MediaCodec SurfaceTexture listener suboptimal performance

STATUS UPDATE No update yet. Edit

DESCRIPTION ys...@gmail.com created issue #1

The current implementation of the SurfaceTexture::OnFrameAvailableListener is suboptimal in terms of performance.

On two devices I have (5.0 and 5.1), a separate thread is created somewhere[1] for each SurfaceTexture to post events from the native code. This separate thread seems to attach and detach its

See the attached ddms traceview screenshot for details.

The removeThread call is selected showing the performance hit that only occurs in the created SurfaceTexture threads.

[1] I haven't found where this thread is created. One possible solution is to attach/detach on thread creation/destruction instead of per frame which should significantly reduce the performance

deleted
0 B

Mentioned issues (1) Links (3)

Mentioned issues (1)

P3 ACodec causes JDWP to leak weak global references "https://issuetracker.google.com/37089782"

Links (3)

" ...0 and 5.1), a separate thread is created somewhere[1] for each SurfaceTexture to post events from the native code. This separate thread seems to attach and detach itself from the JavaVM on eac
"https://stackoverflow.com/questions/27923917/cant-execute-javavm-detachcurrentthread-attempting-to-detach-whi... "
" ...oduce this issue with the steps provided. If you are able to reproduce the issue on the latest build or have additional information to help us, please let us know by filing a new issue along with the la

COMMENTS

dn...@google.com <dn...@google.com> #2

Assigned to dn...@google.com.

[Comment deleted]

dn...@google.com <dn...@google.com> #3

Status: New

[Comment deleted]

sy...@google.com <sy...@google.com> #4

Assigned to sy...@google.com.

Hi,

Thanks for reporting the issue. We are working on this issue and will provide updates as they are available.

Thanks...

am...@getchannels.com <am...@getchannels.com> #5

I'm seeing this same behavior on Oreo, where the repeated java.lang.Thread creations are putting a lot of pressure on the GC. I'm seeing the GC run every ~30s to clean up 14MB of garbage.

```
07-06 19:24:42.425 10907-10924 I/zygote: Background concurrent copying GC freed 51522(14MB) AllocSpace objects, 0(0B) LOS objects, 49% free, 14MB/28MB, paused 660us total 175.82
07-06 19:25:08.025 10907-10924 I/zygote: Background concurrent copying GC freed 57068(14MB) AllocSpace objects, 0(0B) LOS objects, 49% free, 14MB/28MB, paused 456us total 150.67
07-06 19:25:34.427 10907-10924 I/zygote: Background concurrent copying GC freed 46596(14MB) AllocSpace objects, 0(0B) LOS objects, 50% free, 14MB/28MB, paused 450us total 126.12
07-06 19:26:00.328 10907-10924 I/zygote: Background concurrent copying GC freed 49722(14MB) AllocSpace objects, 0(0B) LOS objects, 49% free, 14MB/29MB, paused 4.272ms total 205.
07-06 19:26:27.092 10907-10924 I/zygote: Background concurrent copying GC freed 59695(15MB) AllocSpace objects, 0(0B) LOS objects, 49% free, 14MB/28MB, paused 392us total 135.53
07-06 19:26:53.188 10907-10924 I/zygote: Background concurrent copying GC freed 52069(14MB) AllocSpace objects, 0(0B) LOS objects, 49% free, 14MB/28MB, paused 1.904ms total 139.
```

I suspected that the java.lang.Thread were being created via AttachCurrentThread, and a breakpoint indeed confirms that Attach/Detach are being called for every single frame. See attached

I was able to find some other references online which confirm the same behavior as noticed by others over the years:

https://stackoverflow.com/questions/27923917/cant-execute-javavm-detachcurrentthread-attempting-to-detach-while-still-r/27927141#27927141
https://issuetracker.google.com/issues/37089782#comment6

Poking around the history, I see that the detach was added by @jgennis in 0a8fd9b610b2de92930c92d71ac184dc9e2bcb4d

Because of the constant Attach/Detach and subsequent GC pressure, I'm unable to use MediaCodec with SurfaceTexture/updateTexImage based rendering on Oreo because performance su

deleted
0 B

sy...@google.com <sy...@google.com>

Status: New

sa...@google.com <sa...@google.com> #6

Status: Won't Fix (Not Reproducible)

After investigating, we were not able to reproduce this issue with the steps provided. If you are able to reproduce the issue on the latest build or have additional information to help us, please

am...@getchannels.com <am...@getchannels.com> #7

I believe this bug is still present, and the underlying code in jni/android/graphics/SurfaceTexture.cpp has not changed since 2011.

On a newly release Android TV device running API 28, I see a thread created somewhere referencing JNISurfaceTextureContext:

> thread #79: tid = 5419, 0xf40ded54 libc.so`syscall + 28, name = 'JNISurfaceTextu'

and repeated blocking GC during rendering caused specifically by that tid 5419:

```
2020-05-13 23:30:55.415 3948-5419/myapp I/myapp: Starting a blocking GC Alloc
2020-05-13 23:30:55.415 3948-5419/myapp I/myapp: Starting a blocking GC Alloc
2020-05-13 23:30:55.418 3948-4066/myapp I/myapp: Waiting for a blocking GC Alloc
2020-05-13 23:30:55.419 3948-5399/myapp I/myapp: Waiting for a blocking GC Alloc
2020-05-13 23:30:55.589 3948-5456/myapp I/myapp: Waiting for a blocking GC Alloc
2020-05-13 23:30:55.754 3948-5419/myapp I/myapp: Alloc concurrent copying GC freed 71085(17MB) AllocSpace objects, 0(0B) LOS objects, 50% free, 20MB/41MB, paused 657us total 338
2020-05-13 23:30:55.754 3948-4066/myapp I/myapp: WaitForGcToComplete blocked Alloc on ProfileSaver for 336.186ms
2020-05-13 23:30:55.754 3948-4066/myapp I/myapp: Starting a blocking GC Alloc
2020-05-13 23:30:55.754 3948-5399/myapp I/myapp: WaitForGcToComplete blocked Alloc on ProfileSaver for 334.977ms
2020-05-13 23:30:55.754 3948-5399/myapp I/myapp: Starting a blocking GC Alloc
2020-05-13 23:30:55.754 3948-5456/myapp I/myapp: WaitForGcToComplete blocked Alloc on ProfileSaver for 165.931ms
2020-05-13 23:30:55.754 3948-5456/myapp I/myapp: Starting a blocking GC Alloc
```

This can be reproduced simply by creating a TextureView on screen, then feeding its underlying Surface into MediaCodec for playback.

The solutions are either:

1) update JNISurfaceTextureContext::getJNIEnv to use the pthread_setspecific/pthread_getspecific pattern to bind the JNIEnv once to this thread, and have it automatically detach during thr

2) find out where this thread is being created and ensure it is created in a way such that `AndroidRuntime::getJNIEnv()` returns a value, so `JNISurfaceTextureContext::detachJNI()` is never c

am...@getchannels.com <am...@getchannels.com> #8

A breakpoint on `AttachCurrentThread` (used with any TextureView based application) will show the full stack trace where `onFrameAvailable` binds/unbind, and also confirm that this is ha

```
* thread #111, name = 'CodecLooper', stop reason = breakpoint 8.3
  * frame #0: 0xf1015330 libart.so`art::Runtime::AttachCurrentThread(char const*, bool, _jobject*, bool)
    frame #1: 0xf0efa14c libart.so`art::JII::AttachCurrentThreadInternal(_JavaVM*, _JNIEnv**, void*, bool) + 244
    frame #2: 0xf0d8d972 libart.so`art::(anonymous namespace)::CheckJII::AttachCurrentThread(_JavaVM*, _JNIEnv**, void*) + 82
    frame #3: 0xf3915800 libandroid_runtime.so`android::JNISurfaceTextureContext::onFrameAvailable(android::BufferItem const&) + 80
    frame #4: 0xf3ebf686 libgui.so`android::ConsumerBase::onFrameAvailable(android::BufferItem const&) + 110
    frame #5: 0xf3eb5f1e libgui.so`android::BufferQueue::ProxyConsumerListener::onFrameAvailable(android::BufferItem const&) + 54
    frame #6: 0xf3ebd382 libgui.so`android::BufferQueueProducer::queueBuffer(int, android::IGraphicBufferProducer::QueueBufferInput const&, android::IGrap
    frame #7: 0xf3ed8298 libgui.so`android::Surface::queueBuffer(ANativeWindowBuffer*, int) + 904
    frame #8: 0xf28e25e4 libstagefright.so`android::ACodec::BaseState::onOutputBufferDrained(android::sp<android::AMessage> const&) + 1752
    frame #9: 0xf28e0418 libstagefright.so`android::ACodec::BaseState::onMessageReceived(android::sp<android::AMessage> const&) + 588
    frame #10: 0xf41d25b0 libstagefright_foundation.so`android::AHierarchicalStateMachine::handleMessage(android::sp<android::AMessage> const&) + 64
    frame #11: 0xf41d242a libstagefright_foundation.so`android::AHandler::deliverMessage(android::sp<android::AMessage> const&) + 26
    frame #12: 0xf41d4d1c libstagefright_foundation.so`android::AMessage::deliver() + 64
    frame #13: 0xf41d3104 libstagefright_foundation.so`android::ALooper::loop() + 480
    frame #14: 0xf32d21c0 libutils.so`android::Thread::_threadLoop(void*) + 288
    frame #15: 0xf4128c16 libc.so`__pthread_start(void*) + 24
    frame #16: 0xf40e3066 libc.so`__start_thread + 24

* thread #111, name = 'JNISurfaceTextu', stop reason = breakpoint 8.3
  * frame #0: 0xf1015330 libart.so`art::Runtime::AttachCurrentThread(char const*, bool, _jobject*, bool)
    frame #1: 0xf0efa14c libart.so`art::JII::AttachCurrentThreadInternal(_JavaVM*, _JNIEnv**, void*, bool) + 244
    frame #2: 0xf0d8d972 libart.so`art::(anonymous namespace)::CheckJII::AttachCurrentThread(_JavaVM*, _JNIEnv**, void*) + 82
    frame #3: 0xf3915800 libandroid_runtime.so`android::JNISurfaceTextureContext::onFrameAvailable(android::BufferItem const&) + 80
    frame #4: 0xf3ebf686 libgui.so`android::ConsumerBase::onFrameAvailable(android::BufferItem const&) + 110
    frame #5: 0xf3eb5f1e libgui.so`android::BufferQueue::ProxyConsumerListener::onFrameAvailable(android::BufferItem const&) + 54
    frame #6: 0xf3ebd382 libgui.so`android::BufferQueueProducer::queueBuffer(int, android::IGraphicBufferProducer::QueueBufferInput const&, android::IGrap
    frame #7: 0xf3ed8298 libgui.so`android::Surface::queueBuffer(ANativeWindowBuffer*, int) + 904
    frame #8: 0xf28e25e4 libstagefright.so`android::ACodec::BaseState::onOutputBufferDrained(android::sp<android::AMessage> const&) + 1752
    frame #9: 0xf28e0418 libstagefright.so`android::ACodec::BaseState::onMessageReceived(android::sp<android::AMessage> const&) + 588
    frame #10: 0xf41d25b0 libstagefright_foundation.so`android::AHierarchicalStateMachine::handleMessage(android::sp<android::AMessage> const&) + 64
    frame #11: 0xf41d242a libstagefright_foundation.so`android::AHandler::deliverMessage(android::sp<android::AMessage> const&) + 26
    frame #12: 0xf41d4d1c libstagefright_foundation.so`android::AMessage::deliver() + 64
    frame #13: 0xf41d3104 libstagefright_foundation.so`android::ALooper::loop() + 480
    frame #14: 0xf32d21c0 libutils.so`android::Thread::_threadLoop(void*) + 288
    frame #15: 0xf4128c16 libc.so`__pthread_start(void*) + 24
    frame #16: 0xf40e3066 libc.so`__start_thread + 24
```