug`
  #5470 opened 4 hours ago by pminearo
- Loop vectorization in GraalVM CE JIT compilations `compiler` `feature`
  #5468 opened 16 hours ago by gergo- Future
- native-image-agent generates non existent classes for Swing application `bug` `native-image` 1
  #5450 opened 2 days ago by AlexanderScherbatiy
- Native JFR: Hangs or segfaults intermittently on Quarkus app shutdown `bug` `native-image`
  #5434 opened 4 days ago by Karm
- `native-image` crashes when passing --gc=epsilon with some `serialGCOnly` option `bug` `native-image`
  #5432 opened 4 days ago by zakkak
- classpath vs. modulepath - errors and different results `bug` `native-image`
  #5428 opened 4 days ago by fipro78
- how to use Date.toString() in java? `javascript` `truffle` 1
  #5427 opened 4 days ago by kingkong3
- JFR Support in Native Image `feature`
  #5410 opened 8 days ago by roberttoyonaga 100+ Future
- [GR-42465] JFR jdk.ContainerConfiguration event reports wrong values in a container `bug` `native-image` 1
  #5396 opened 10 days ago by jerboaa
- Push image to docker hub `feature` 1
  #5378 opened 12 days ago by zgqq
- [Documentation] Clarify location of native-image-agent `bug` `native-image` 5
  #5375 opened 13 days ago by DaveJarvis
- Improve clarity around using resources `documentation` `enhancement` 2
  #5373 opened 13 days ago by DaveJarvis

Q & A

@dbg_nsk

@ugliansky

Последствия

```
# ---- JDK 17, +BiasedLocking
Buffers.fis              1000  avgt   10     319.211 ±    2.068  us/op
Buffers.fis_bis          1000  avgt   10       5.855 ±    0.027  us/op

# ---- JDK 17, -BiasedLocking
Buffers.fis              1000  avgt   10     316.564 ±    2.085  us/op
Buffers.fis_bis          1000  avgt   10      13.120 ±    0.059  us/op

# ----  JDK 19 (no biased locking support)
Buffers.fis              1000  avgt   10     310.152 ±    2.300  us/op
Buffers.fis_bis          1000  avgt   10      13.626 ±    0.032  us/op
```

https://gist.github.com/shipilev/71fd881f7cbf3f87028bf87385e84889