DESCRIPTION qi...@xiaomi.corp-partner.google.com created issue #1

1.Problem Description

ANR for app com.tencent.qqlivei18n.

2. Prerequisites

1. Open Google Play and search keyword wetv.
2. Download and install the first app in search result.

3. Steps to reproduce

1. Start the app.
2. Drag up and down in the main Activity for several seconds. No more than 1 min.
3. App ANR is notified.
4. The problem will only occur when the app is opened for the first time and the second time, which can be achieved by clearing the data.

4. Expected Result

App runs normally.

5. Actual result

App not respond after

6. Recurrence probability

100%

7. Initial analyze

Stack trace of main thread:

```
----- pid 10491 at 2022-06-20 19:05:23.971123404+0800 -----
Cmd line: com.tencent.qqlivei18n
ABI: 'arm64'

"cent.qqlivei18n" sysTid=10491
    #00 pc 0000000000086f90  /apex/com.android.runtime/lib64/bionic/libc.so (syscall+32) (BuildId: 94065bf91428f6ae9fb310c478171302)
    #01 pc 000000000028dc74  /apex/com.android.art/lib64/libart.so (art::ConditionVariable::WaitHoldingLocks(art::Thread*)+152) (BuildId: 0b1233dc4b2f0a31e3b
    #02 pc 0000000000466d4c  /apex/com.android.art/lib64/libart.so (art::JNI<false>::FindClass(_JNIEnv*, char const*)+516) (BuildId: 0b1233dc4b2f0a31e3b16ee8
    #03 pc 00000000001a7f2c  /system/lib64/libandroid_runtime.so ((anonymous namespace)::Receiver::handleEvent(int, int, void*)+128) (BuildId: 555f129e00323a
    #04 pc 0000000000018184  /system/lib64/libutils.so (android::Looper::pollInner(int)+916) (BuildId: 16796d84bdcf185b2112267dbd820c19)
    #05 pc 0000000000017d84  /system/lib64/libutils.so (android::Looper::pollOnce(int, int*, int*, void**)+116) (BuildId: 16796d84bdcf185b2112267dbd820c19)
    #06 pc 0000000000154674  /system/lib64/libandroid_runtime.so (android::android_os_MessageQueue_nativePollOnce(_JNIEnv*, _jobject*, long, int)+48) (BuildI
    #07 pc 00000000020103dc  /memfd:jit-cache (deleted) (offset 0x2000000) (art_jni_trampoline+108)
```

Frame #02 addr2line:

```
0000000000466d4c
art::JNI<false>::FindClass(_JNIEnv*, char const*)
art/runtime/base/mutex.cc:1068
art::Thread::TransitionFromSuspendedToRunnable()
art/runtime/thread-inl.h:293
ScopedThreadStateChange
art/runtime/scoped_thread_state_change-inl.h:47
ScopedObjectAccessUnchecked
art/runtime/scoped_thread_state_change-inl.h:105
ScopedObjectAccess
art/runtime/scoped_thread_state_change-inl.h:116
art::JNI<false>::FindClass(_JNIEnv*, char const*)
art/runtime/jni/jni_internal.cc:597
```

Please note that this version of libart is from MIUI. I've replaced with it for debugging. Implementation of TransitionFromSuspendedToRunnable:

```
//art/runtime/thread-inl.h
243 inline ThreadState Thread::TransitionFromSuspendedToRunnable() {
244   union StateAndFlags old_state_and_flags;
245   old_state_and_flags.as_int = tls32_.state_and_flags.as_int;
246   int16_t old_state = old_state_and_flags.as_struct.state;
247   DCHECK_NE(static_cast<ThreadState>(old_state), kRunnable);
248   do {
249     Locks::mutator_lock_->AssertNotHeld(this);  // Otherwise we starve GC..
250     old_state_and_flags.as_int = tls32_.state_and_flags.as_int;
251     DCHECK_EQ(old_state_and_flags.as_struct.state, old_state);
252     if (LIKELY(old_state_and_flags.as_struct.flags == 0)) {
...
275     } else if ((old_state_and_flags.as_struct.flags & kSuspendRequest) != 0) {
...
281       Thread* thread_to_pass = nullptr;
282       if (kIsDebugBuild && !IsDaemon()) {
283         // We know we can make our debug locking checks on non-daemon threads,
284         // so re-enable them on debug builds.
285         thread_to_pass = this;
286       }
287       MutexLock mu(thread_to_pass, *Locks::thread_suspend_count_lock_);
288       ScopedTransitioningToRunnable scoped_transitioning_to_runnable(this);
289       old_state_and_flags.as_int = tls32_.state_and_flags.as_int;
290       DCHECK_EQ(old_state_and_flags.as_struct.state, old_state);
291       while ((old_state_and_flags.as_struct.flags & kSuspendRequest) != 0) {
292         // Re-check when Thread::resume_cond_ is notified.
293         Thread::resume_cond_->Wait(thread_to_pass);
294         old_state_and_flags.as_int = tls32_.state_and_flags.as_int;
295         DCHECK_EQ(old_state_and_flags.as_struct.state, old_state);
296       }
297       DCHECK_EQ(GetSuspendCount(), 0);
298     }
299   } while (true);
```

We can know it's waiting condition variable Thread::resume_cond_. It seems that no one would notify it to continue, so it hangs.

After checking the state of all threads, I found that most of them are in state futex_wait_queue_me, including main thread. And I found a thread in a rare state:

```
"Thread-101" sysTid=11596
    #00 pc 0000000000086f94  /apex/com.android.runtime/lib64/bionic/libc.so (syscall+36) (BuildId: 94065bf91428f6ae9fb310c478171302)
    #01 pc 000000000028bdd0  /apex/com.android.art/lib64/libart.so (art::Mutex::ExclusiveUnlock(art::Thread*)+292) (BuildId: 0b1233dc4b2f0a31e3b16ee8701a0265
    #02 pc 00000000006a2db4  /apex/com.android.art/lib64/libart.so (art::ThreadList::SuspendThreadByThreadId(unsigned int, art::SuspendReason, bool*)+1228) (
    #03 pc 000000000059dbd8  /apex/com.android.art/lib64/libart.so (art::DdmVmInternal_getStackTraceById(_JNIEnv*, _jclass*, int)+548) (BuildId: 0b1233dc4b2f
    #04 pc 000000000001057c  /apex/com.android.art/javalib/arm64/boot-core-libart.oat (art_jni_trampoline+108) (BuildId: e0613e00e37546ce5aeb39b7362d35446030
    #05 pc 00000000001beb98  /apex/com.android.art/javalib/arm64/boot.oat (java.lang.Thread.getStackTrace+40) (BuildId: b56892b25f4b9c0189054b2248db0a5d170be
    #06 pc 0000000000212520  /apex/com.android.art/lib64/libart.so (nterp_helper+4016) (BuildId: 0b1233dc4b2f0a31e3b16ee8701a0265)
    #07 pc 00000000000ed13a  /apex/com.android.art/javalib/core-oj.jar
    #08 pc 0000000002081634  /memfd:jit-cache (deleted) (offset 0x2000000) (com.appsflyer.internal.b.$$a+116)
```

```
0000000000466d4c
art::JNI<false>::FindClass(_JNIEnv*, char const*)
art/runtime/base/mutex.cc:1068
art::Thread::TransitionFromSuspendedToRunnable()
art/runtime/thread-inl.h:293
ScopedThreadStateChange
art/runtime/scoped_thread_state_change-inl.h:47
ScopedObjectAccessUnchecked
art/runtime/scoped_thread_state_change-inl.h:105
ScopedObjectAccess
art/runtime/scoped_thread_state_change-inl.h:116
art::JNI<false>::FindClass(_JNIEnv*, char const*)
art/runtime/jni/jni_internal.cc:597
```

```
1012 Thread* ThreadList::SuspendThreadByThreadId(uint32_t thread_id,
1013                                             SuspendReason reason,
1014                                             bool* timed_out) {
...
1022   while (true) {
1023     {
...
1049       {
1050         MutexLock suspend_count_mu(self, *Locks::thread_suspend_count_lock_);
...
1101       }
```

The thread is blocked when unlocking. It's so strange. I start to think it's a problem of ART.
So I checked the version of art on Pixel, in which problem doesn't occur:

```
Active APEX packages:

  com.google.android.art
    Version: 311810000
    Path: /data/apex/active/com.android.art@311810000.apex
    IsActive: true
    IsFactory: false
```

and on MIUI, in which problem occurs:

```
Active APEX packages:

  com.google.android.art
    Version: 311713000
    Path: /data/apex/active/com.android.art@311713000.apex
    IsActive: true
    IsFactory: false
```

Settings app shows the date of version 311713000 is May the 1st, and 311810000 is June the 1st.
So could you tell me when we can update art to version 311810000? Is there any way to get it for test? Or my analyze is wrong? Thanks very much!

COMMENTS

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

Assigned to su...@google.com.

Thank you for reporting this issue. For us to further investigate this issue, please provide the following additional information:

We are not able to find an application using the package name " com.tencent.qqlivei18n " and also we tried to get for another application as you metioned in https://buganizer.corp.google.co
couldn't find it .

Please confirm that in which application ANR happened and also share the package name , playstore link .

Android full bug report capturing

After reproducing the issue, press the volume up, volume down, and power button simultaneously. This will capture a bug report on your device in the “bug reports” directory.

Alternate method

Navigate to “Developer options”, ensure “USB debugging” is enabled, then enable “Bug report shortcut”. Capture bug report by holding the power button and selecting the “Take bug report” op

Screen record of the issue, for clarity

Please capture screen record or video of the issue using following steps:

adb shell screenrecord /sdcard/video.mp4

Subsequently use following command to pull the recorded file:

adb pull /sdcard/video.mp4

Note: Please upload the files to google drive and share the folder to android-bugreport@google.com, then share the link here.

---

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

Please share the information requested in comment#2 to proceed further.

---

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

Status: Won't Fix (Infeasible)

We are closing this issue since we didn't receive a response. If you are still facing this problem, please open a new issue and add the relevant information along with reference to this issue.