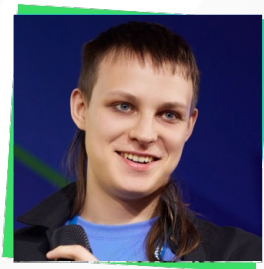


Практические аспекты FFM API: приручаем Rapama



Пётр Портнов
ведущий разработчик
информационных систем, Ozon

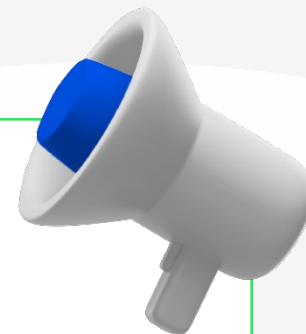


Пётр Портнов

Разработчик поисковой платформы Ozon,
open-source контрибутор, читаю пары
по Java и по алгоритмам



pportnov@ozon.ru



Дисклеймеры & Вводные

01

Доклад предназначен для всех



02

Будут мемы



03

Примеры кода упрощены



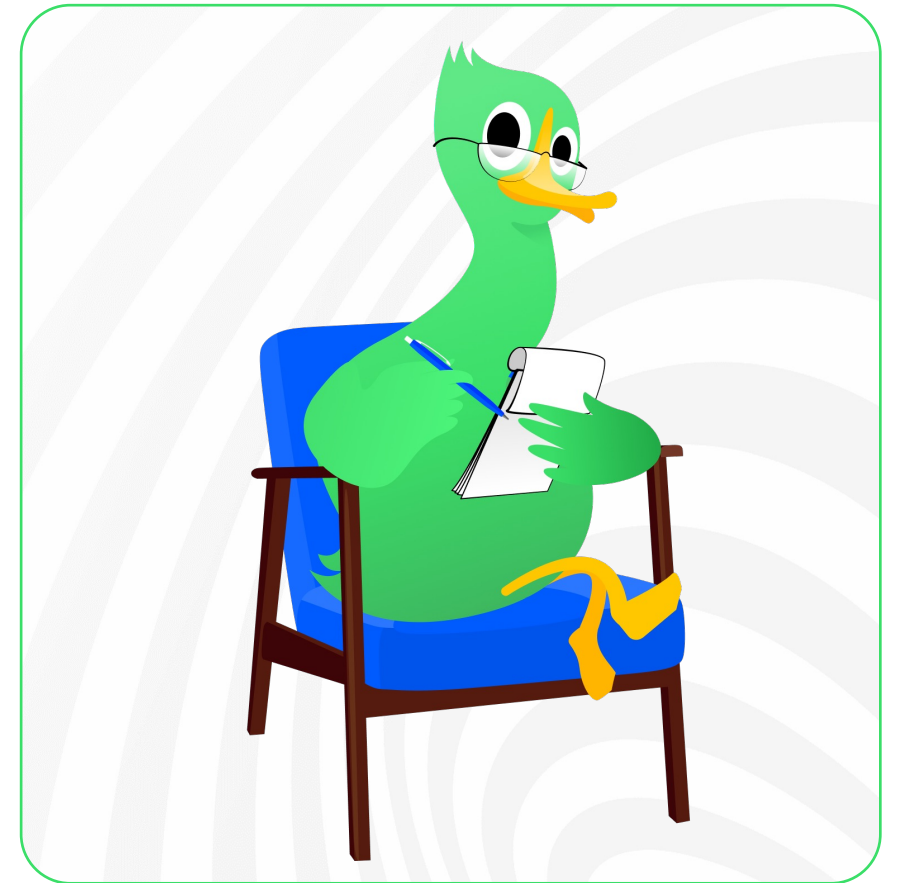
04

Произношение названий может варьировать



05

Делайте выводы осознанно



AGENDA

01

Что есть Panama?



02

Простые биндинги



03

Jextract
& Биндинги к hnswlib



04

Арены
& Биндинги к CatBoost



05

Panama in the wild



06

Заключение



Ответы и активность из зала – приветствуются



01



Что есть Ранама?

Вызов функции

Старый-добрый

```
void main() {  
    var result = bar(1, 20);  
    IO.println(result);  
}
```

```
long bar(int x, long y) {  
    return x * y;  
}
```

Вызов функции

Старый-добрый

```
void main() {  
    var result = bar(1, 20);  
    IO.println(result);  
}
```

```
long bar(int x, long y) {  
    return x * y;  
}
```

Вызов функции

Старый-добрый

```
void main() {  
    var result = bar(1, 20);  
    IO.println(result);  
}
```

```
long bar(int x, long y) {  
    return x * y;  
}
```

Другая функция

А вот эта функция на чём написана?

```
int square(int num) {  
    return num * num;  
}
```

Другая функция

А вот эта функция на чём написана?

```
int square(int num) {  
    return num * num;  
}
```



Другая функция

А вот эта функция на чём написана?

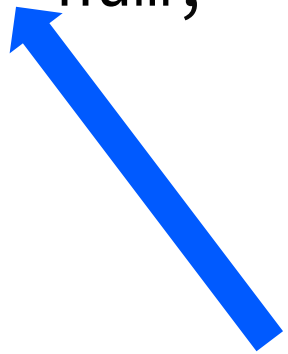
```
// my_square.c  
int square(int num) {  
    return num * num;  
}
```



Другая функция

А вот эта функция на чём написана?

```
// my_square.c  
int square(int num) {  
    return num * num;  
}
```



Что нам стоит вызвать нативную функцию?

Смысл то тот же

```
// my_square.c  
int square(int num) {  
    return num * num;  
}
```

```
// Main.java  
void main() {  
    var result = square(123);  
    IO.println(result);  
}
```

Что нам стоит вызвать нативную функцию?

Смысл то тот же

```
// my_square.c
int square(int num) {
    return num * num;
}

// Main.java
void main() {
    var result = square(123);
    IO.println(result);
}
```



Есть же JNI!

Достаточно всего лишь написать `native`

```
// MySquare.java  
static native int square(int num);  
// Функция она где-то там
```

```
// Main.java  
void main() {  
    var result = MySquare.square(123);  
    IO.println(result);  
}
```

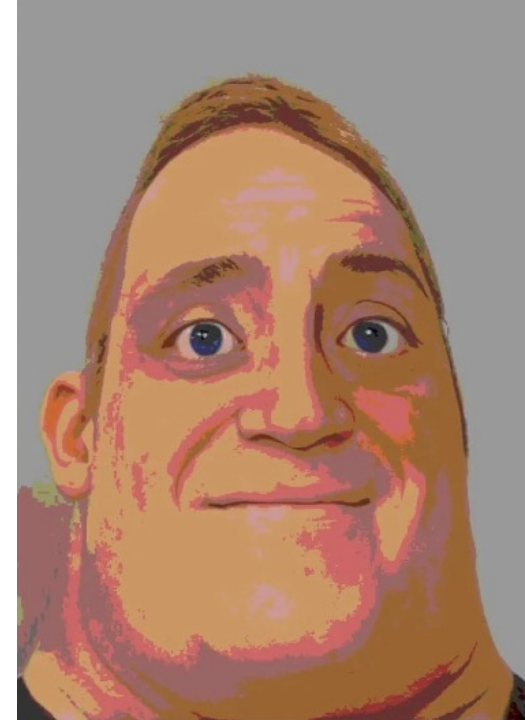


Достаточно всего лишь написать native

И немножечко клея...

```
// my_square_jni.c
#include <jni.h>
#include "my_square.c"

JNIEXPORT jint JNICALL
Java_ru_jokerconf_pportnov_jni_MySquare_square(
    JNIEnv *env, jclass clazz,
    jint num
) {
    return square(num);
}
```



И немножечко клея...

Множечко...

```
// my_square_jni.c
#include <jni.h>
#include "my_square.c"

JNIEXPORT jint JNICALL
Java_ru_jokerconf_pportnov_jni_MySquare_square(
    JNIEnv *env, jclass clazz,
    jint num
) {
    return square(num);
}
```

Множечко...

И как-то в рантайме прилинковать эту библиотеку...

```
static {  
    System.loadLibrary("libmysquarejni");  
}
```

И как-то в рантайме прилинковать эту библиотеку...

Которую ещё надо собрать...

```
$ cc -O2 -fPIC -shared \  
-I "$JAVA_HOME/include" \  
-I "$JAVA_HOME/include/linux" \  
'native/my_square_jni.c' \  
-lm \  
-o 'libmysquarejni'
```

Которую ещё надо собрать...

Под разные архитектуры...

```
$ cc -O2 -fPIC -shared \  
-I "$JAVA_HOME/include" \  
-I "$JAVA_HOME/include/darwin" \  
'native/my_square_jni.c' \  
-lm \  
-o 'libmysquarejni'
```

Под разные архитектуры...

А ещё...

Cross-компиляция не всегда доступна



Нужно уметь делать это при сборке



Нужно уметь делать это в CI



И другие увлекательные нюансы



Обычно мы просто
хотим позвать
существующую
функцию из уже
готовой библиотеки



Не нам одним

JEP 412: Foreign Function & Memory API (Incubator)

<i>Owner</i>	Maurizio Cimadamore
<i>Type</i>	Feature
<i>Scope</i>	JDK
<i>Status</i>	Closed / Delivered
<i>Release</i>	17
<i>Component</i>	core-libs
<i>Discussion</i>	panama dash dev at openjdk dot java dot net
<i>Effort</i>	M
<i>Duration</i>	M
<i>Relates to</i>	JEP 389: Foreign Linker API (Incubator) JEP 393: Foreign-Memory Access API (Third Incubator) JEP 419: Foreign Function & Memory API (Second Incubator) JEP 424: Foreign Function & Memory API (Preview)
<i>Reviewed by</i>	Adam Pockock, Alex Buckley, Paul Sandoz
<i>Created</i>	2021/04/10 21:05
<i>Updated</i>	2022/03/02 17:11
<i>Issue</i>	8265033

Summary

Introduce an API by which Java programs can interoperate with code and data outside of the Java runtime. By efficiently invoking foreign functions (i.e., code outside the JVM), and by safely accessing foreign memory (i.e., memory not managed by the JVM), the API enables Java programs to call native libraries and process native data without the brittleness and danger of JNI.

Не нам одним

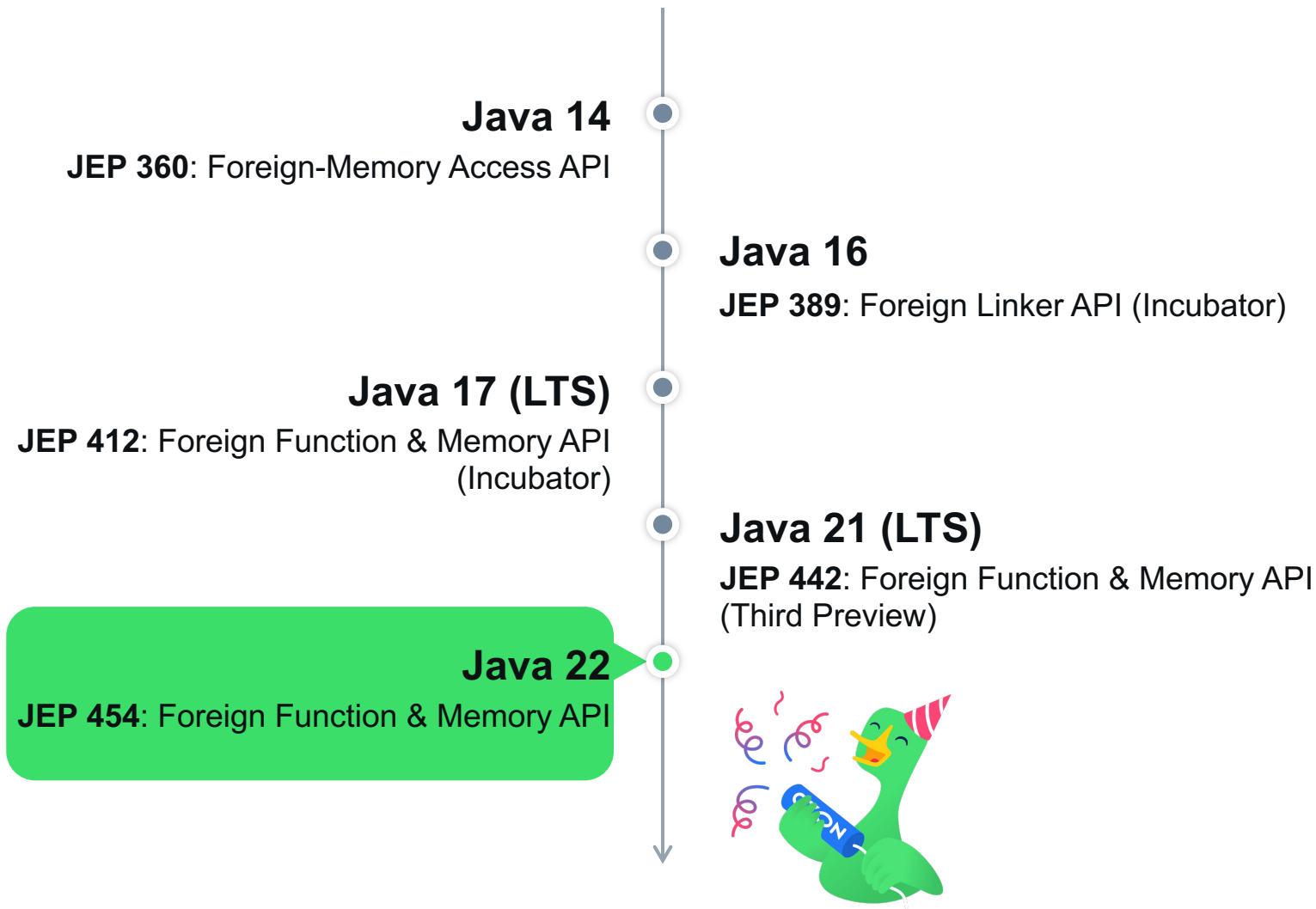
JEP 412: Foreign Function & Memory API (Incubator)

Owner	Mauricio Comandatore
Type	Feature
Scope	JDK
Status	Closed/Delivered
Release	17
Component	core-libs
Discussion	panama-dev at openjdk dot java dot net
Effort	M
Duration	M
Relates to	JEP 386: Foreign Linker API (Incubator) JEP 383: Foreign-Memory Access API (Third Incubator) JEP 418: Foreign Function & Memory API (Second Incubator) JEP 424: Foreign Function & Memory API (Preview)
Reviewed by	Adam Rozock, Alex Buckley, Paul Sandhu
Created	2021/04/10 21:05
Updated	2022/03/02 17:11
Issue	8285033

Summary

Introduce an API by which Java programs can interoperate with code and data outside of the Java runtime. By efficiently invoking foreign functions (i.e., code outside the JVM), and by safely accessing foreign memory (i.e., memory not managed by the JVM), the API enables Java programs to call native libraries and process native data without the brittleness and danger of JNI.

Таймлайн FFM API



<https://openjdk.org/jeps/0>

FFM API (Panama) – API для доступа к внешним функциям и памяти из Java



02



Простые биндинги

Начнём с простого примера

Кто пишет числодробилки на Java?





Мы!
И делаем это
эффективно



 <https://s.ozon.ru/a8fFoMq>



 <https://s.ozon.ru/Za1cANw>

Что нам стоит посчитать `pow(x, y)`?

На `float`'ах

Что нам стоит посчитать `pow(x, y)`?

На `float`'ax

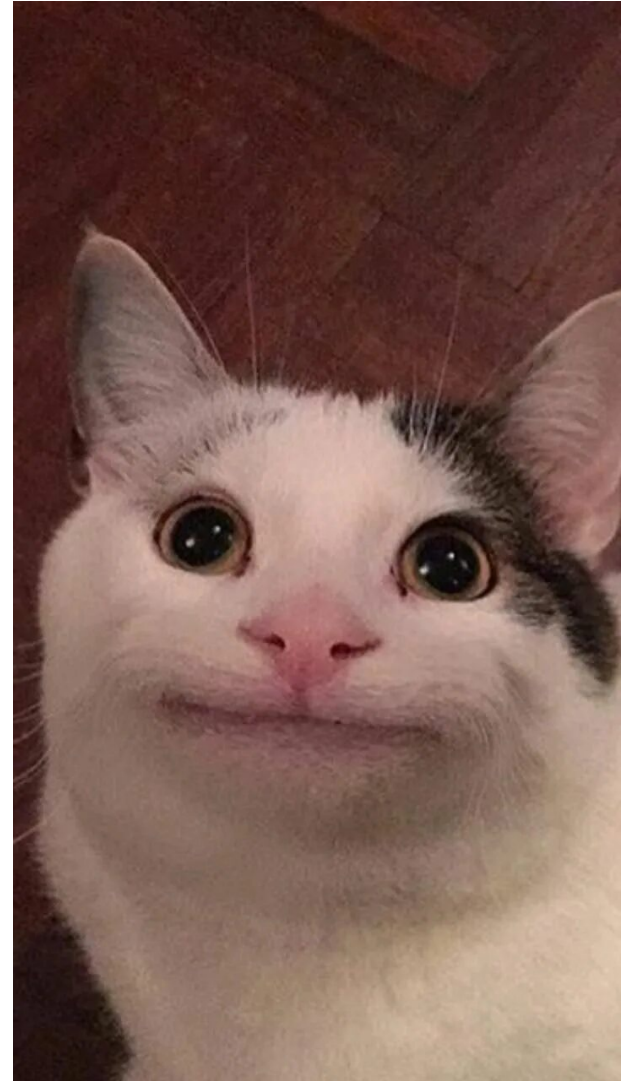
```
static float floatPow(float x, float y) {  
    return Math.pow(x, y);  
}
```

Что нам стоит посчитать `pow(x, y)`?

На `float`'ax

```
static float floatPow(float x, float y) {  
    return Math.pow(x, y);  
}
```

error: incompatible types:
possible lossy conversion
from double to float
}



Приходится roundtrip'ать

Через double'ы

```
static float floatPow(float x, float y) {  
    return (float) Math.pow(x, y);  
}
```

Приходится roundtrip'ать

Через double'ы

```
// Если явно записать преобразования  
static float floatPow(float x, float y) {  
    return (float) Math.pow((double) x, (double) y);  
}
```

Проблема: вычисления
на double'ах ощутимо
дороже



Интернететы говорят

Что мы правы!



Stack Overflow

Ответов: 8 · 15 лет назад :

double or float, which is faster? [duplicate] - c++

Floats are faster than doubles when you don't need double's precision and you are memory-bandwidth bound and your hardware doesn't carry a ...

Is using **double** faster than **float**? 10 ответов 6 авг. 2010 г.

c# - **Float** vs **Double** Performance 5 ответов 6 янв. 2009 г.

Другие результаты с сайта stackoverflow.com

Ну наверняка JIT заметит

Что тут float-арифметика

Ну наверняка JIT заметит

Что тут float-арифметика

```
@State( Scope.Benchmark )
public static class BenchmarkState {
    @Param( "1000" ) public int size;
    public float[] floats;
    @Setup
    public void setup() {
        var random = new Random( 12345 );
        floats = new float[ 2 * size ];
        for ( var index = 0; index < 2 * size; index++ )
            floats[ index ] = random.nextFloat();
    }
}
```

Ну наверняка JIT заметит

Что тут float-арифметика

```
@State( Scope.Benchmark )
public static class BenchmarkState {
    @Param( "1000" ) public int size;
    public float[] floats;
    @Setup
    public void setup() {
        var random = new Random(12345);
        floats = new float[2 * size];
        for (var index = 0; index < 2 * size; index++)
            floats[index] = random.nextFloat();
    }
}
```

Ну наверняка JIT заметит

Что тут float-арифметика

@Benchmark

```
public void pow(BenchmarkState state, Blackhole bh) {  
    var floats = state.floats;  
    for (var index = 0; index < floats.length; index += 2) {  
        var base = floats[index];  
        var exponent = floats[index + 1];  
  
        bh.consume((float) Math.pow(base, exponent));  
    }  
}
```

Ну наверняка JIT заметит

Что тут float-арифметика

@Benchmark

```
public void pow(BenchmarkState state, Blackhole bh) {  
    var floats = state.floats;  
    for (var index = 0; index < floats.length; index += 2) {  
        var base = floats[index];  
        var exponent = floats[index + 1];  
  
        bh.consume((float) Math.pow(base, exponent));  
    }  
}
```



Окружение бенчмарков

x86_64

- **CPU:**
Intel Xeon Gold 6240R
(96) @ 4.000GHz
- **OS:**
Ubuntu 22.04.5 LTS
x86_64
- **Kernel:**
6.8.0-124-generic

ARM

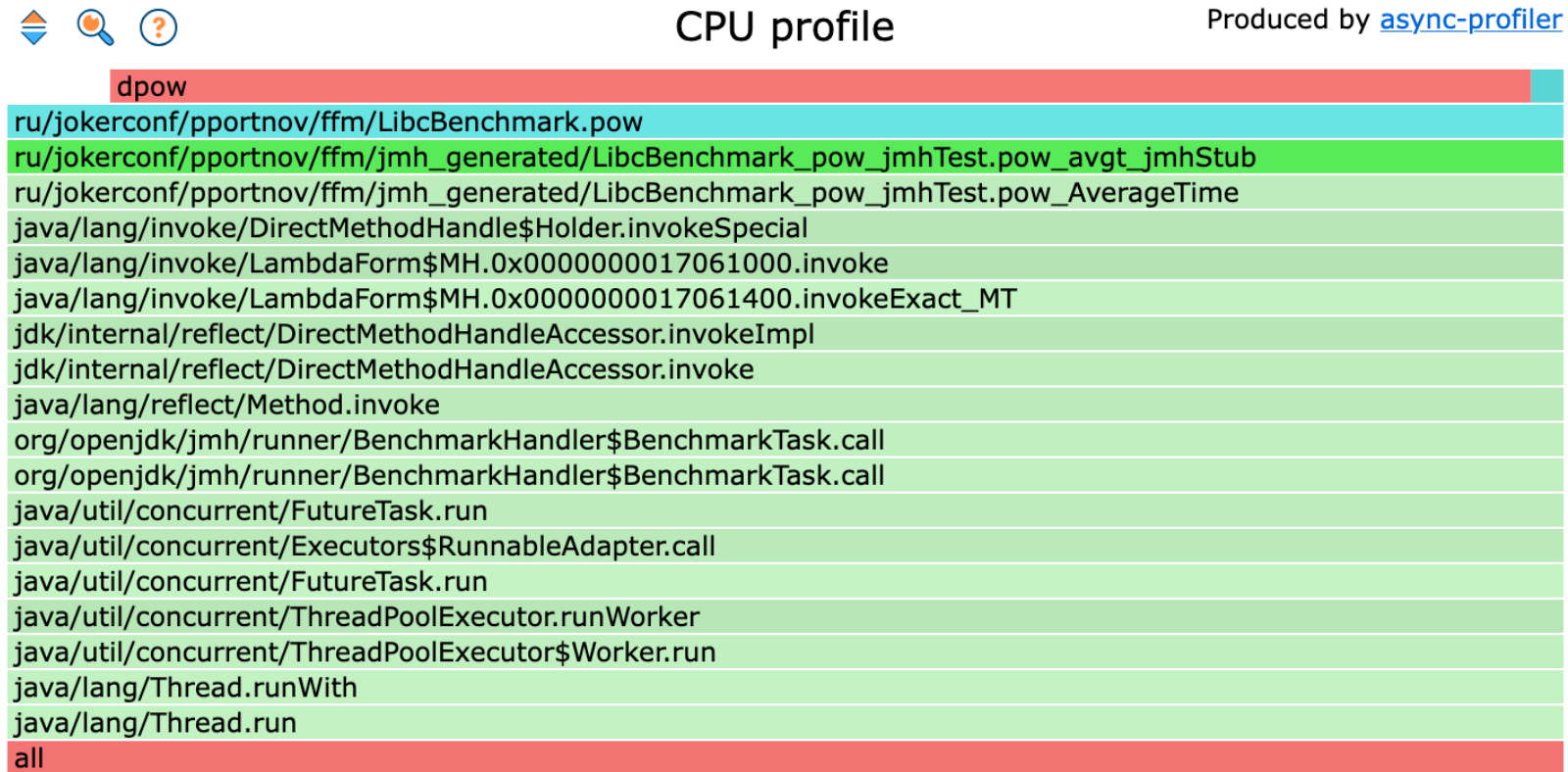
- **CPU:**
Apple M4 Pro
(8+4) @ 4.51 GHz
- **OS:**
macOS Tahoe 26.6.2
(25G83) arm64
- **Kernel:**
Darwin 25.6.0

Как думаете? Заменял
ли JIT вызовы на
`float-арифметику`



JIT компилирует вызов в dpow

На x86_64



JIT компилирует вызов в `SharedRuntime::dpow`

На ARM



CPU profile

Produced by [async-profiler](#)



Но ведь есть нативная™ функция

Для подсчёта float-pow

cppreference.com [Create account](#)

Page [Discussion](#) Standard revision: [Diff](#) [Read](#) [View source](#) [View history](#)

[C](#) [Numerics](#) [Common mathematical functions](#)

pow, powf, powl

Defined in header `<math.h>`

```
float powf( float base, float exponent );
```

(1) (since C99)

```
double pow( double base, double exponent );
```

(2)

```
long double powl( long double base, long double exponent );
```

(3) (since C99)

Defined in header `<tgmath.h>`

```
#define pow( base, exponent )
```

(4) (since C99)

1-3) Computes the value of `base` raised to the power `exponent`.

Но ведь есть нативная™ функция

Для подсчёта float-pow

cppreference.com [Create account](#)

Page [Discussion](#) Standard revision: [Diff](#) [Read](#) [View source](#) [View history](#)

[C](#) [Numerics](#) [Common mathematical functions](#)

pow, powf, powl

Defined in header `<math.h>`

```
float powf( float base, float exponent );
```

(1) (since C99)

```
double pow( double base, double exponent );
```

(2)

```
long double powl( long double base, long double exponent );
```

(3) (since C99)

Defined in header `<tgmath.h>`

```
#define pow( base, exponent )
```

(4) (since C99)

1-3) Computes the value of `base` raised to the power `exponent`.

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker( );

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup( ).findOrThrow( "powf" ),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName( "base" ),
            ValueLayout.JAVA_FLOAT.withName( "exponent" )
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();  
  
private static final MethodHandle POWF = LINKER  
    .downcallHandle(  
        LINKER.defaultLookup().findOrThrow("powf"),  
        FunctionDescriptor.of(  
            ValueLayout.JAVA_FLOAT,  
            ValueLayout.JAVA_FLOAT.withName("base"),  
            ValueLayout.JAVA_FLOAT.withName("exponent")  
        )  
    );
```



Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

Создадим `MethodHandle` для доступа к функции

```
private static final Linker LINKER = Linker.nativeLinker();

private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        )
    );
```

Настало время FFM API

И сделаем над ним функцию-обёртку

```
public static float powf(float base, float exponent) {  
    try {  
        return (float) POWF.invokeExact(base, exponent);  
    } catch (Throwable x) {  
        throw new RuntimeException("powf failed", x);  
    }  
}
```

Настало время FFM API

И сделаем над ним функцию-обёртку

```
public static float powf(float base, float exponent) {  
    try {  
        return (float) POWF.invokeExact(base, exponent);  
    } catch (Throwable x) {  
        throw new RuntimeException("powf failed", x);  
    }  
}
```

Но тут появился какой-то каст...

А мы боролись за перф!

```
public static float powf(float base, float exponent) {  
    try {  
        return (float) POWF.invokeExact(base, exponent);  
    } catch (Throwable x) {  
        throw new RuntimeException(x);  
    }  
}
```



Polymorphic Signature

Это особые функции в JDK

```
// MethodHandle.java (JDK)
```

```
@IntrinsicCandidate
```

```
public final native @PolymorphicSignature Object invokeExact(  
    Object... Args  
) throws Throwable;
```

```
// Libc.java
```

```
GETSTATIC ru/jokerconf/pportnov/ffmpeg/Libc.P0WF  
    : Ljava/lang/invoke/MethodHandle;
```

```
FLOAD 0
```

```
FLOAD 1
```

```
INVOKEVIRTUAL java/lang/invoke/MethodHandle.invokeExact (FF)F
```

```
FRETURN
```

Polymorphic Signature

Это особые функции в JDK

```
// MethodHandle.java (JDK)
```

```
@IntrinsicCandidate
```

```
public final native @PolymorphicSignature Object invokeExact(  
    Object... Args  
) throws Throwable;
```

```
// Libc.java
```

```
GETSTATIC ru/jokerconf/pportnov/ffmpeg/Libc.POWF  
    : Ljava/lang/invoke/MethodHandle;
```

```
FLOAD 0
```

```
FLOAD 1
```

```
INVOKEVIRTUAL java/lang/invoke/MethodHandle.invokeExact (FF)F
```

```
FRETURN
```

Polymorphic Signature

Это особые функции в JDK

```
// MethodHandle.java (JDK)
@IntrinsicCandidate
public final native @PolymorphicSignature Object invokeExact(
    Object... Args
) throws Throwable;
```

```
// Libc.java
GETSTATIC ru/jokerconf/pportnov/ffmpeg/Libc.P0WF
    : Ljava/lang/invoke/MethodHandle;
```

```
FLOAD 0
```

```
FLOAD 1
```

```
INVOKEVIRTUAL java/lang/invoke/MethodHandle.invokeExact (FF)F
```

```
FRETURN
```

Теперь нужно всё это замерить

Бенчмарк у нас уже есть

```
@Benchmark
public void powf(BenchmarkState state, Blackhole bh) {
    var floats = state.floats;
    for (var index = 0; index < floats.length; index += 2) {
        var base = floats[index];
        var exponent = floats[index + 1];

        bh.consume(Libc.powf(base, exponent));
    }
}
```

Теперь нужно всё это замерить

Бенчмарк у нас уже есть

```
@Benchmark
public void powf(BenchmarkState state, Blackhole bh) {
    var floats = state.floats;
    for (var index = 0; index < floats.length; index += 2) {
        var base = floats[index];
        var exponent = floats[index + 1];

        bh.consume(Libc.powf(base, exponent));
    }
}
```

Какие результаты?

Ваши варианты

Функция	Время (x86_64)	Время (ARM)
pow		
powf		

Какие результаты?

Стабильный выигрыш на x86_64

Функция	Время (x86_64)	Время (ARM)
pow	21.952 ± 0.355 us/op	
powf	15.077 ± 0.001 us/op	

Какие результаты?

Радикальный выигрыш на ARM

Функция	Время (x86_64)		Время (ARM)	
pow	21.952	± 0.355 us/op	17.707	± 0.242 us/op
powf	15.077	± 0.001 us/op	3.442	± 0.013 us/op

Выводы были бы нечестными без JNI

Добавим и его в сравнение

Выводы были бы нечестными без JNI

Добавим и его в сравнение

```
// libcjni.c
#include <jni.h>
#include <math.h>

JNIEXPORT jfloat JNICALL
Java_ru_jokerconf_pportnov_jni_LibcJni_powf(
    JNIEnv *env, jclass clazz,
    jfloat base, jfloat exponent
) {
    return powf(base, exponent);
}
```

Выводы были бы нечестными без JNI

Добавим и его в сравнение

```
// LibcJni.java
static {
    System.loadLibrary("libcjni");
}

public static native float powf(
    float base,
    float exponent
);
```

Выводы были бы нечестными без JNI

Добавим и его в сравнение

```
// LibcJni.java
static {
    System.loadLibrary("libcjni");
}

public static native float powf(
    float base,
    float exponent
);
```

Выводы были бы нечестными без JNI

Добавим и его в сравнение

```
@Benchmark
public void powfJni(BenchmarkState state, Blackhole bh) {
    var floats = state.floats;
    for (var index = 0; index < floats.length; index += 2) {
        var base = floats[index];
        var exponent = floats[index + 1];

        bh.consume(LibcJni.powf(base, exponent));
    }
}
```

И получим

Ваши варианты?

Функция	Время (x86_64)		Время (ARM)	
pow	21.952	± 0.355 us/op	17.707	± 0.242 us/op
powf	15.077	± 0.001 us/op	3.442	± 0.013 us/op
powfJni				

И получим

Выигрыш близкий к `powf` через `Rapata`, но худший

Функция	Время (x86_64)		Время (ARM)	
<code>pow</code>	21.952	± 0.355 us/op	17.707	± 0.242 us/op
<code>powf</code>	15.077	± 0.001 us/op	3.442	± 0.013 us/op
<code>powfJni</code>	16.054	± 0.050 us/op	3.847	± 0.031 us/op

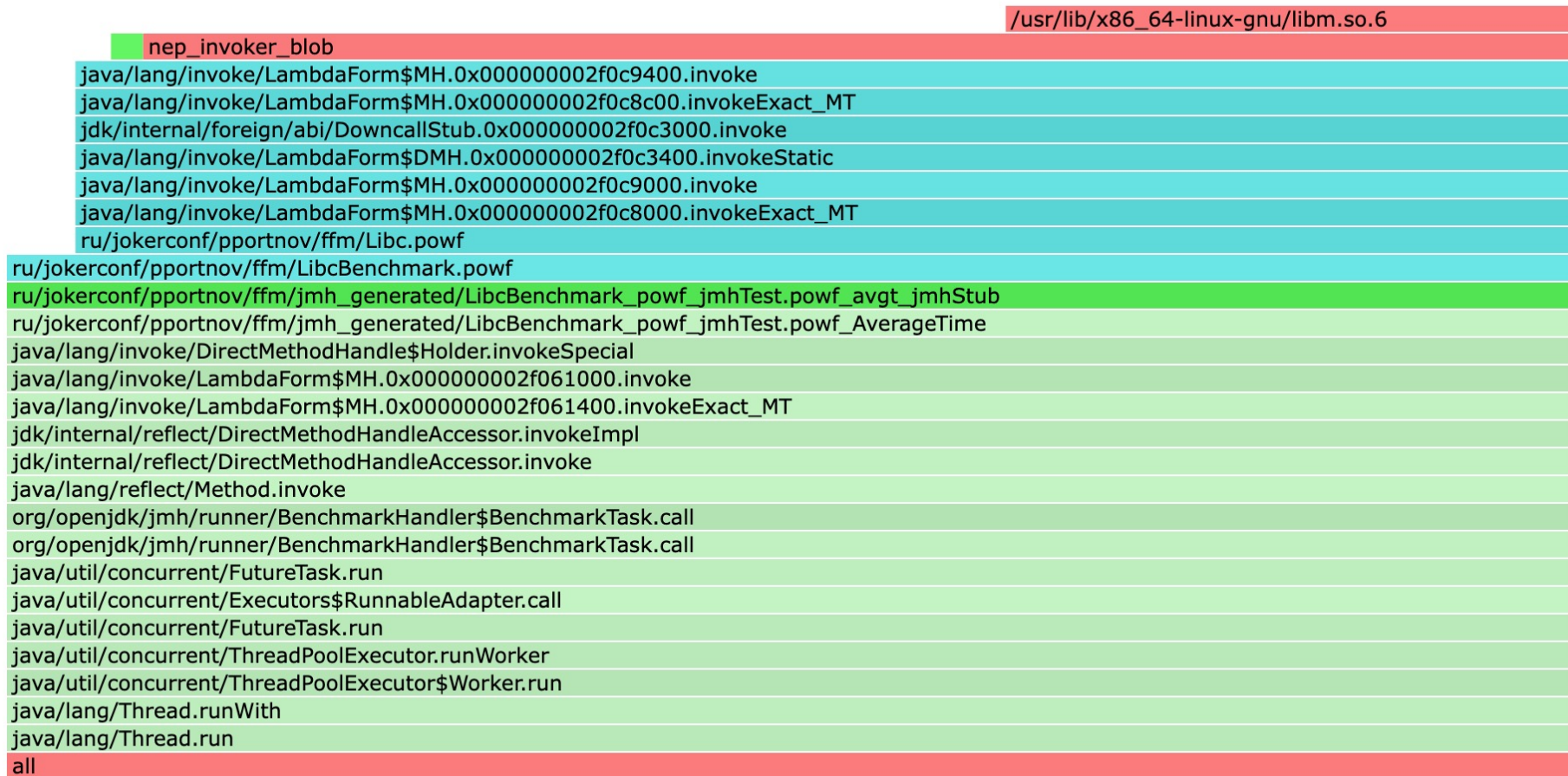
Как это выглядит на профилях

Панама



CPU profile

Produced by [async-profiler](#)



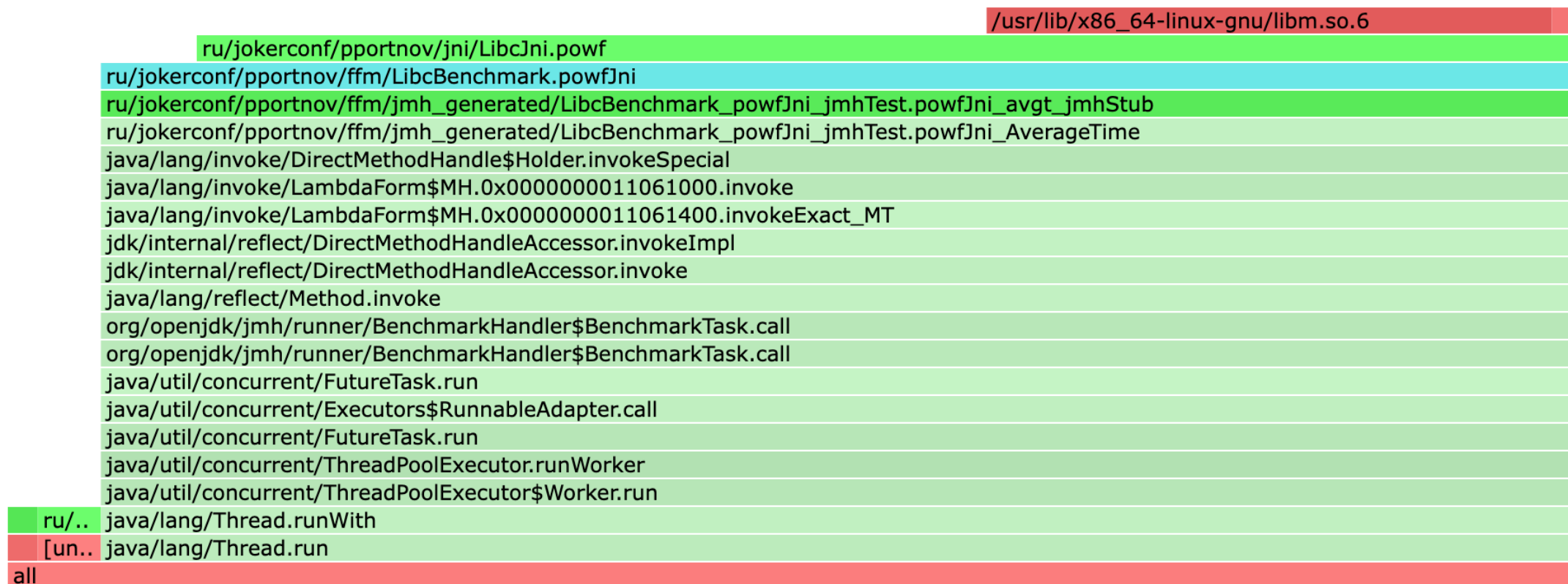
Как это выглядит на профилях

JNI



CPU profile

Produced by [async-profiler](#)



We need to go deeper: ASM для Panama

x86_64: подготовка к вызову

```
vmovss 0x10(%rdx,%rbp,4),%xmm0
        ; base
vmovss 0x14(%rdx,%rbp,4),%xmm1
        ; exponent
movabs $0x7d64ef4adef0,%rsi
movabs $0x101fe4fc0,%rdx
        ; {oop(a 'jdk/internal/foreign/abi/NativeEntryPoint')}
call   0x00007d64d7a1e4e0
        ; *invokestatic linkToNative
        ; - jdk.internal.foreign.abi.DowncallStub::invoke
        ; - ru.jokerconf.pportnov ffm.Libc::powf (line 43)
```

We need to go deeper: ASM для Panama

x86_64: подготовка к вызову

```
vmovss 0x10(%rdx,%rbp,4),%xmm0
        ; base
vmovss 0x14(%rdx,%rbp,4),%xmm1
        ; exponent
movabs $0x7d64ef4adef0,%rsi
movabs $0x101fe4fc0,%rdx
        ; {oop(a 'jdk/internal/foreign/abi/NativeEntryPoint')}
call   0x00007d64d7a1e4e0
        ; *invokestatic linkToNative
        ; - jdk.internal.foreign.abi.DowncallStub::invoke
        ; - ru.jokerconf.pportnov ffm.Libc::powf (line 43)
```

We need to go deeper: ASM для Panama

x86_64: подготовка к вызову

```
vmovss 0x10(%rdx,%rbp,4),%xmm0
        ; base
vmovss 0x14(%rdx,%rbp,4),%xmm1
        ; exponent
movabs $0x7d64ef4adef0,%rsi
movabs $0x101fe4fc0,%rdx
        ; {oop(a 'jdk/internal/foreign/abi/NativeEntryPoint')}
call   0x00007d64d7a1e4e0
        ; *invokestatic linkToNative
        ; - jdk.internal.foreign.abi.DowncallStub::invoke
        ; - ru.jokerconf.pportnov ffm.Libc::powf (line 43)
```

We need to go deeper: ASM

x86_64: stub вызова

```
mov    %rbp,0x490(%r15)    ; last_Java_frame
mov    %rsp,0x480(%r15)    ; last_Java_sp
movl   $0x4,0x52c(%r15)    ; state = _thread_in_native
call   *%r10               ; → powf (libm)
movl   $0x5,0x52c(%r15)    ; state = _thread_in_native_trans
lock addl $0x0,-0x40(%rsp) ; memory barrier
cmp    0x28(%r15),%rbp     ; safepoint/GC-handshake poll ← 37% всех циклов!
ja     slow_path           ; (GC подождёт? нет – выполняем handshake)
movl   $0x8,0x52c(%r15)    ; state = _thread_in_Java
```

We need to go deeper: ASM

x86_64: stub вызова

```
mov    %rbp,0x490(%r15)    ; last_Java_frame
mov    %rsp,0x480(%r15)    ; last_Java_sp
movl   $0x4,0x52c(%r15)    ; state = _thread_in_native
call  *%r10               ; → powf (libm)
movl   $0x5,0x52c(%r15)    ; state = _thread_in_native_trans
lock addl $0x0,-0x40(%rsp) ; memory barrier
cmp    0x28(%r15),%rbp     ; safepoint/GC-handshake poll ← 37% всех циклов!
ja     slow_path           ; (GC подождёт? нет – выполняем handshake)
movl   $0x8,0x52c(%r15)    ; state = _thread_in_Java
```

Да, многовато

И для этого есть решение

critical

```
static Linker.Option critical(boolean allowHeapAccess)
```

Returns a linker option used to mark a foreign function as *critical*.

A critical function is a function that has an extremely short running time in all cases (similar to calling an empty function), and does not call back into Java (e.g. using an upcall stub).

Using this linker option is a hint that some implementations may use to apply optimizations that are only valid for critical functions.

Using this linker option when linking non-critical functions is likely to have adverse effects, such as loss of performance or JVM crashes.

Critical functions can optionally allow access to the Java heap. This allows clients to pass heap memory segments as addresses, where normally only off-heap memory segments would be allowed. The memory region inside the Java heap is exposed through a temporary native address that is valid for the duration of the function call. Use of this mechanism is therefore only recommended when a function needs to do short-lived access to Java heap memory, and copying the relevant data to an off-heap memory segment would be prohibitive in terms of performance.

Parameters:

`allowHeapAccess` - whether the linked function should allow access to the Java heap.

Returns:

a linker option used to mark a foreign function as *critical*

[https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical\(boolean\)](https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical(boolean))

Наша функция – попадает под критерий критической

Можно поделиться этим знанием с рантаймом

A critical function is a function that has an extremely short running time in all cases (similar to calling an empty function), and does not call back into Java (e.g. using an upcall stub).

[https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical\(boolean\)](https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical(boolean))

Гарантии – во!

Но для нашего случая – самое то

Using this linker option when linking non-critical functions is likely to have adverse effects, such as loss of performance or JVM crashes.

[https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical\(boolean\)](https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/lang/foreign/Linker.Option.html#critical(boolean))

Указываем `critical` при создании `MethodHandle`

И... Всё

```
private static final MethodHandle POWF = LINKER
    .downcallHandle(
        LINKER.defaultLookup().findOrThrow("powf"),
        FunctionDescriptor.of(
            ValueLayout.JAVA_FLOAT,
            ValueLayout.JAVA_FLOAT.withName("base"),
            ValueLayout.JAVA_FLOAT.withName("exponent")
        ),
        Linker.Option.critical(false)
    );
```

Дало ли эффект?

Ваши варианты?

Функция	Время (x86_64)		Время (ARM)	
pow	21.952	± 0.355 us/op	17.707	± 0.242 us/op
powf	15.077	± 0.001 us/op	3.442	± 0.013 us/op
powfJni	16.054	± 0.050 us/op	3.847	± 0.031 us/op
powfCritical				

Дало

И ещё какой!

Функция	Время (x86_64)		Время (ARM)	
pow	21.952	± 0.355 us/op	17.707	± 0.242 us/op
powf	15.077	± 0.001 us/op	3.442	± 0.013 us/op
powfJni	16.054	± 0.050 us/op	3.847	± 0.031 us/op
powfCritical	9.436	± 0.011 us/op	2.840	± 0.062 us/op

Deeper again: ASM

x86_64: stub критического вызова

<code>sub \$0x10,%rsp</code>	; выравнивание под SysV ABI
<code>mov %rsi,%r10</code>	
<code>call *%r10</code>	; → powf (libm)
<code>leave</code>	
<code>ret</code>	

Thrpt-бенчмарки

Как подтверждение результата

Функция	thrpt (x86_64)	thrpt (ARM)
pow	45988.817 ± 1291.359 ops/s	56131.437 ± 515.947 ops/s
powf	66178.550 ± 224.176 ops/s	289403.474 ± 1879.845 ops/s
powfJni	62376.267 ± 116.154 ops/s	257955.689 ± 1105.689 ops/s
powfCritical	105242.096 ± 1161.990 ops/s	356140.166 ± 1988.876 ops/s

FFM API даёт средства
для простого доступа к
нативным функциям, и
бонусом – лучший перф



03



Jextract & Биндинги к hnswlib

Что такое hnswlib?

С++-библиотеке для быстрого ANN

nmslib/hnswlib

Header-only C++/python library for fast approximate nearest neighbors



👤 54

Contributors



9k

Used by



5k

Stars



853

Forks



<https://github.com/nmslib/hnswlib>



ANN – не единственный
ПОДХОД



 <https://s.ozon.ru/pHExk31>

Нюансы вне Hello World'ов

В C++ есть неймспейсы и перегрузки

// В коде

```
new hnswlib::InnerProductSpace(dim);
```

// В бинаре

```
$ strings libhnswlib_panama.so | rg InnerProductSpace
_ZN7hnswlib17InnerProductSpace13get_data_sizeEv
_ZN7hnswlib17InnerProductSpace13get_dist_funcEv
_ZN7hnswlib17InnerProductSpace19get_dist_func_paramEv
_ZN7hnswlib17InnerProductSpaceD0Ev
_ZTIN7hnswlib17InnerProductSpaceE
_ZTSN7hnswlib17InnerProductSpaceE
_ZTVN7hnswlib17InnerProductSpaceE
N7hnswlib17InnerProductSpaceE
_ZTVN7hnswlib17InnerProductSpaceE
_ZN7hnswlib17InnerProductSpace19get_dist_func_paramEv
_ZTIN7hnswlib17InnerProductSpaceE
_ZTSN7hnswlib17InnerProductSpaceE
_ZN7hnswlib17InnerProductSpace13get_dist_funcEv
_ZN7hnswlib17InnerProductSpace13get_data_sizeEv
_ZN7hnswlib17InnerProductSpaceD0Ev
```

У C++ нет ни стабильного ABI

Ни стабильной схемы mangling'a

ABI (Application Binary Interface) – соглашение о том, как скомпилированные библиотеки взаимодействуют друг с другом



Решение для hnswlib

С-клей для C++ библиотеки

```
// bindings.h
Index *create_new_index(int32_t spaceName, int32_t dimension);

// bindings.cpp
extern "C" DLLEXPORT Index *create_new_index(
    std::int32_t spaceName,
    std::int32_t dimension
) {
    Index *index;
    try {
        index = new Index(spaceName, dimension);
    } catch (...) {
        index = nullptr;
    }
    return index;
}
```

Решение для hnswlib

С-клей для C++ библиотеки

```
// bindings.h
Index *create_new_index(int32_t spaceName, int32_t dimension);

// bindings.cpp
extern "C" DLLEXPORT Index *create_new_index(
    std::int32_t spaceName,
    std::int32_t dimension
) {
    Index *index;
    try {
        index = new Index(spaceName, dimension);
    } catch (...) {
        index = nullptr;
    }
    return index;
}
```

Решение для hnswlib

С-клей для C++ библиотеки

```
// bindings.h
Index *create_new_index(int32_t spaceName, int32_t dimension);

// bindings.cpp
extern "C" DLLEXPORT Index *create_new_index(
    std::int32_t spaceName,
    std::int32_t dimension
) {
    Index *index;
    try {
        index = new Index(spaceName, dimension);
    } catch (...) {
        index = nullptr;
    }
    return index;
}
```

Решение для hnswlib

С-клей для C++ библиотеки

```
// bindings.h
Index *create_new_index(int32_t spaceName, int32_t dimension);

// bindings.cpp
extern "C" DLLEXPORT Index *create_new_index(
    std::int32_t spaceName,
    std::int32_t dimension
) {
    Index *index;
    try {
        index = new Index(spaceName, dimension);
    } catch (...) {
        index = nullptr;
    }
    return index;
}
```

Приходится собирать такой клей

Но это всё ещё лучше JNI-клея

```
$ strings libhnswnlib_panama.so | rg create_new_index  
create_new_index  
create_new_index
```

Приходится собирать такой клей

Но это всё ещё лучше JNI-клея



Нюансы вне Hello World'ов

Кроме скаляров, иногда хочется передавать в разные стороны и указатели

```
Index *create_new_index(int32_t spaceName, int32_t dimension);

int32_t get_index_length(Index *index);

int32_t add_item(Index *index, float *item, int32_t normalized, int32_t label);

int32_t knn_query(
    Index *index,
    float *input,
    int32_t normalized,
    int32_t k,
    int32_t *indices /* output */,
    float *coefficients /* output */
);
```

Нюансы вне Hello World'ов

Кроме скаляров, иногда хочется передавать в разные стороны и указатели

```
Index *create_new_index(int32_t spaceName, int32_t dimension);

int32_t get_index_length(Index *index);

int32_t add_item(Index *index, float *item, int32_t normalized, int32_t label);

int32_t knn_query(
    Index *index,
    float *input,
    int32_t normalized,
    int32_t k,
    int32_t *indices /* output */,
    float *coefficients /* output */
);
```

Нюансы вне Hello World'ов

Кроме скаляров, иногда хочется передавать в разные стороны и указатели

```
Index *create_new_index(int32_t spaceName, int32_t dimension);

int32_t get_index_length(Index *index);

int32_t add_item(Index *index, float *item, int32_t normalized, int32_t label);

int32_t knn_query(
    Index *index,
    float *input,
    int32_t normalized,
    int32_t k,
    int32_t *indices /* output */,
    float *coefficients /* output */
);
```

Нюансы вне Hello World'ов

Кроме скаляров, иногда хочется передавать в разные стороны и указатели

```
Index *create_new_index(int32_t spaceName, int32_t dimension);
```

```
int32_t get_index_length(Index *index);
```

```
int32_t add_item(Index *index, float *item, int32_t normalized, int32_t label);
```

```
int32_t knn_query(  
    Index *index,  
    float *input,  
    int32_t normalized,  
    int32_t k,  
    int32_t *indices /* output */,  
    float *coefficients /* output */  
);
```

Нюансы вне Hello World'ов

Кроме скаляров, иногда хочется передавать в разные стороны и указатели

```
Index *create_new_index(int32_t spaceName, int32_t dimension);

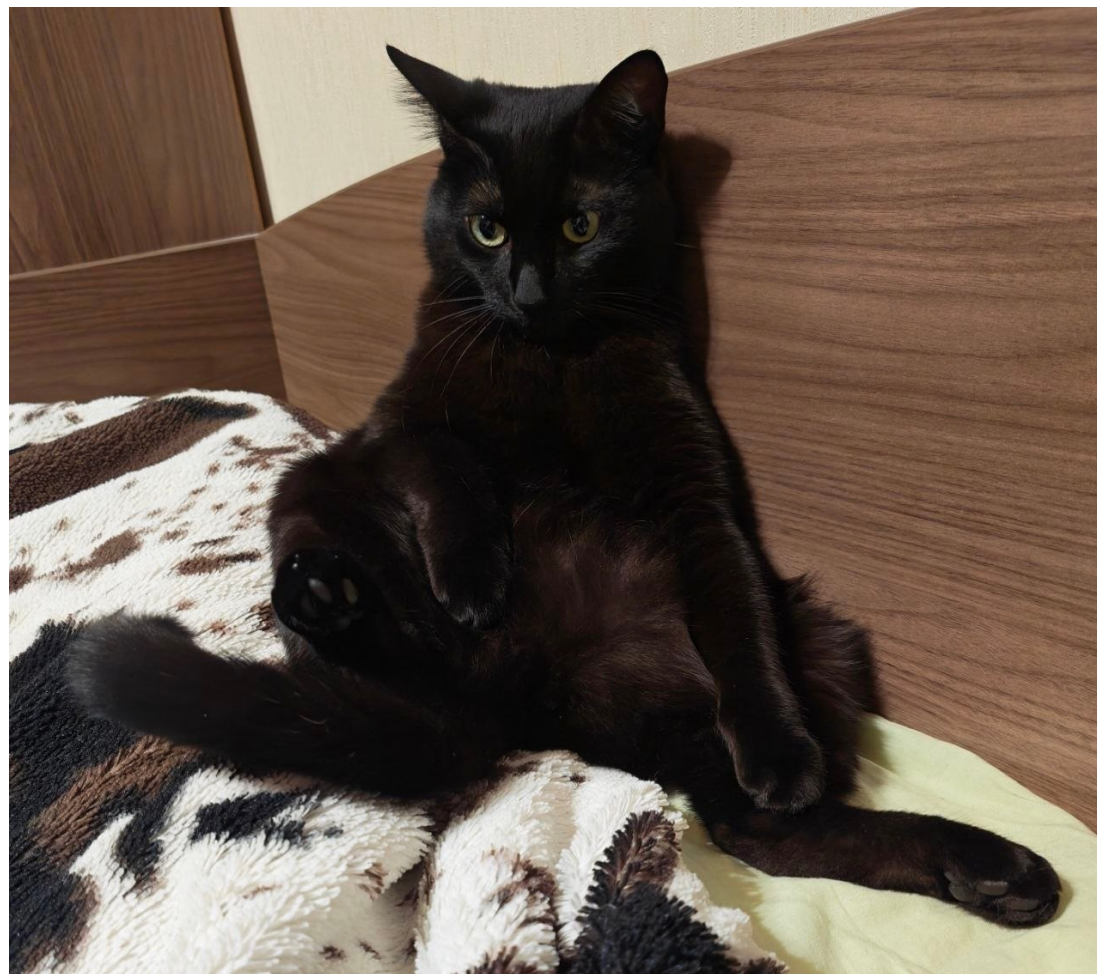
int32_t get_index_length(Index *index);

int32_t add_item(Index *index, float *item, int32_t normalized, int32_t label);

int32_t knn_query(
    Index *index,
    float *input,
    int32_t normalized,
    int32_t k,
    int32_t *indices /* output */,
    float *coefficients /* output */
);
```

Нюансы вне Hello World'ов

Писать руками биндинги – муторное и error prone'истое занятие



Поэтому есть средства автоматизации

А именно Jextract

```
$ jextract \  
  --target-package 'org.nmslib.hnswlib.bindings' \  
  --library /usr/local/lib/libhnswlib_panama.so \  
  ./src/include/bindings.h
```

<https://jdk.java.net/jextract/>

Поэтому есть средства автоматизации

А именно Jextract

```
$ jextract \  
  --target-package 'org.nmslib.hnswlib.bindings' \  
  --library /usr/local/lib/libhnswlib_panama.so \  
  ./src/include/bindings.h
```

<https://jdk.java.net/jextract/>

Поэтому есть средства автоматизации

А именно Jextract

```
$ jextract \  
  --target-package 'org.nmslib.hnswlib.bindings' \  
  --library /usr/local/lib/libhnswlib_panama.so \  
  ./src/include/bindings.h
```

<https://jdk.java.net/jextract/>

Поэтому есть средства автоматизации

А именно Jextract

```
$ jextract \  
  --target-package 'org.nmslib.hnswlib.bindings' \  
  --library /usr/local/lib/libhnswlib_panama.so \  
  ./src/include/bindings.h
```

<https://jdk.java.net/jextract/>

Генерируется Java-код биндингов

Похожий на то, что мы писали руками

```
private static class create_new_index {  
    public static final FunctionDescriptor DESC = FunctionDescriptor.of(  
        bindings_h.C_POINTER,  
        bindings_h.C_INT,  
        bindings_h.C_INT  
    );  
  
    public static final MemorySegment ADDR  
        = SYMBOL_LOOKUP.findOrThrow( "create_new_index" );  
  
    public static final MethodHandle HANDLE  
        = Linker.nativeLinker().downcallHandle(ADDR, DESC);  
}
```

Генерируется Java-код биндингов

Похожий на то, что мы писали руками

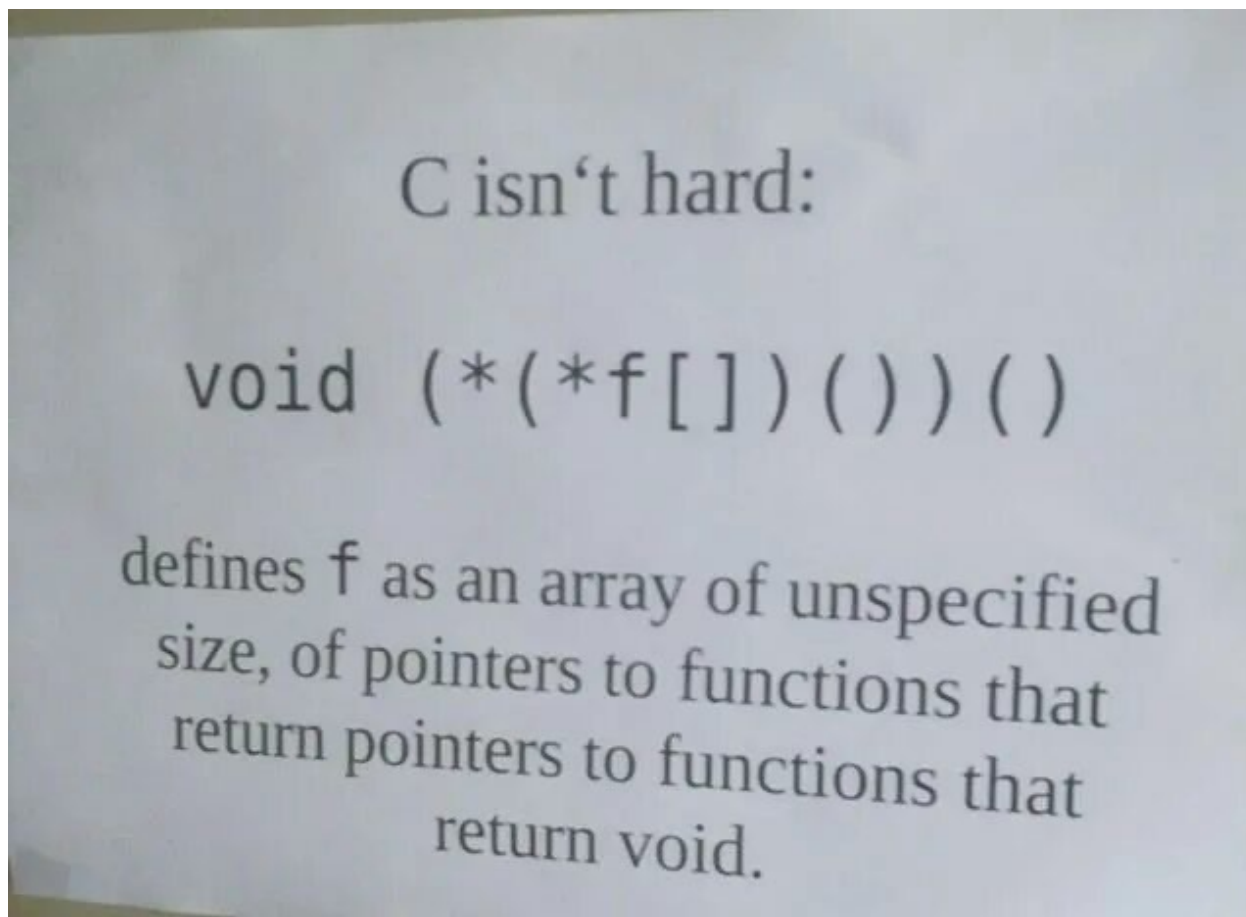
```
public static MemorySegment create_new_index(  
    int spaceName, int dimension  
) {  
    var mh$ = create_new_index.HANDLE;  
    try {  
        if (TRACE_DOWNCALLS) {  
            traceDowncall("create_new_index", spaceName, dimension);  
        }  
        return (MemorySegment)mh$.invokeExact(spaceName, dimension);  
    } catch (Error | RuntimeException ex) {  
        throw ex;  
    } catch (Throwable ex$) {  
        throw new AssertionError("should not reach here", ex$);  
    }  
}
```

Так что там с указателями?

Именно за них мы так любим C

Так что там с указателями?

Именно за них мы так любим C



Доступ к области
памяти в FFM API
предоставляется через
объекты

MemorySegment



Полученный извне MemorySegment

По смыслу – Index*

// bindings.h

```
Index *create_new_index(int32_t spaceName, int32_t dimension);  
int32_t clear_index(Index *index);
```

// bindings_h.java

```
public static MemorySegment create_new_index(int spaceName, int dimension) { ... }  
public static int clear_index(MemorySegment index) { ... }
```

// Код высокоуровневой обёртки

// Например, в конструкторе

```
index = bindings_h.create_new_index(spaceName.ordinal(), dimension)
```

...

// Например, в close()

```
bindings_h.clear_index(index);
```

Полученный извне MemorySegment

По смыслу – Index*

```
// bindings.h
```

```
Index *create_new_index(int32_t spaceName, int32_t dimension);  
int32_t clear_index(Index *index);
```

```
// bindings_h.java
```

```
public static MemorySegment create_new_index(int spaceName, int dimension) { ... }  
public static int clear_index(MemorySegment index) { ... }
```

```
// Код высокоуровневой обёртки
```

```
// Например, в конструкторе
```

```
index = bindings_h.create_new_index(spaceName.ordinal(), dimension)
```

```
...
```

```
// Например, в close()
```

```
bindings_h.clear_index(index);
```

Полученный извне MemorySegment

По смыслу – Index*

```
// bindings.h
```

```
Index *create_new_index(int32_t spaceName, int32_t dimension);  
int32_t clear_index(Index *index);
```

```
// bindings_h.java
```

```
public static MemorySegment create_new_index(int spaceName, int dimension) { ... }  
public static int clear_index(MemorySegment index) { ... }
```

```
// Код высокоуровневой обёртки
```

```
// Например, в конструкторе
```

```
index = bindings_h.create_new_index(spaceName.ordinal(), dimension)
```

```
...
```

```
// Например, в close()
```

```
bindings_h.clear_index(index);
```

Полученный извне MemorySegment

По смыслу – Index*

```
// bindings.h
```

```
Index *create_new_index(int32_t spaceName, int32_t dimension);  
int32_t clear_index(Index *index);
```

```
// bindings_h.java
```

```
public static MemorySegment create_new_index(int spaceName, int dimension) { ... }  
public static int clear_index(MemorySegment index) { ... }
```

```
// Код высокоуровневой обёртки
```

```
// Например, в конструкторе
```

```
index = bindings_h.create_new_index(spaceName.ordinal(), dimension)
```

```
...
```

```
// Например, в close()
```

```
bindings_h.clear_index(index);
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);

// bindings_h.java
public static int get_data(
    MemorySegment index, int id, MemorySegment vector, int dim
) { ... }

// Код высокоуровневой обёртки
try (var arena = Arena.ofConfined()) {
    var floatArray = arena.allocate(
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)
    );
    bindings_h.get_data(index, id, floatArray, dimension);
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h  
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);
```

```
// bindings_h.java  
public static int get_data(  
    MemorySegment index, int id, MemorySegment vector, int dim  
) { ... }
```

```
// Код высокоуровневой обёртки  
try (var arena = Arena.ofConfined()) {  
    var floatArray = arena.allocate(  
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)  
    );  
    bindings_h.get_data(index, id, floatArray, dimension);  
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);  
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h  
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);
```

```
// bindings_h.java  
public static int get_data(  
    MemorySegment index, int id, MemorySegment vector, int dim  
) { ... }
```

```
// Код высокоуровневой обёртки  
try (var arena = Arena.ofConfined()) {  
    var floatArray = arena.allocate(  
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)  
    );  
    bindings_h.get_data(index, id, floatArray, dimension);  
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);  
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);

// bindings_h.java
public static int get_data(
    MemorySegment index, int id, MemorySegment vector, int dim
) { ... }

// Код высокоуровневой обёртки
try (var arena = Arena.ofConfined()) {
    var floatArray = arena.allocate(
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)
    );
    bindings_h.get_data(index, id, floatArray, dimension);
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);

// bindings_h.java
public static int get_data(
    MemorySegment index, int id, MemorySegment vector, int dim
) { ... }

// Код высокоуровневой обёртки
try (var arena = Arena.ofConfined()) {
    var floatArray = arena.allocate(
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)
    );
    bindings_h.get_data(index, id, floatArray, dimension);
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);

// bindings_h.java
public static int get_data(
    MemorySegment index, int id, MemorySegment vector, int dim
) { ... }

// Код высокоуровневой обёртки
try (var arena = Arena.ofConfined()) {
    var floatArray = arena.allocate(
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)
    );
    bindings_h.get_data(index, id, floatArray, dimension);
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);
}
```

А если нужно самому выделить память?

На помощь приходит Arena

```
// bindings.h
int32_t get_data(Index *index, int32_t id, float *vector, int32_t dim);

// bindings_h.java
public static int get_data(
    MemorySegment index, int id, MemorySegment vector, int dim
) { ... }

// Код высокоуровневой обёртки
try (var arena = Arena.ofConfined()) {
    var floatArray = arena.allocate(
        MemoryLayout.sequenceLayout(dimension, ValueLayout.JAVA_FLOAT)
    );
    bindings_h.get_data(index, id, floatArray, dimension);
    return floatArray.toArray(ValueLayout.JAVA_FLOAT);
}
```

Jextract делает для нас
Java-клеи,
а мы — пишем

высокоуровневые обёртки



Нюансы Jextract

Нетривиальные определения типов

```
private static final Map<String, MemoryLayout> CANONICAL_LAYOUTS
    = Linker.nativeLinker().canonicalLayouts();

public static final ValueLayout.OfBoolean C_BOOL
    = (ValueLayout.OfBoolean) CANONICAL_LAYOUTS.get("bool");

...

public static final ValueLayout.OfLong C_LONG_LONG
    = (ValueLayout.OfLong) CANONICAL_LAYOUTS.get("long long");

...

public static final AddressLayout C_POINTER
    = ((AddressLayout) CANONICAL_LAYOUTS.get("void*"))
        .withTargetLayout(sequenceLayout(java.lang.Long.MAX_VALUE, C_CHAR));
```

Нюансы Jextract

Нетривиальные определения типов

```
private static final Map<String, MemoryLayout> CANONICAL_LAYOUTS
    = Linker.nativeLinker().canonicalLayouts();

public static final ValueLayout.OfBoolean C_BOOL
    = (ValueLayout.OfBoolean) CANONICAL_LAYOUTS.get("bool");

...
public static final ValueLayout.OfLong C_LONG_LONG
    = (ValueLayout.OfLong) CANONICAL_LAYOUTS.get("long long");

...
public static final AddressLayout C_POINTER
    = ((AddressLayout) CANONICAL_LAYOUTS.get("void*"))
        .withTargetLayout(sequenceLayout(java.lang.Long.MAX_VALUE, C_CHAR));
```

Нюансы Jextract

Нетривиальные определения типов

```
private static final Map<String, MemoryLayout> CANONICAL_LAYOUTS
    = Linker.nativeLinker().canonicalLayouts();

public static final ValueLayout.OfBoolean C_BOOL
    = (ValueLayout.OfBoolean) CANONICAL_LAYOUTS.get("bool");

...
public static final ValueLayout.OfLong C_LONG_LONG
    = (ValueLayout.OfLong) CANONICAL_LAYOUTS.get("long long");

...
public static final AddressLayout C_POINTER
    = ((AddressLayout) CANONICAL_LAYOUTS.get("void*"))
        .withTargetLayout(sequenceLayout(java.lang.Long.MAX_VALUE, C_CHAR));
```

Нюансы Jextract

Нетривиальные определения типов

```
private static final Map<String, MemoryLayout> CANONICAL_LAYOUTS
    = Linker.nativeLinker().canonicalLayouts();

public static final ValueLayout.OfBoolean C_BOOL
    = (ValueLayout.OfBoolean) CANONICAL_LAYOUTS.get("bool");

...
public static final ValueLayout.OfLong C_LONG_LONG
    = (ValueLayout.OfLong) CANONICAL_LAYOUTS.get("long long");

...
public static final AddressLayout C_POINTER
    = ((AddressLayout) CANONICAL_LAYOUTS.get("void*"))
        .withTargetLayout(sequenceLayout(java.lang.Long.MAX_VALUE, C_CHAR));
```

Нюансы Jextract

Нетривиальные определения типов

```
private static final Map<String, MemoryLayout> CANONICAL_LAYOUTS
    = Linker.nativeLinker().canonicalLayouts();

public static final ValueLayout.OfBoolean C_BOOL
    = (ValueLayout.OfBoolean) CANONICAL_LAYOUTS.get("bool");

...
public static final ValueLayout.OfLong C_LONG_LONG
    = (ValueLayout.OfLong) CANONICAL_LAYOUTS.get("long long");

...
public static final AddressLayout C_POINTER
    = ((AddressLayout) CANONICAL_LAYOUTS.get("void*"))
        .withTargetLayout(sequenceLayout(java.lang.Long.MAX_VALUE, C_CHAR));
```

Нюансы Jextract

Jextract не умеет расставлять `Linker.Option`'ы по настройкам пользователя

Linker options. It is currently not possible to tell jextract to use linker options to link a particular function. Code will have to be edited manually to add them.
(for instance, if the function sets `errno`, the `Linker.Option.captureCallState` option has to be added manually).

<https://github.com/openjdk/jextract/blob/master/doc/GUIDE.md#unsupported-features>

Нюансы Jextract

Другие ограничения

Некоторые C-типы >64 бит

...

Функцие-подобные макросы

...

Битовые поля

...

Оpaque-типы

...

Атрибуты, управляющие порядком полей

...

YOUR

ONLY

Limit is

Jextract
unsupported
features

Jextract автоматизирует
получение биндингов,
но разработчик всё ещё
должен понимать,
что под капотом



05

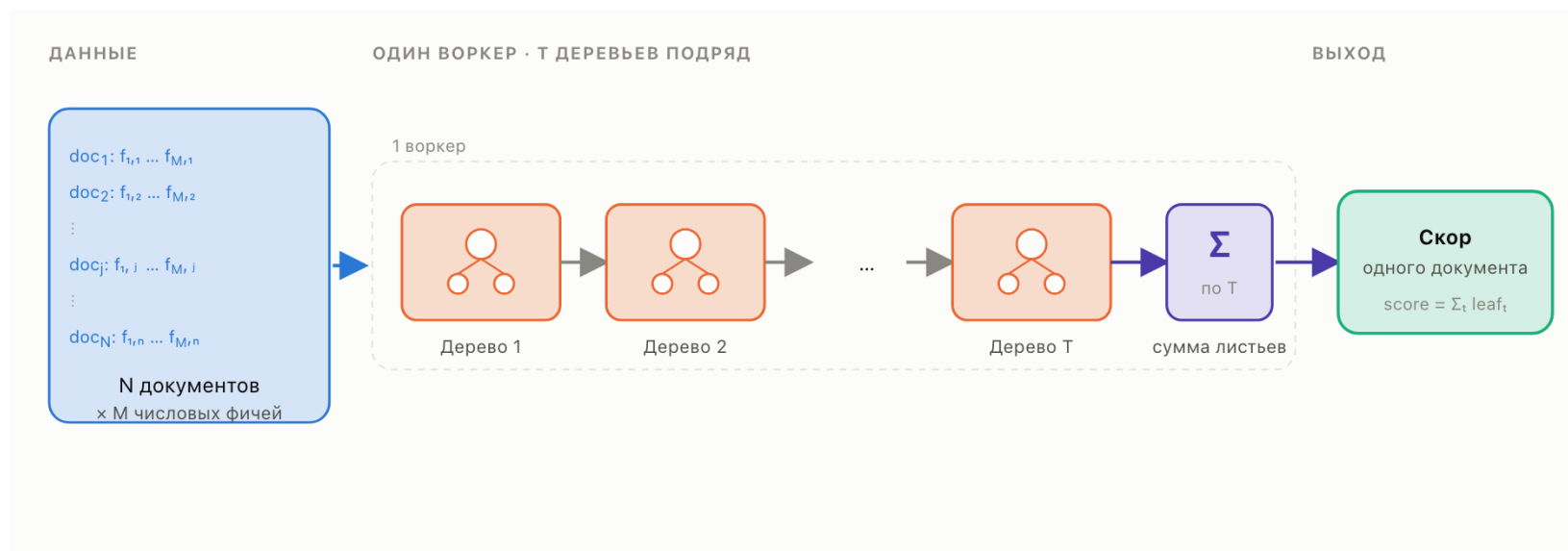


Арены & Биндинги к CatBoost

Что есть CatBoost?

Нативная библиотека для градиентного бустинга

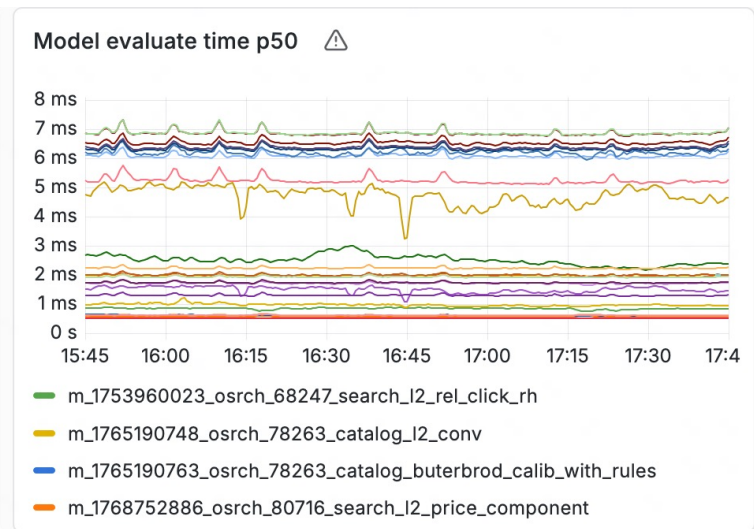
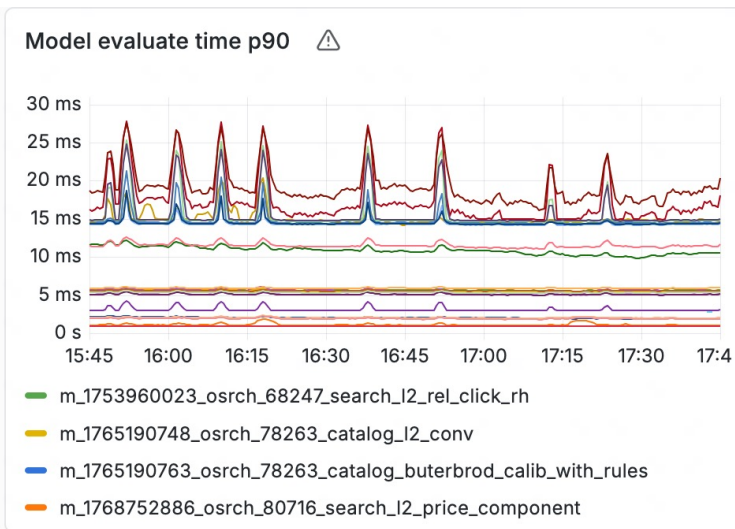
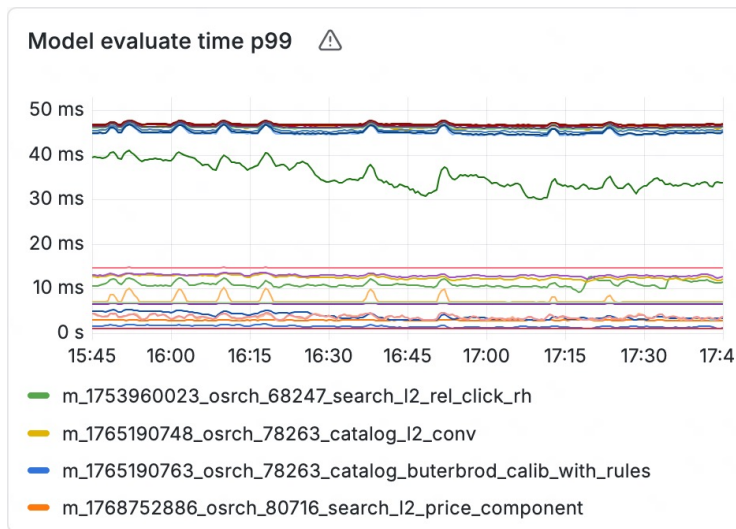
- Используется нами для ранжирования товаров
- Оперирует деревьями решений
- Умеет работать с прямым (строчки-документы) и транспонированным (колонки-фичи) данными



<https://catboost.ai/>

Расчёт CatBoost-моделей – это дорого

Для нас – критично дорого



CatBoost предоставляет стандартные биндинги

Поверх JNI, синхронные

Home » ai.catboost » catboost-prediction



CatBoost Model Applier

Java module to apply CatBoost models

Overview

Versions (32)

Used By (5)

Badges

Books (5)

LICENSE

Apache 2.0

TAGS

ai

RANKING

#45653 in MvnRepository

HOMEPAGE

<https://catboost.ai> [Inspect URL](#)

LINKS



Latest Versions

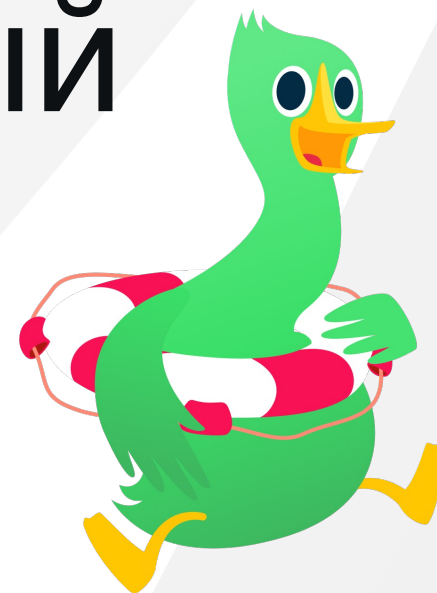
32 versions →

Central (32)

	VERSION	VULNERABILITIES	USAGES	DATE
1.2.x	1.2.10		0	Feb 18, 2026
	1.2.8		0	Apr 13, 2025
	1.2.7		1	Sep 07, 2024

<https://mvnrepository.com/artifact/ai.catboost/catboost-prediction>

Для изменений нужно
трогать апстрим/форк и
пересобирать нативный
клей



Как мы теперь умеем решать эту проблему?

С помощью FFM API!

Как мы теперь умеем решать эту проблему?

С помощью FFM API!

Threads	Backend	Doc count	Score			
1	JNI	100	105.505	± 1.158	us/op	
1	PANAMA	100	93.943	± 7.489	us/op	
1	JNI	1000	903.652	± 1.961	us/op	
1	PANAMA	1000	896.479	± 40.342	us/op	
16	JNI	100	269.416	± 92.590	us/op	
16	PANAMA	100	250.234	± 35.095	us/op	
16	JNI	1000	2346.441	± 407.353	us/op	
16	PANAMA	1000	2277.543	± 428.600	us/op	

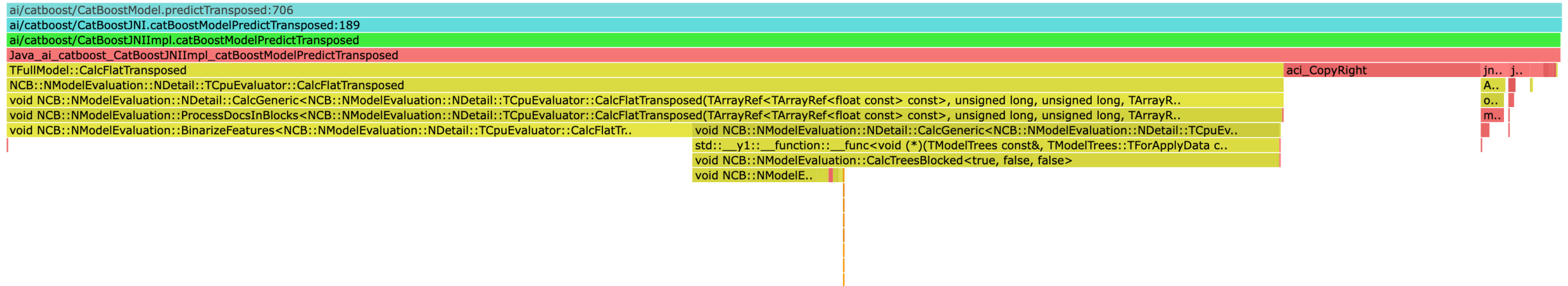
Как мы теперь умеем решать эту проблему?

С помощью FFM API!

Backend	Doc count	Score
JNI	1000	1251.4 $\pm$ 9.5 us/op
PANAMA	1000	1231.1 $\pm$ 17.5 us/op

Почему так?

Потому что надо смотреть на профили!



В FFM API нам также надо
копировать данные, но
можно **делать это умнее**



Арены

Какие они есть?

Арена	Освобождение памяти	Доступ к сегментам из других потоков	Можно ли вызывать <code>close()</code> ?	Зачем?
<code>Arena.global()</code>	С завершением JVM	Да	Нет	Вечноживущие объекты
<code>Arena.ofAuto()</code>	В момент GC	Да	Нет	Жизненный цикл нетривиален
<code>Arena.ofConfined()</code>	При закрытии	Нет	Да	Короткоживущие локальные аллокации
<code>Arena.ofShared()</code>	При закрытии	Да	Да	Ограниченные, но шарящиеся аллокации

Confined нам подходит идеально

Ограниченный скоуп

```
try (var arena = Arena.ofConfined()) {  
    var numericFeatureMatrixRows = arena  
        .allocate(ValueLayout.ADDRESS, featureCount);  
  
    ...  
  
    var predictions = arena  
        .allocate(ValueLayout.JAVA_DOUBLE, predictionsSize);  
  
    calcModelPredictionFlatTransposed(  
        size,  
        numericFeatureMatrixRows, featureCount,  
        predictions, predictionsSize  
    );  
    return unwrapPredictions(Math.toIntExact(documentCount), predictions);  
}
```

Confined нам подходит идеально

Ограниченный скоуп

```
try (var arena = Arena.ofConfined()) {  
    var numericFeatureMatrixRows = arena  
        .allocate(ValueLayout.ADDRESS, featureCount);  
  
    ...  
  
    var predictions = arena  
        .allocate(ValueLayout.JAVA_DOUBLE, predictionsSize);  
  
    calcModelPredictionFlatTransposed(  
        size,  
        numericFeatureMatrixRows, featureCount,  
        predictions, predictionsSize  
    );  
    return unwrapPredictions(Math.toIntExact(documentCount), predictions);  
}
```

Confined нам подходит идеально

Ограниченный скоуп

```
try (var arena = Arena.ofConfined()) {  
    var numericFeatureMatrixRows = arena  
        .allocate(ValueLayout.ADDRESS, featureCount);  
  
    ...  
  
    var predictions = arena  
        .allocate(ValueLayout.JAVA_DOUBLE, predictionsSize);  
  
    calcModelPredictionFlatTransposed(  
        size,  
        numericFeatureMatrixRows, featureCount,  
        predictions, predictionsSize  
    );  
    return unwrapPredictions(Math.toIntExact(documentCount), predictions);  
}
```

Confined нам подходит идеально

Ограниченный скоуп

```
try (var arena = Arena.ofConfined()) {  
    var numericFeatureMatrixRows = arena  
        .allocate(ValueLayout.ADDRESS, featureCount);  
  
    ...  
  
    var predictions = arena  
        .allocate(ValueLayout.JAVA_DOUBLE, predictionsSize);  
  
    calcModelPredictionFlatTransposed(  
        size,  
        numericFeatureMatrixRows, featureCount,  
        predictions, predictionsSize  
    );  
    return unwrapPredictions(Math.toIntExact(documentCount), predictions);  
}
```

Confined нам подходит идеально

Ограниченный скоуп

```
try (var arena = Arena.ofConfined()) {  
    var numericFeatureMatrixRows = arena  
        .allocate(ValueLayout.ADDRESS, featureCount);  
  
    ...  
  
    var predictions = arena  
        .allocate(ValueLayout.JAVA_DOUBLE, predictionsSize);  
  
    calcModelPredictionFlatTransposed(  
        size,  
        numericFeatureMatrixRows, featureCount,  
        predictions, predictionsSize  
    );  
    return unwrapPredictions(Math.toIntExact(documentCount), predictions);  
}
```

Slicing Allocator

Можно сразу выделить память нужного размера и дёшево нарезать её

```
var rowLayout = MemoryLayout
    .sequenceLayout(documentCount, ValueLayout.JAVA_FLOAT);

var rowsAllocator = SegmentAllocator.slicingAllocator(
    arena.allocate(rowLayout, featureCount)
);

for (var index = 0; index < featureCount; index++) {
    var inMemoryRow = numericFeaturesTransposed[index];
    var row = rowsAllocator
        .allocateFrom(ValueLayout.JAVA_FLOAT, inMemoryRow);
    numericFeatureMatrixRows
        .setAtIndex(ValueLayout.ADDRESS, index, row);
}
```

Slicing Allocator

Можно сразу выделить память нужного размера и дёшево нарезать её

```
var rowLayout = MemoryLayout
    .sequenceLayout(documentCount, ValueLayout.JAVA_FLOAT);

var rowsAllocator = SegmentAllocator.slicingAllocator(
    arena.allocate(rowLayout, featureCount)
);

for (var index = 0; index < featureCount; index++) {
    var inMemoryRow = numericFeaturesTransposed[index];
    var row = rowsAllocator
        .allocateFrom(ValueLayout.JAVA_FLOAT, inMemoryRow);
    numericFeatureMatrixRows
        .setAtIndex(ValueLayout.ADDRESS, index, row);
}
```

Slicing Allocator

Можно сразу выделить память нужного размера и дешево нарезать её

```
var rowLayout = MemoryLayout
    .sequenceLayout(documentCount, ValueLayout.JAVA_FLOAT);

var rowsAllocator = SegmentAllocator.slicingAllocator(
    arena.allocate(rowLayout, featureCount)
);

for (var index = 0; index < featureCount; index++) {
    var inMemoryRow = numericFeaturesTransposed[index];
    var row = rowsAllocator
        .allocateFrom(ValueLayout.JAVA_FLOAT, inMemoryRow);
    numericFeatureMatrixRows
        .setAtIndex(ValueLayout.ADDRESS, index, row);
}
```

Slicing Allocator

Можно сразу выделить память нужного размера и дёшево нарезать её

```
var rowLayout = MemoryLayout
    .sequenceLayout(documentCount, ValueLayout.JAVA_FLOAT);

var rowsAllocator = SegmentAllocator.slicingAllocator(
    arena.allocate(rowLayout, featureCount)
);

for (var index = 0; index < featureCount; index++) {
    var inMemoryRow = numericFeaturesTransposed[index];
    var row = rowsAllocator
        .allocateFrom(ValueLayout.JAVA_FLOAT, inMemoryRow);
    numericFeatureMatrixRows
        .setAtIndex(ValueLayout.ADDRESS, index, row);
}
```

Slicing Allocator

Можно сразу выделить память нужного размера и дёшево нарезать её

```
var rowLayout = MemoryLayout
    .sequenceLayout(documentCount, ValueLayout.JAVA_FLOAT);

var rowsAllocator = SegmentAllocator.slicingAllocator(
    arena.allocate(rowLayout, featureCount)
);

for (var index = 0; index < featureCount; index++) {
    var inMemoryRow = numericFeaturesTransposed[index];
    var row = rowsAllocator
        .allocateFrom(ValueLayout.JAVA_FLOAT, inMemoryRow);
    numericFeatureMatrixRows
        .setAtIndex(ValueLayout.ADDRESS, index, row);
}
```

И немножко экономии на спичках

Чёрной магией делаем `uninit`-аллокации

```
var numericFeatureMatrixRows = UninitAllocations  
    .allocateUninit(arena, ValueLayout.ADDRESS, featureCount);
```

...

```
var rowsAllocator = SegmentAllocator.slicingAllocator(  
    UninitAllocations.allocateUninit(arena, rowLayout, featureCount)  
);
```

...

```
var predictions = UninitAllocations  
    .allocateUninit(arena, ValueLayout.JAVA_DOUBLE, predictionsSize);
```

И немножко экономии на спичках

Чёрной магией делаем `uninit`-аллокации

```
var numericFeatureMatrixRows = UninitAllocations  
    .allocateUninit(arena, ValueLayout.ADDRESS, featureCount);
```

...

```
var rowsAllocator = SegmentAllocator.slicingAllocator(  
    UninitAllocations.allocateUninit(arena, rowLayout, featureCount)  
);
```

...

```
var predictions = UninitAllocations  
    .allocateUninit(arena, ValueLayout.JAVA_DOUBLE, predictionsSize);
```

И немножко экономии на спичках

Чёрной магией делаем uninit-аллокации

```
var numericFeatureMatrixRows = UninitAllocations  
    .allocateUninit(arena, ValueLayout.ADDRESS, featu
```

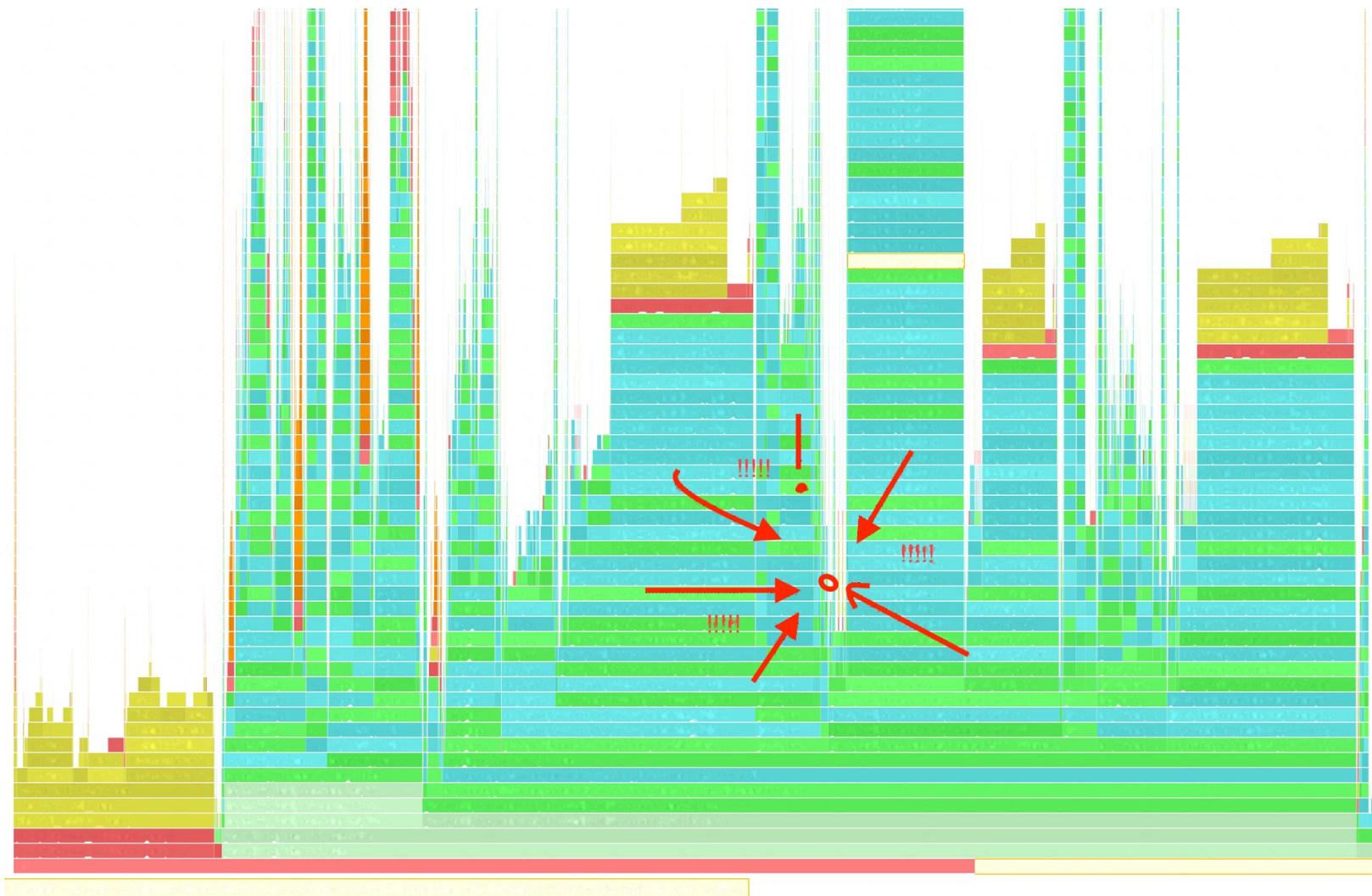
Чтение
неинициализированной
памяти – это UB

```
var predictions = UninitAllocations  
    .allocateUninit(arena, ValueLayout.JAVA_DOUBLE, p
```



**Beware
of moving
parts**

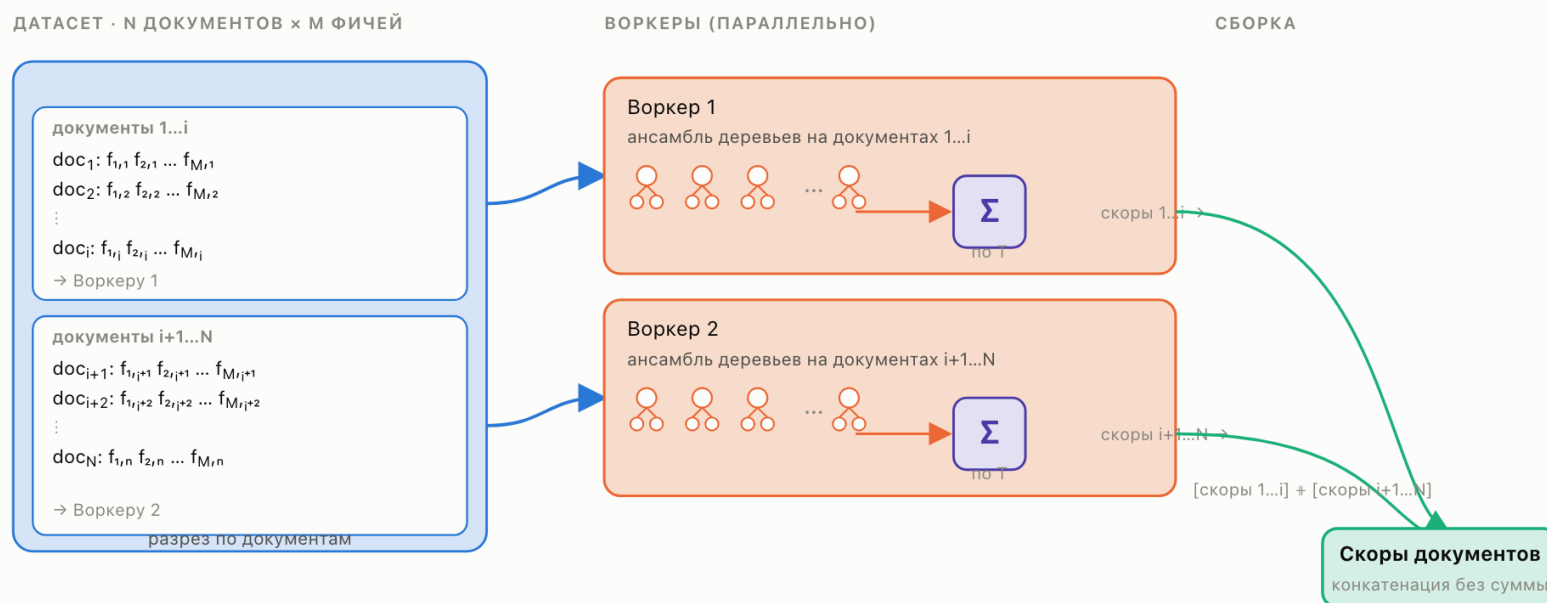
Как я ищу места для оптимизации



Параллелизм уровня документов

Задача красиво разбивается на подзадачи

- Можно *параллельно* запускать CatBoost для диапазонов документов
- Значения фичей мы в любом случае перекладываем
- Надо склеивать результаты подзадач, либо давать им писать в общий массив



Подзадачи независимы,
поэтому каждая может
использовать
свою confined-арену



И вот после оптимизации алгоритмики

Стало красиво (очень)

Backend	Score
JNI	1223.7 $\pm$ 4.8 us/op
PANAMA	1212.2 $\pm$ 24.0 us/op
PANAMA_BLOCK512	608.1 $\pm$ 7.1 us/op
PANAMA_BLOCK256	325.1 $\pm$ 1.6 us/op
PANAMA_BLOCK128	200.6 $\pm$ 1.2 us/op

А причём тут панама?

Мы помним, что ты преподаёшь алгосики

```
res-avx512-default-avgt-PANAMA-sequential_24-b24.json
res-avx512-default-thrpt-JNI-sequential_12-b12.json
res-avx512-default-thrpt-JNI-sequential_24-b24.json
res-avx512-default-thrpt-PANAMA_BLOCK128-parallel-b12.json
res-avx512-default-thrpt-PANAMA_BLOCK128-parallel-b24.json
res-avx512-default-thrpt-PANAMA_BLOCK256-parallel-b12.json
res-avx512-default-thrpt-PANAMA_BLOCK256-parallel-b24.json
res-avx512-default-thrpt-PANAMA_BLOCK512-parallel-b12.json
res-avx512-default-thrpt-PANAMA_BLOCK512-parallel-b24.json
res-avx512-default-thrpt-PANAMA-sequential_12-b12.json
res-avx512-default-thrpt-PANAMA-sequential_24-b24.json
res-before-avgt-JNI-BIG-t12.json
res-before-avgt-JNI-BIG-t1.json
res-before-avgt-JNI-BIG-t24.json
res-before-avgt-JNI-BIG-t48.json
res-before-avgt-JNI-BIG-t96.json
res-before-avgt-PANAMA-BIG-t12.json
res-before-avgt-PANAMA-BIG-t1.json
res-before-avgt-PANAMA-BIG-t24.json
res-before-avgt-PANAMA-BIG-t48.json
res-before-avgt-PANAMA-BIG-t96.json
res-before-avgt-PANAMA_BLOCK128-BIG-t12.json
res-before-avgt-PANAMA_BLOCK128-BIG-t1.json
res-before-avgt-PANAMA_BLOCK128-BIG-t24.json
res-before-avgt-PANAMA_BLOCK128-BIG-t48.json
res-before-avgt-PANAMA_BLOCK128-BIG-t96.json
res-before-avgt-PANAMA_BLOCK128-SMALL-t12.json
res-before-avgt-PANAMA_BLOCK128-SMALL-t1.json
res-before-avgt-PANAMA_BLOCK128-SMALL-t24.json
res-before-avgt-PANAMA_BLOCK128-SMALL-t48.json
res-before-avgt-PANAMA_BLOCK128-SMALL-t96.json
res-before-avgt-PANAMA_BLOCK256-BIG-t12.json
res-before-avgt-PANAMA_BLOCK256-BIG-t1.json
res-before-avgt-PANAMA_BLOCK256-BIG-t24.json
res-before-avgt-PANAMA_BLOCK256-BIG-t48.json
res-before-avgt-PANAMA_BLOCK256-BIG-t96.json
res-before-avgt-PANAMA_BLOCK256-SMALL-t12.json
res-before-avgt-PANAMA_BLOCK256-SMALL-t1.json
res-before-avgt-PANAMA_BLOCK256-SMALL-t24.json
res-before-avgt-PANAMA_BLOCK256-SMALL-t48.json
res-before-avgt-PANAMA_BLOCK256-SMALL-t96.json
res-before-avgt-PANAMA_BLOCK512-BIG-t12.json
res-before-avgt-PANAMA_BLOCK512-BIG-t1.json
res-before-avgt-PANAMA_BLOCK512-BIG-t24.json
res-before-avgt-PANAMA_BLOCK512-BIG-t48.json
res-before-avgt-PANAMA_BLOCK512-BIG-t96.json
res-before-avgt-PANAMA_BLOCK512-SMALL-t12.json
res-before-avgt-PANAMA_BLOCK512-SMALL-t1.json
res-before-avgt-PANAMA_BLOCK512-SMALL-t24.json
res-before-avgt-PANAMA_BLOCK512-SMALL-t48.json
res-before-avgt-PANAMA_BLOCK512-SMALL-t96.json
res-before-thrpt-JNI-BIG-t12.json
res-before-thrpt-JNI-BIG-t1.json
res-before-thrpt-JNI-BIG-t24.json
res-before-thrpt-JNI-BIG-t48.json
res-before-thrpt-JNI-BIG-t96.json
```

```
res-t1
results.csv
run-align-percore.sh
run-align-user-percore.sh
run-asyncprof-alignu-collapsed.sh
run-asyncprof-alignu.sh
run-asyncprof-alignu-v2.sh
run-asyncprof-avx512.sh
run-asyncprof.log
run-asyncprof-pad16.sh
run-avx512-final.log
run-avx512-final.sh
run-avx512.log
run-avx512-percore.sh
run-blocksize-sweep.sh
run-c2c.sh
run-compare-highthr.sh
run-compcore.sh
run-docprof.sh
run-opt2.sh
run-pad16
run-percore.sh
run-poolcmp.sh
run-prof-cpu.sh
run-sharedfeats.sh
run-simd-cmp.sh
run-staged-percore.log
run-staged-percore.sh
run-thrpt.sh
run-t1.log
run-t1-percore.sh
run-verify.sh
sharedfeats.log
simd-catboost-panama-aligned-percore-B128-b12.txt
simd-catboost-panama-baseline-percore-B128-b12.txt
src
staged-percore2.log
staged-percore.log
stat-PANAMA_BLOCK64-parallel-b12.err
stat-PANAMA_BLOCK64-parallel-b12.log
stat-PANAMA-sequential_12-b12.err
stat-PANAMA-sequential_12-b12.log
thrpt-JNI-t12.json
thrpt-JNI-t1.json
thrpt-JNI-t24.json
thrpt-JNI-t48.json
thrpt-JNI-t96.json
thrpt.log
thrpt-PANAMA_BLOCK256-t12.json
thrpt-PANAMA_BLOCK256-t1.json
thrpt-PANAMA_BLOCK256-t24.json
thrpt-PANAMA_BLOCK256-t48.json
thrpt-PANAMA_BLOCK256-t96.json
verify-avgt.json
verify.log
verify-thrpt.json
```

opt-log

00-стенд.md

02-baseline.md

03-summary.md

04-throughput.md

05-compare-before-after.md

06-per-core-comparison.md

07-staged-trees-comparison.md

08-parallel-profiling-root-cause.md

09-sharedfeats-comparison.md

10-blocksize-sweep.md

11-opt2-arena-reuse.md

12-perf-c2c-stat.md

13-docprof-cpu-work.md

14-per-thread-handles-experiment.md

15-data-alignment-experiment.md

16-aligned-via-layout-and-frame-profile.md

17-aligned-size-pad16.md

18-avx512-binary.md

19-avx512-proper-cpu-profiles.md

opt-log-opencode

00-стенд.md

01-first-benchmark.md

02-baseline.md

03-summary.md

measure-idle-cpu.sh

А причём тут панама?

FJP – мегаудобен

```
@RequiredArgsConstructor(access = AccessLevel.PRIVATE)
private final class EvaluationTask extends RecursiveAction {
    private final float[][] features;
    private final int docs;
    private final int blockSize;

    private final float[] receiver;
    private final int start;
    private final int end;

    @Override
    protected void compute() {
        ...
    }
}
```

И немножко экономии на свичках

FJP – мегаудобен

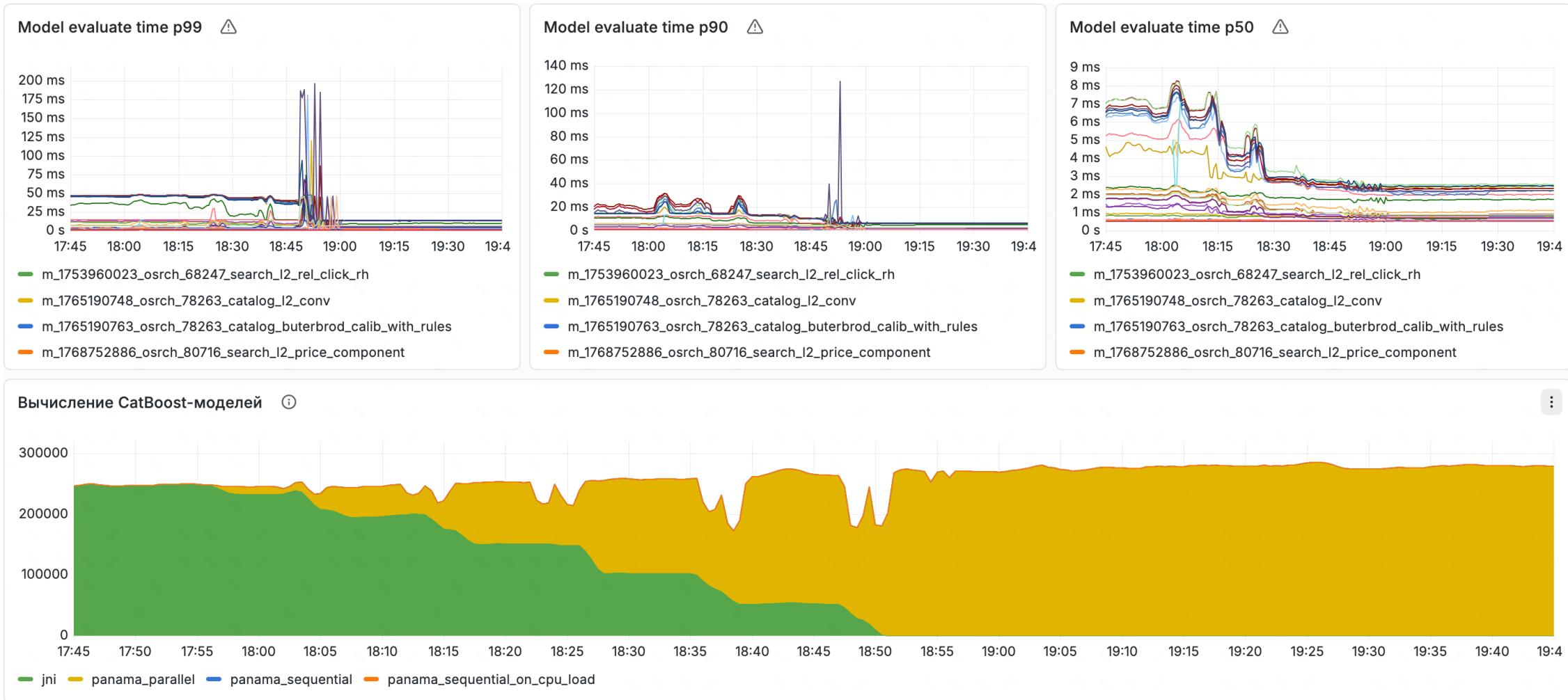
```
@Override
protected void compute() {
    if (end - start < blockSize) {
        evalTransposedInto(features, docs, receiver, start, end);
    } else {
        var mid = (start + end) >>> 1;
        invokeAll(
            new EvaluationTask(features, docs, blockSize, receiver, start, mid),
            new EvaluationTask(features, docs, blockSize, receiver, mid, end)
        );
    }
}
```

Бенчмарки ничто

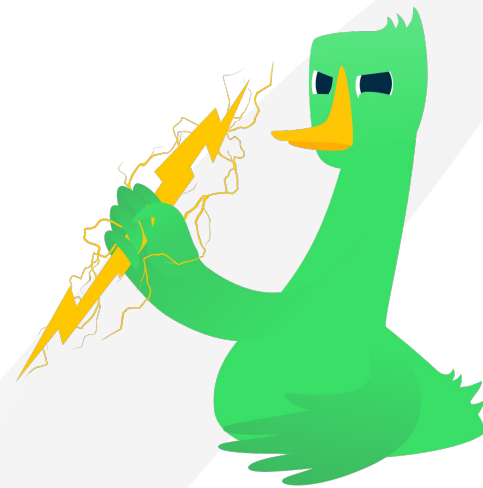
Покажите нам prod

Бенчмарки ничто

Покажите нам prod



FFM API позволил легко
проверять перформанс
гипотезы и строить
эффективное
параллельное
вычисление **без строчки C**



05



Panama in the wild

zstd-jni

Недавно стали использовать FFM API



<https://github.com/luben/zstd-ini#foreign-function--memory-api>

Lucene

Давно использует FFM API для mmap

This class will use the modern MemorySegment API available since Java 21 which allows to safely unmap previously mmapmed files after closing the IndexInputs. There is no need to enable the "preview feature" of your Java version; it works out of box with some compilation tricks. For more information about the foreign memory API read documentation of the `java.lang.foreign` package.

https://lucene.apache.org/core/10_5_2/core/org/apache/lucene/store/MMapDirectory.html

Есть хорошие Jextract-плагины для систем сборки

Например, `gradle-jextract`

```
jextract {
    libraries {
        val cApi = register("c_api") {
            header = layout.projectDirectory
                .file("src/main/public/catboost/c_api.h")
            headerClassName = "CatBoostCApi"
            targetPackage = "ru.ozon.o2models.catboost.panama.ffi"
        }
    }

    sourceSets.main {
        jextract.libraries.addLater(cApi)
    }
}
```

<https://infolektuell.github.io/gradle-jextract/>

JPassport

Интересный проект по описанию биндингов Java-интерфейсам

```
public interface Linked extends Passport {  
    // char* in C is equivalent to a Java String  
    int string_length(String s);  
    // Java does not have pointers like the C function, so we use arrays.  
    double sumArrD(double[] arr, int count);  
}
```

...

```
// Get JPassport to create the native bindings for you.  
Linked L = PassportFactory.link("libforeign", Linked.class);
```

```
//These are the calls into native code.  
int n = L.string_length("hello");  
double sum = L.sumArrD(new double[] {1, 2, 3}, 3);
```

<https://github.com/boulder-on/JPassport>

JPassport

Интересный проект по описанию биндингов Java-интерфейсам

```
public interface Linked extends Passport {  
    // char* in C is equivalent to a Java String  
    int string_length(String s);  
    // Java does not have pointers like the C function, so we use arrays.  
    double sumArrD(double[] arr, int count);  
}
```

...

```
// Get JPassport to create the native bindings for you.  
Linked L = PassportFactory.link("libforeign", Linked.class);
```

```
//These are the calls into native code.  
int n = L.string_length("hello");  
double sum = L.sumArrD(new double[] {1, 2, 3}, 3);
```

<https://github.com/boulder-on/JPassport>

JPassport

Интересный проект по описанию биндингов Java-интерфейсам

```
public interface Linked extends Passport {  
    // char* in C is equivalent to a Java String  
    int string_length(String s);  
    // Java does not have pointers like the C function, so we use arrays.  
    double sumArrD(double[] arr, int count);  
}
```

...

```
// Get JPassport to create the native bindings for you.  
Linked L = PassportFactory.link("libforeign", Linked.class);
```

```
//These are the calls into native code.  
int n = L.string_length("hello");  
double sum = L.sumArrD(new double[] {1, 2, 3}, 3);
```

<https://github.com/boulder-on/JPassport>

Мы не отстаём!

И пилим общую библиотеку с биндингами к `libm`

```
"name": "powf",
"description": [ "Calculates `x` raised to the exponent `y`." ,
    "",
    "If `x` and `y` are both `0`, the result is `1`." ],
"returns": "the calculated value, or `HUGE_VAL`
    when the argument values would produce overflow",
"return_type": "float",
"parameters": [
    { "name": "x", "type": "float", "description": "base value" },
    { "name": "y", "type": "float", "description": "exponent value" }
],
"critical" : true
```

<https://github.com/ozontech>

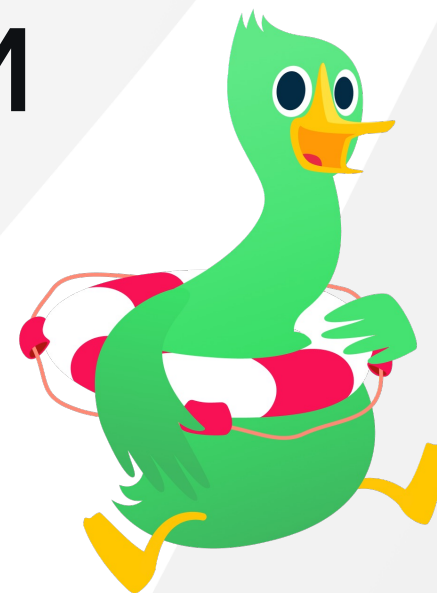
Мы не отстаём!

И пилим общую библиотеку с биндингами к `libm`

```
/// Calculates `x` raised to the exponent `y`.  
///  
/// If `x` and `y` are both `0`, the result is `1`.  
///  
/// @param x base value  
/// @param y exponent value  
/// @return the calculated value,  
///         or `HUGE_VAL` when the argument values would produce overflow  
public static float powf(float x, float y) {  
    try {  
        return (float) MH_powf.invokeExact(x, y);  
    } catch (Throwable throwable) {  
        throw LibcFactories.invocationFailure(  
            throwable, "powf(x=%f, y=%f)", x, y  
        );  
    }  
}
```

<https://github.com/ozontech>

Примеров больше, но
есть и куда **посмотреть**, и
как **воспользоваться**, и
что **изобрести**



06



Заключение

Выводы

 FFM API – это в первую очередь про удобство интеропа



Выводы



FFM API – это в первую очередь про удобство интеропа



Работа может быть быстрее JNI, но не стоит ждать от неё чудес



Выводы



FFM API – это в первую очередь про удобство интеропа



Ranata может быть быстрее JNI, но не стоит ждать от неё чудес



Реальную скорость дают алгоритмы и архитектура, но FFM API упрощает их построение



Выводы



FFM API – это в первую очередь про удобство интеропа



Реальную скорость дают алгоритмы и архитектура, но FFM API упрощает их построение



Rapama может быть быстрее JNI, но не стоит ждать от неё чудес



Туллинг – не без ограничений, но всё в Ваших руках



Темы для кулуаров

01

Зачем тогда писать на Java?

...

02

Как поддерживать смешанные кодовые базы?

...

03

Craby и другие экзотические ABI

...

04

Компромиссы между гарантированной безопасностью и производительностью

...

05

Куда тут приплести ИИ?

...



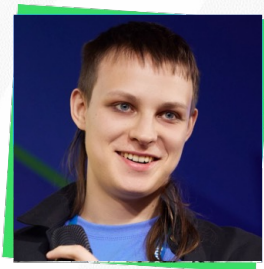


Foreign Function and Memory API
<https://s.ozon.ru/D4Y6m6Q>



Java FFI benchmark
<https://s.ozon.ru/6IHWvsP>

Спасибо за внимание!



Пётр Портнов
ведущий разработчик
информационных систем, Ozon



State of foreign memory support (2023)
<https://s.ozon.ru/j7UNAfC>

 pportnov@ozon.ru

 github.com/JarvisCraft

 github.com/ozontech