Java and C interoperability And Profiling

Zimu Yang
Qiqi Zhou

Objective

Run Java program in pure C environment

Able to retrieve profiling information of Java Program dynamically

How?

JVM runs when C program running

Interface to retrieve JVM status dynamically

Components

Embedded JVM

JNI provides interface of operating on JVM

JVMTI provides interface of retrieving profiling information from JVM

Basic Set-up

#include <jni.h>

#include <jvmti.h>

Embedded JVM

gcc hello_world.c -o hello_world

-L $JAVA_HOME/jre/lib/amd64/server

-I $JAVA_HOME/include/

-I $JAVA_HOME/include/linux/

-ljvm

JVMTI

Communicate with JVM during runtime

Stack Trace

```
Top5 Stack Frame Tracing:
Executing method: clone
Executing method: getParameterTypes
Executing method: getConstructor0
Executing method: newInstance
Executing method: loadImpl
Top5 Stack Frame Tracing:
Executing method: getName0
Executing method: getName
Executing method: isSameClassPackage
Executing method: verifyMemberAccess
Executing method: ensureMemberAccess
```

Heap Allocation

```
type Ljava/lang/reflect/Constructor; object allocated with size 80
type [Ljava/lang/reflect/Constructor; object allocated with size 24
type Ljava/awt/SystemColor$$Lambda$13/1225358173; object allocated with size 16
type [Ljava/lang/Class; object allocated with size 24
type [I object allocated with size 32
type Ljava/awt/Insets; object allocated with size 32
type [Lsun/java2d/loops/GraphicsPrimitive; object allocated with size 1040
type [Lsun/awt/X11/XBaseWindow$InitialiseState; object allocated with size 32
type [Lsun/java2d/loops/GraphicsPrimitive; object allocated with size 1632
type [I object allocated with size 32
type [Ljava/awt/Component; object allocated with size 24
type Ljava/awt/Insets; object allocated with size 32
type Ljava/lang/String; object allocated with size 24
type [C object allocated with size 72
type Ljava/awt/Insets; object allocated with size 32
type [Ljava/lang/reflect/Constructor; object allocated with size 24
type [Ljava/lang/Class; object allocated with size 16
type Ljava/lang/reflect/Constructor; object allocated with size 80
type [Ljava/lang/Class; object allocated with size 16
type [Ljava/lang/reflect/Constructor; object allocated with size 24
type [Ljava/lang/Class; object allocated with size 16
type Ljava/lang/reflect/Constructor; object allocated with size 80
```

Important Point

Memory management

JVM doesn’t know memory usage situation of C program and neither C program does

Both eat up the memory of this process

Important Point

Automatic Garbage Collection is partly discarded

Local Reference should be managed by the programmer

Func(){

Jobject tmp= (*env)->NewObject(env,class1,construct_method);

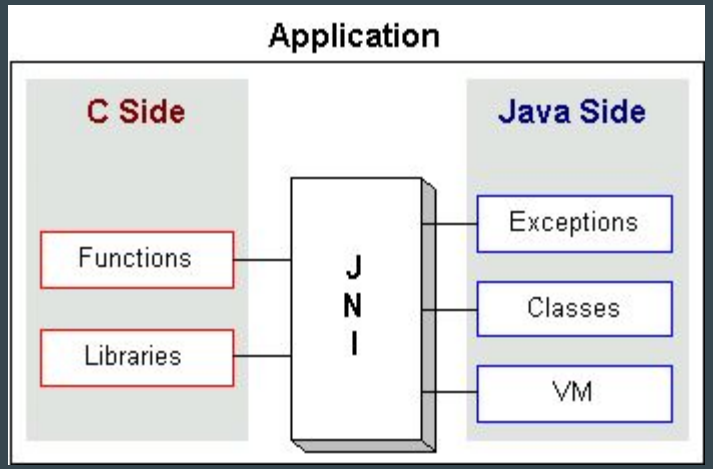
}

Reference tmp keeps on existing after Func returns.

```
Reference kind:JVMTI_HEAP_REFERENCE_JNI_LOCAL
Threads count:4
Threads name:Signal Dispatcher
Threads name:Finalizer
Threads name:Reference Handler
Threads name:main
Reference kind:JVMTI_HEAP_REFERENCE_JNI_LOCAL
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Reference kind:JVMTI_HEAP_REFERENCE_FIELD
Threads count:10
Threads name:Timer-1     Priority:5running for 122 us
Threads name:Timer-0     Priority:5running for 77 us
Threads name:AWT-EventQueue-0    Priority:6running for 1252 us
Threads name:AWT-Shutdown        Priority:5running for 103 us
Threads name:AWT-XAWT    Priority:6running for 3145 us
Threads name:Java2D Disposer     Priority:10running for 87 us
Threads name:Signal Dispatcher   Priority:9running for 54 us
Threads name:Finalizer   Priority:8running for 81 us
Threads name:Reference Handler   Priority:10running for 102 us
Threads name:main        Priority:5running for 170901 us
```

Call C from Java

- Java Native Interface
- A Java layer that allows Java code running in the Java Virtual Machine (JVM) to call and be called by native applications and libraries written in C/C++



JNI in Android

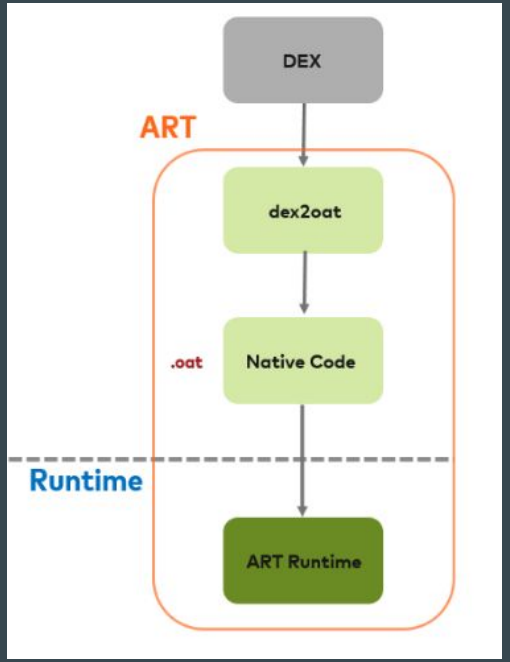
- Widely used in Android application
- Why?
  - Use already built, or share C/C++ libraries
  - C/C++ can offer better performance than what the Java bytecode code will be interpreted to.
- How?
  - Java uses a lot more memory than C due to objects and GC
  - Java doesn't provide the same low-level functionality C does
  - ....

So?

- Android Native Development Kit(NDK)
- Prior to the NDK..
- Android runtime (ART)

Android runtime (ART)

- ART introduces ahead-of-time (AOT) compilation.
- In Android, Java classes converted into DEX bytecode, then translate to machine code.
- Advantages:
  - Apps run fast during installation.
  - Reduces startup time of applications as native code is directly executed
  - Improves garbage collection
    - One GC pause instead of two
    - Parallelized processing during the remaining GC pause

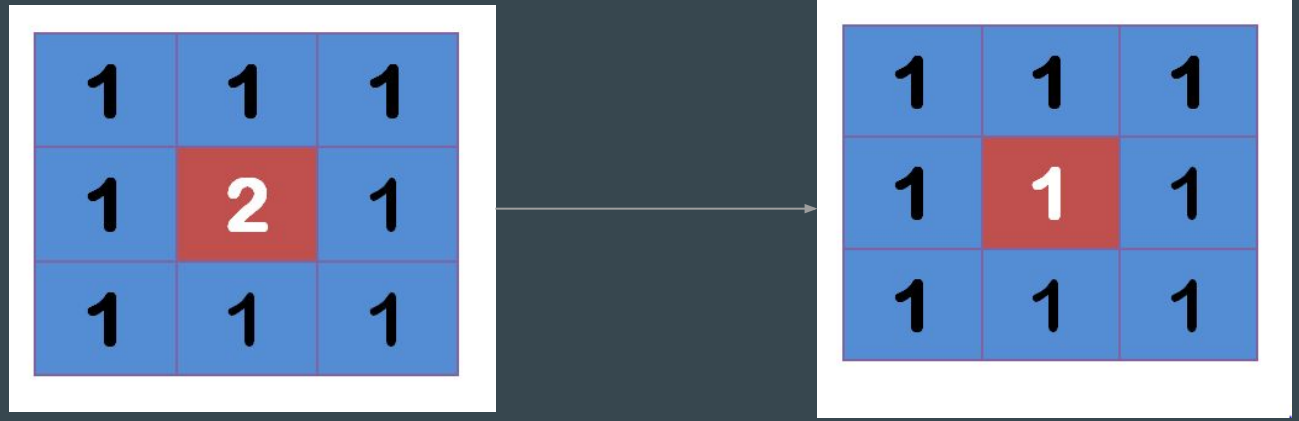


Android Native Development Kit(NDK)

- Allows developers to write code in C/C++ that compiles to native code
- Get better performance because the source code is compiled directly into machine code for the CPU. (Unity or Unreal Engine).
- Based on Java Native Interface.

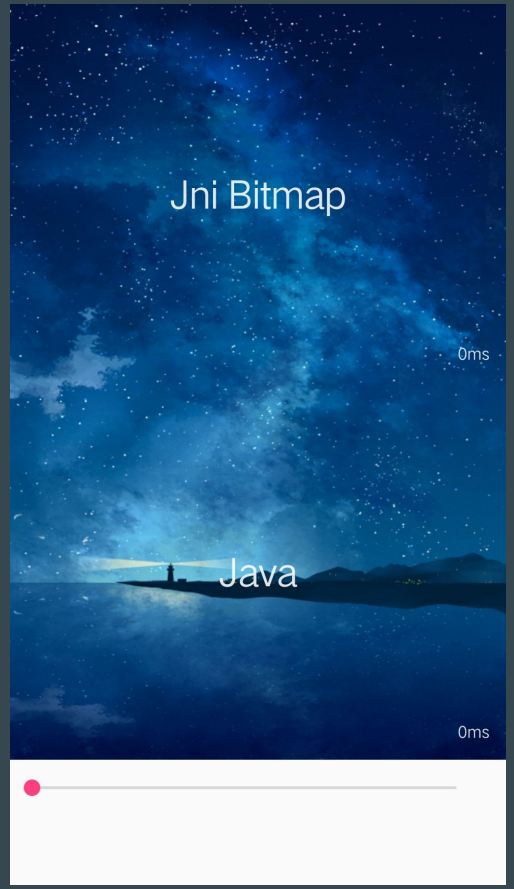
Test Algorithm - Gaussian Blur

- Use Gaussian function to smooth a data set is to create an approximating function that attempts to capture important patterns in the data.

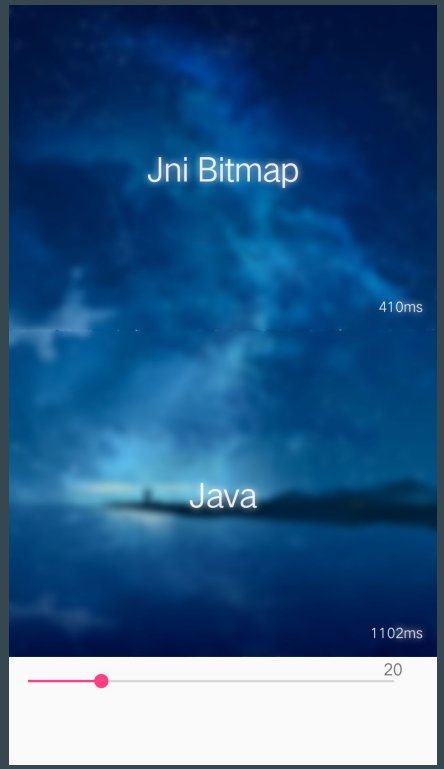


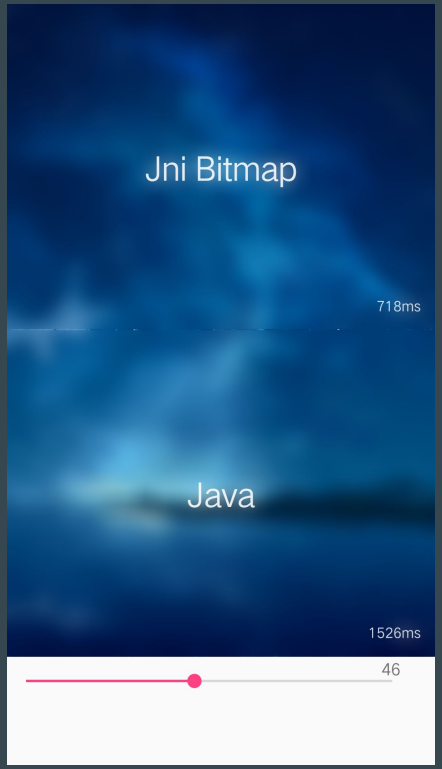
Demo

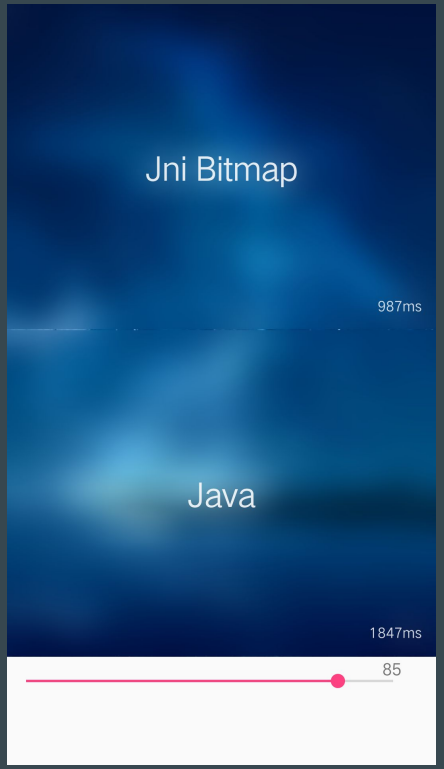
- Android version: 6.0
- CPU: MTK MT6755 2.0GHz 8 cores
- Ram: 4GB



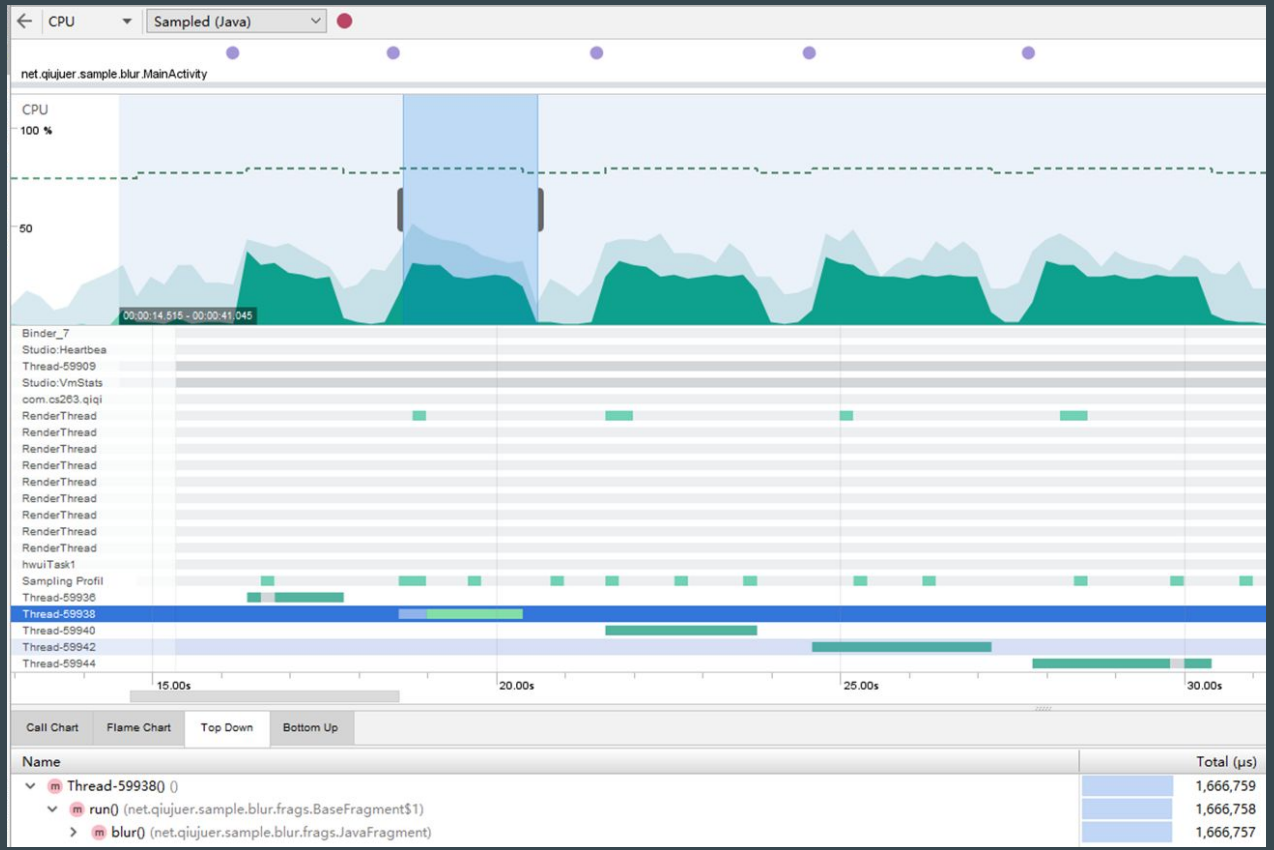
Demo


Jni Bitmap
410ms
Java
1102ms
20

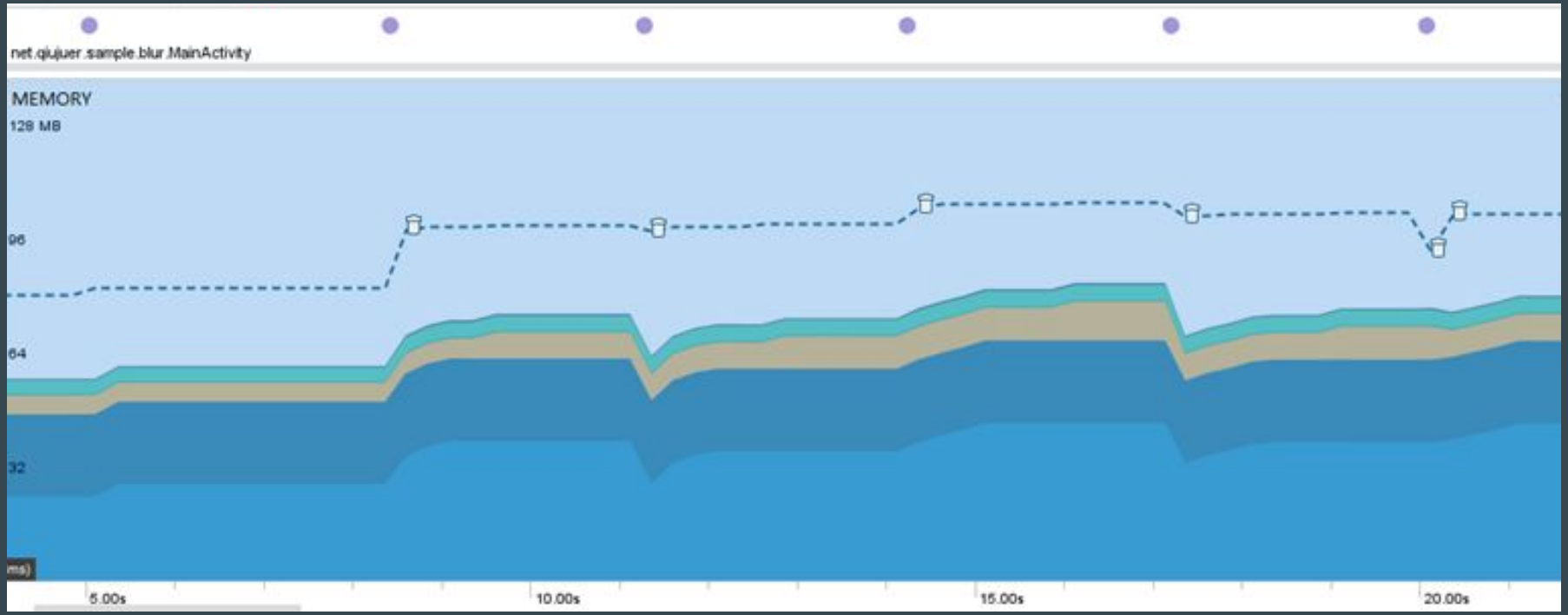

Jni Bitmap
718ms
Java
1526ms
46


Jni Bitmap
987ms
Java
1847ms
85

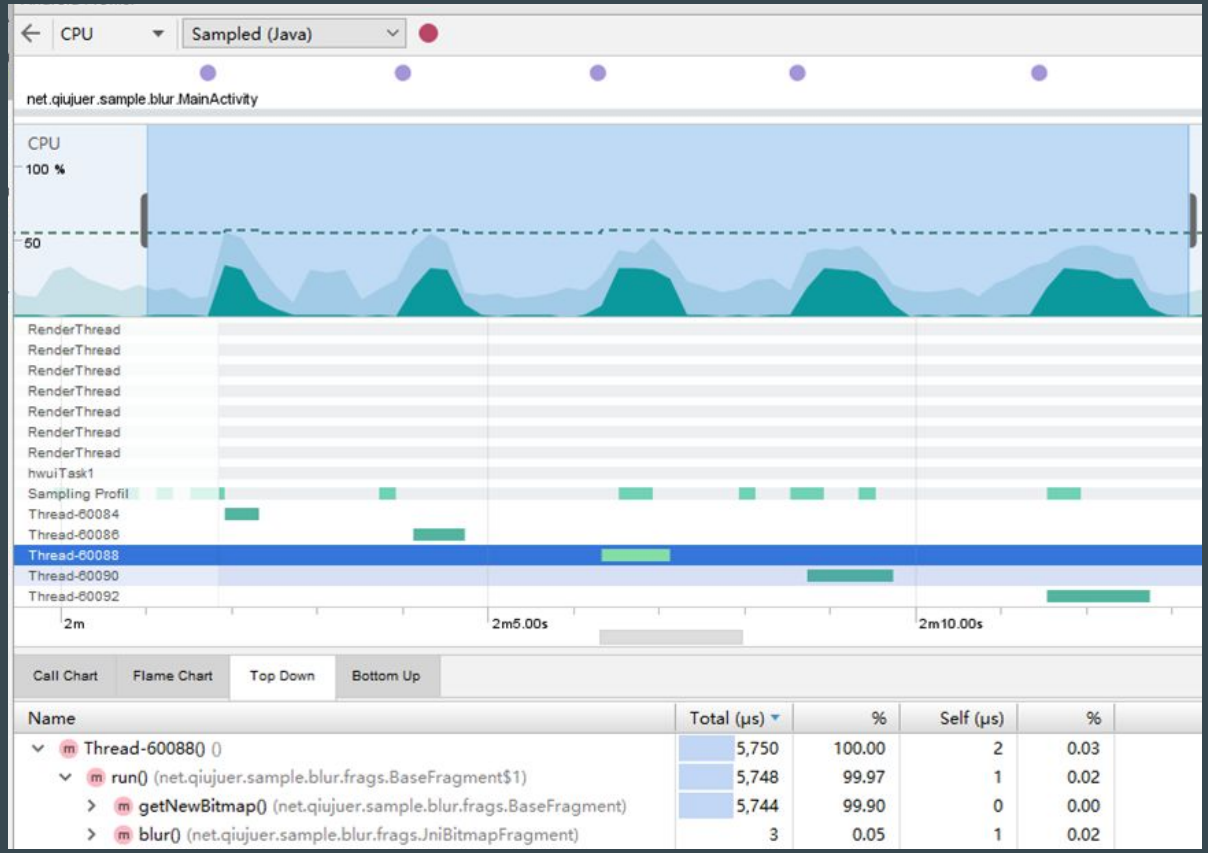
Android Profiler - Java method



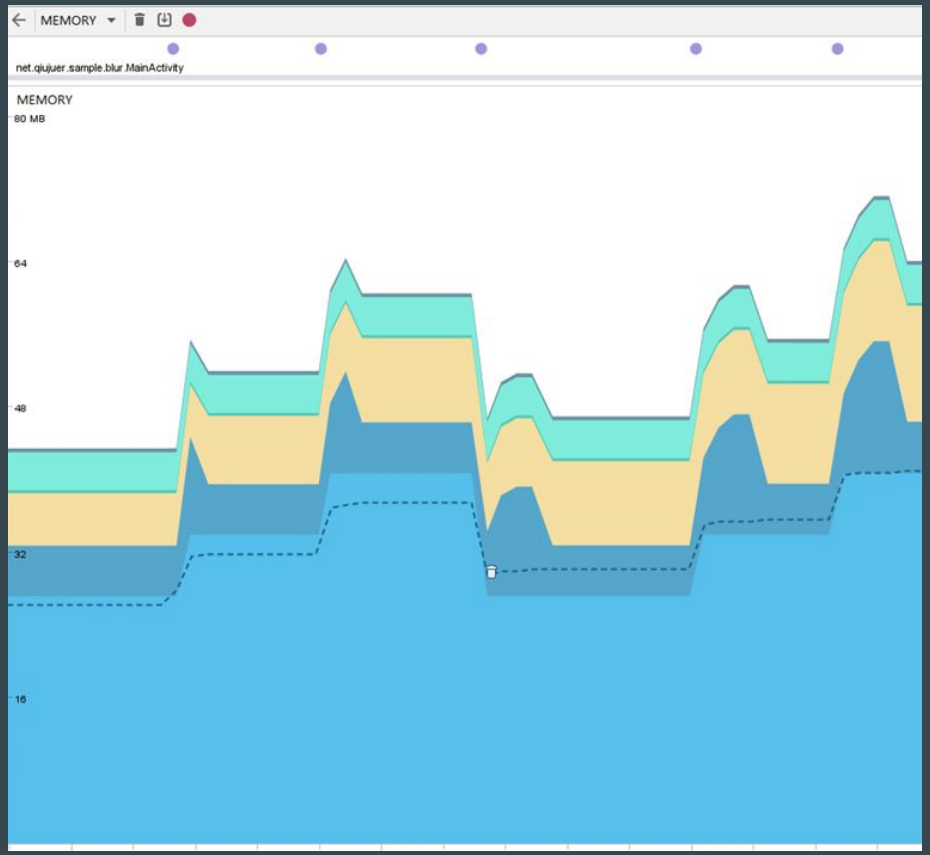
Android Profiler - Java method



Android Profiler - JNI method



Android Profiler - JNI method



Thank you

...