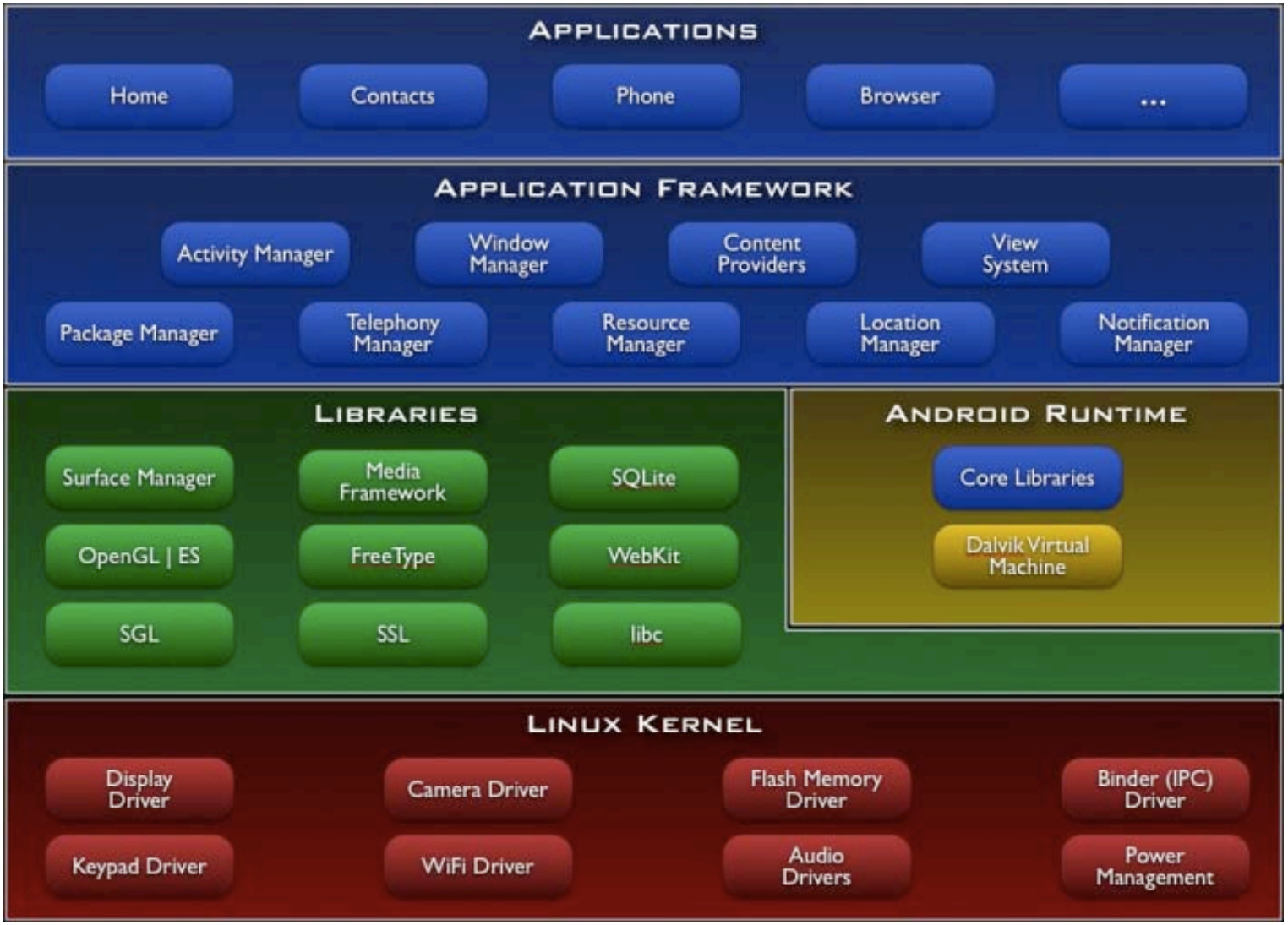
Android Architecture Patterns I



Android Architecture Patterns II

This architecture patterns goes beyond the shared library and runtime linkage stuff that is used by traditional operating systems.

A process (Zygote child) is a container for software components.
Software components are activities, services and content providers.

Android offers that software component architecture in a multi-process environment that is also protected by multiple virtual machines.

The Zygote architecture pattern allows Android to set up such software component containers (OS processes that contain a Dalvik VM together with the Android framework classes) within milliseconds (→ see also Microsoft‘s Singularity OS architecture).

Android‘s Binder runtime helps to make these software component containers interoperable in a very efficient way.

→ On top of the monolithic Linux kernel Android looks very much like a modern Microkernel OS (comparable to QNX Neutrino RTOS or Miray Symobi).

Android startActivity Log I

```
1.      I/Activity(  112): startActivity (TID: 1)
2.      I/Activity(  112): startActivityForResult (TID: 1)
3.      I/Instrumentation(  112): execStartActivity (TID: 1)
4.      I/ActivityManagerProxy(  112): startActivity (TID: 1)
5.      I/ActivityManagerNative(   58): onTransact(START_ACTIVITY_TRANSACTION) (TID: 7)
6.      I/ActivityManagerService(   58): startActivity (TID: 7)
7.      I/ActivityManagerService(   58): Starting activity: Intent { act=android.intent.action.MAIN
        cat=[android.intent.category.LAUNCHER] flg=0x10200000 cmp=com.android.alarmclock/.AlarmClock }
8.      I/ActivityManagerService(   58): startActivityLocked -> new HistoryRecord caller ProcessRecord{43c8a0f0
        android.process.acore}, HistoryRecord{43ce7db0 com.android.alarmclock/.AlarmClock}
9.      I/ActivityManagerService(   58): startActivityUncheckedLocked (TID: 7)
10.     I/ActivityManagerService(   58): startActivityLocked (3 args) (TID: 7)
11.     I/ActivityManagerService(   58): resumeTopActivityLocked (TID: 7)
12.     I/ActivityManagerService(   58): resumeTopActivityLocked -> startPausingLocked (TID: 7)
13.     I/ActivityManagerService(   58): startPausingLocked -> schedulePauseActivity (TID: 7)
14.     I/ApplicationThreadProxy(   58): schedulePauseActivity (TID: 7)
15.     I/ApplicationThreadNative(  112): onTransact(SCHEDULE_PAUSE_ACTIVITY_TRANSACTION) (TID: 7)
16.     I/ActivityThread.ApplicationThread(  112): schedulePauseActivity (TID: 7)
17.     I/ActivityManagerProxy(  112): startActivity (TID: 1) returned: 0
18.     I/ActivityThread(  112): handleMessage: PAUSE_ACTIVITY
19.     I/ActivityThread(  112): handlePauseActivity (TID: 1)
20.     I/ActivityManagerProxy(  112): activityPaused (TID: 1)
21.     I/ActivityManagerNative(   58): onTransact(ACTIVITY_PAUSED_TRANSACTION) (TID: 35)
22.     I/ActivityManagerService(   58): activityPaused
23.     I/ActivityManagerService(   58): completePauseLocked -> resumeTopActivityLocked (TID: 35)
24.     I/ActivityManagerService(   58): resumeTopActivityLocked (TID: 35)
25.     I/ActivityManagerService(   58): calling startSpecificActivityLocked (2) (TID: 35)
26.     I/ActivityManagerService(   58): startSpecificActivityLocked -> process not running: HistoryRecord{43ce7db0
        com.android.alarmclock/.AlarmClock}
27.     I/ActivityManagerService(   58): startSpecificActivityLocked -> startProcessLocked (TID: 35)
28.     I/ActivityManagerService(   58): startProcessLocked
29.     I/ActivityManagerService(   58): newProcessRecordLocked: ProcessRecord{43ceb438 com.android.alarmclock}
30.     I/ActivityManagerService(   58): startProcessLocked - Process.start (TID: 35)
31.     I/Process (   58): startViaZygote
32.     I/Process (   58): zygoteSendArgsAndGetPid
33.     I/Process (   58): openZygoteSocketIfNeeded (TID: 35)
34.     I/Process (   58): zygoteSendArgsAndGetPid1 (TID: 35)
35.     I/ZygoteInit(   29): runSelectLoopMode2
36.     I/ZygoteConnection(   29): runOnce -> forkAndSpecialize
37.     E/dalvikvm(   29): Dalvik_dalvik_system_Zygote_forkAndSpecialize
38.     E/dalvikvm(   29): NOW FORKING!!!!
39.     I/Process (   58): zygoteSendArgsAndGetPid2: 237
40.     I/ActivityManager(   58): Start proc com.android.alarmclock for activity com.android.alarmclock/.AlarmClock:
        pid=237 uid=10016 gids={}
```

Android startActivity Log II

```
41.     E/dalvikvm(  237): Forked new process from Zygote with PID 237
42.     E/ZygoteConnection(  237): handleChildProc
43.     E/ZygoteConnection(  237): invokeStaticMain: android.app.ActivityThread
44.     E/ZygoteConnection(  237): calling RuntimeInit.zygoteInit
45.     D/ddm-heap(  237): Got feature list request
46.     I/RuntimeInit(  237): zygoteInit
47.     E/appproc (  237): AppRuntime::onZygoteInit
48.     E/ProcessState(  237): ProcessState::self creating ProcessState
49.     E/ProcessState(  237): ProcessState - open_driver
50.     E/ProcessState(  237): ProcessState - ctor
51.     E/ProcessState(  237): ProcessState - startThreadPool
52.     E/ProcessState(  237): ProcessState - Spawning new pooled thread, name=Binder Thread #1
53.     E/ProcessState(  237): PoolThread - ID: 241
54.     E/IPCThreadState(  237): IPCThreadState::self
55.     E/IPCThreadState(  237): IPCThreadState ctor
56.     E/IPCThreadState(  237): **** THREAD 0x138740 (241) (PID 237) IS JOINING THE THREAD POOL
57.     E/IPCThreadState(  237): IPCThreadState::executeCommand -> spawnPooledThread (TID: 241, PID: 237)
58.     E/ProcessState(  237): ProcessState - Spawning new pooled thread, name=Binder Thread #2
59.     E/ProcessState(  237): PoolThread - ID: 242
60.     E/IPCThreadState(  237): IPCThreadState::self
61.     E/IPCThreadState(  237): IPCThreadState ctor
62.     E/IPCThreadState(  237): **** THREAD 0x13bae8 (242) (PID 237) IS JOINING THE THREAD POOL
63.     E/ActivityThread(  237): main (TID: 1)
64.     I/ApplicationThreadNative(  239): ctor
65.     I/ActivityThread(  237): attach (TID: 1)
66.     E/ServiceManager(  237): getIServiceManager: 1
67.     E/JavaBinder(  237): android_os_BinderInternal_getContextObject
68.     E/IPCThreadState(  237): IPCThreadState::self
69.     E/IPCThreadState(  237): IPCThreadState ctor
70.     I/ActivityManagerNative(  237): connecting to ActivityManagerService...
71.     I/ActivityManagerProxy(  237): attachApplication (TID: 1)
72.     I/ActivityManagerService(   58): attachApplicationLocked (TID: 34)
73.     I/ApplicationThreadProxy(   58): bindApplication (TID: 34)
74.     I/ApplicationThreadNative(  237): onTransact(BIND_APPLICATION_TRANSACTION) (TID: 6)
75.     I/ActivityThread.ApplicationThread(  237): bindApplication (TID: 6)
76.     I/ActivityManagerService(   58): attachApplicationLocked -> realStartActivityLocked
77.     I/ActivityManagerService(   58): realStartActivityLocked: ProcessRecord{43ceb438 com.android.alarmclock},
        HistoryRecord{43ce7db0 com.android.alarmclock/.AlarmClock}
78.     I/ActivityManagerService(   58): realStartActivityLocked1: 34, 58, com.android.alarmclock
79.     I/ApplicationThreadProxy(   58): scheduleLaunchActivity (TID: 34)
80.     I/ApplicationThreadNative(  237): onTransact(SCHEDULE_LAUNCH_ACTIVITY_TRANSACTION) (TID: 7)
```

Android startActivity Log III

```
81.     I/ActivityThread.ApplicationThread(  237): scheduleLaunchActivity (TID: 7)
82.     I/ActivityManagerService(   58): realStartActivityLocked2: 34, 58, com.android.alarmclock
83.     I/ActivityThread(  237): My Looper ID: 1
84.     I/ActivityThread(  237): Main Looper ID: 1
85.     I/ActivityThread(  237): handleMessage: BIND_APPLICATION
86.     I/ActivityThread(  237): handleBindApplication (TID: 1)
87.     I/ActivityThread(  237): handleBindApplication -> makeApplication
88.     I/ActivityThread.PackageInfo(  237): makeApplication (TID: 1)
89.     I/ActivityThread.PackageInfo(  237): makeApplication (new ApplicationContext)
90.     I/Instrumentation(  237): newApplication(android.app.Application) (TID: 1)
91.     I/ActivityThread(  237): Publishing provider com.android.alarmclock: com.android.alarmclock.AlarmProvider
92.     I/ActivityThread(  237): handleMessage: LAUNCH_ACTIVITY
93.     I/ActivityThread(  237): LAUNCH_ACTIVITY from thread ID: 1, 237
94.     I/Instrumentation(  237): newActivity(com.android.alarmclock.AlarmClock) (TID: 1)
95.     I/ActivityThread(  237): performLaunchActivity -> makeApplication (TID: 1)
96.     I/ActivityThread.PackageInfo(  237): makeApplication returns existing app (TID: 1)
97.     I/ActivityThread(  237): performLaunchActivity -> new ApplicationContext (TID: 1)
98.     I/ActivityThread(  237): performLaunchActivity -> activity.attach
99.     I/Activity(  237): attach -> attaching ApplicationContext to this activity
100.    I/Activity(  237): PolicyManager.makeNewWindow
101.    I/Activity(  237): UI Thread ID: 1 - PID: 237
102.    I/Window  (  237): setWindowManager(null)
103.    I/Instrumentation(  237): callActivityOnCreate (TID: 1)
104.    I/Activity(  237): onCreate (TID: 1)
105.    I/ActivityThread(  237): performLaunchActivity -> Activity.performStart
106.    I/Activity(  237): performStart
107.    I/Activity(  237): onStart
108.    I/ActivityThread(  237): performLaunchActivity -> Instrumentation.callActivityOnPostCreate
109.    I/Instrumentation(  237): callActivityOnPostCreate (TID: 1)
110.    I/Activity(  237): onPostCreate
111.    I/ActivityThread(  237): performResumeActivity (TID: 1)
112.    I/Activity(  237): performRestart
113.    I/Activity(  237): onResume
114.    I/Activity(  237): onPostResume
115.    I/ActivityThread(  237): handleResumeActivity: adding the DecorView to the Activity's WM
116.    I/Window.LocalWindowManager(  237): addView
117.    I/WindowManagerImpl(  237): addView view=com.android.internal.policy.impl.PhoneWindow$DecorView@43b829e0
118.    I/ViewRoot(  237): ViewRoot ctor
119.    I/ViewRoot(  237): getWindowSession
120.    I/WindowManagerService(   58): openSession
```

Android startActivity Log IV

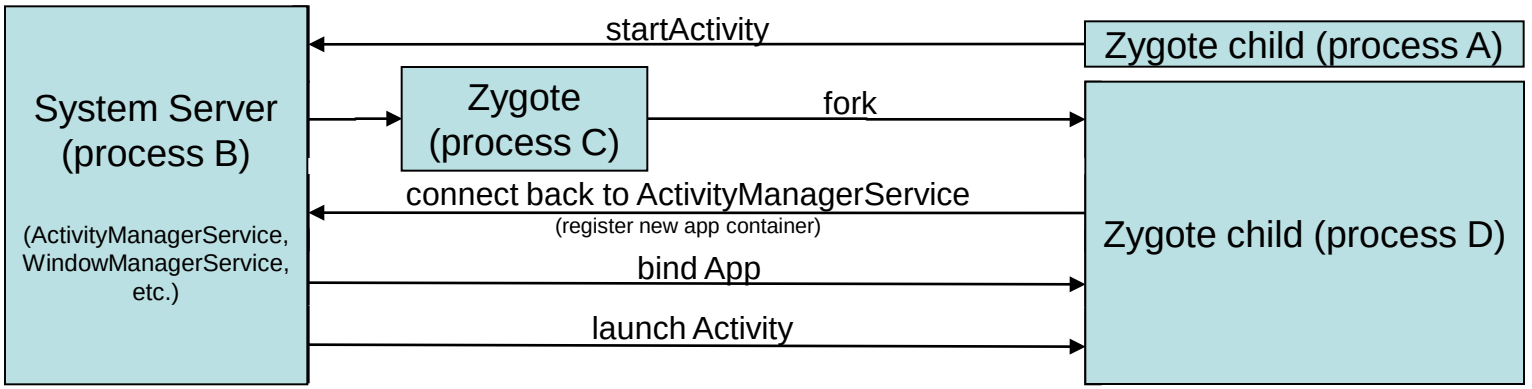
```
121.    I/ViewRoot(  237): getWindowSession (mInitialized = true)
122.    I/WindowManagerImpl(  237): mViews == null
123.    I/ViewRoot(  237): setView - sWindowSession.add
124.    I/WindowManagerService.Session(   58): add
125.    I/WindowManagerService(   58): addWindow - adjustWindowParams
126.    I/WindowManagerImpl(  237): root.setView
127.    I/Activity(  237): makeVisible
128.    I/WindowManagerService.Session(   58): relayout
129.    I/ViewRoot(  237): dispatchResized
130.    D/dalvikvm(   58): GREF has increased to 301
131.    I/WindowManagerService.Session(   58): relayout
132.    I/ActivityThread(  237): handleMessage: ACTIVITY_CONFIGURATION_CHANGED
133.    I/ARMAssembler(   58): generated scanline__00000077:03545404_00000004_00000000 [ 47 ipp] (67 ins) at
        [0x35b9c8:0x35bad4] in 1581835 ns
134.    I/ActivityManager(   58): Displayed activity com.android.alarmclock/.AlarmClock: 2972 ms (total 2972 ms)
135.    I/ActivityThread(  112): handleMessage: STOP_ACTIVITY_HIDE
```

ActivityManager

- What is it?

  - Manages the Activity lifecycles etc.
  - Service side is implemented by ActivityManagerService.java
  - Client side is implemented by ActivityManager.java and Activity.java

- How are Activities started?



Zygote I

- What is it?

  Zygote is a process including the Dalvik VM that preloads the complete Android framework classes. Zygote is the template process for each app and service.
- What does it do?

  It listens on a local socket for requests to fork child processes. After forking each child process initializes the Binder runtime and calls ActivityThread.main. Remember that fork is copy-on-write in Linux.

Zygote II

Call Stack:

Processes A, B, C and D. M is the main thread, B is a Binder thread

C, M: ZygoteInit.main (ZygoteInit.java)
C, M: ZygoteInit.runSelectLoopMode (ZygoteInit.java)
C, M: ZygoteConnection.runOnce (ZygoteConnection.java)
C, M: Zygote.forkAndSpecialize (Zygote.java)
C, M: Dalvik_dalvik_system_Zygote_forkAndSpecialize (dalvik_system_Zygote.c)
C, M: forkAndSpecializeCommon (dalvik_system_Zygote.c) → this method does the fork syscall
D, M: … initialize Dalvik VM
D, M: ZygoteConnection.handleChildProc (ZygoteConnection.java) (Zygote [process C] calls handleParentProc)
D, M: RuntimeInit.zygoteInit (RuntimeInit.java)
D, M: RuntimeInit.zygoteInitNative (RuntimeInit.java)
D, M: maps to com_android_internal_os_RuntimeInit_zygoteInit (AndroidRuntime.cpp)
D, M: AppRuntime.onZygoteInit (app_main.cpp) → this method initializes the binder stuff
D, M: RuntimeInit.invokeStaticMain (RuntimeInit.java) → resolves to android.app.ActivityThread.main
D, M: → invokeStaticMain packs the call to ActivityThread.main into a ZygoteInit.MethodAndArgsCaller object
D, M: → this object is then thrown as exception to clean up the stack frame used to set up the child process
D, M: ZygoteInit.main catches the ZygoteInit.MethodAndArgsCaller exception thrown by RuntimeInit.invokeStaticMain
→ since the ZygoteInit.MethodAndArgsCaller object implements Runnable its run method is called
→ this calls ActivityThread.main

Binder I

- What is it?

  Binder is the IPC framework used by Android.

  The Binder IPC is blocking.

- How does it work?

  Each child of Zygote contains the Binder runtime. This runtime contains a small thread pool (~3 threads) that block on Binder‘s kernel driver to handle requests.
  → Binder threads only handle Binder requests.
  → Binder requests are directly sent by the calling thread using Binder‘s (blocking) `transact` method to perform a remote method call. (Proxy - BinderProxy.transact, Service - Binder.transact)

  → To switch thread contexts use Handlers and Loopers.

Binder II

Each Binder runtime is represented by an instance of class ProcessState.

This instance is created by the first call to ProcessState.self

Binder Setup Call Stack:

Processes A, B, C and D. M is the main thread, B is a Binder thread

Zygote‘s child init calls:

D, M: AppRuntime.onZygoteInit (app_main.cpp)

D, M: ProcessState.self (ProcessState.cpp)

→ creates a ProcessState instance for this Zygote child process and connects to Binder‘s kernel driver (see ctor)

D, M: ProcessState.startThreadPool (ProcessState.cpp)

D, M: ProcessState.spawnPooledThread (ProcessState.cpp) → creates a new PoolThread

D, M: PoolThread.run (ProcessState.cpp)

→ now the Zygote init stuff goes on inside of this thread context…

→ we now switch the thread context to the newly created Binder thread for this call stack

D, B: PoolThread.threadLoop (ProcessState.cpp)

D, B: IPCThreadState.self (IPCThreadState.cpp) → creates an instance of IPCThreadState for the current thread

→ each thread that uses Binder (caller or callee) creates his own IPCThreadState object

D, B: IPCThreadState.joinThreadPool (IPCThreadState.cpp) → joins the Binder IPC thread pool and starts looping

D, B: IPCThreadState.talkWithDriver → wait for some Binder request while blocking on the Binder kernel driver

D, B: IPCThreadState.executeCommand

…calls Binder‘s onTransact method

(→ IPCThreadState.executeCommand → BBinder.transact → JavaBBinder.onTransact → Binder.execTransact → Binder.onTransact)

Handler and Looper

- What is it?

  A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue. Handlers bind themselves to a thread context. If a Handler is created with its empty ctor it is bound to the calling thread.

  A Looper used to run a message loop for a thread. Threads by default do not have a message loop associated with them; to create one, call prepare() in the thread that is to run the loop, and then loop() to have it process messages until the loop is stopped.

startActivity I

Call Stack:

Processes A, B, C and D. M is the main thread, B is a Binder thread

A, M: Activity.startActivity (Activity.java)
A, M: Instrumentation.execStartActivity (Activity.java)
A, M: ActivityManagerProxy.startActivity (ActivityManagerNative.java)
B, B: ActivityManagerNative.onTransact(START_ACTIVITY_TRANSACTION) (ActivityManagerNative.java)
B, B: ActivityManagerService.startActivity (ActivityManagerService.java)
B, B: ActivityManagerService.startActivityLocked (ActivityManagerService.java) → creates the Activity's HistoryRecord
B, B: ActivityManagerService.startActivityUnchechedLocked (ActivityManagerService.java)
B, B: ActivityManagerService.resumeTopActivityLocked (ActivityManagerService.java)
B, B: ActivityManagerService.startPausingLocked (ActivityManagerService.java)
B, B: ApplicationThreadProxy.schedulePauseActivity (ApplicationThreadNative.java)
A, B: ApplicationThreadNative.onTransact(SCHEDULE_PAUSE_ACTIVITY_TRANSACTION) (ApplicationThreadNative.java)
A, B: ActivityThread.ApplicationThread.schedulePauseActivity (ActivityThread.java)
A, M: ActivityThread.handleMessage(PAUSE_ACTIVITY) (ActivityThread.java)
A, M: ActivityThread.handlePauseActivity (ActivityThread.java)
A, M: ActivityManagerProxy.activityPaused (ActivityManagerNative.java)
B, B: ActivityManagerNative.onTransact(ACTIVITY_PAUSED_TRANSACTION) (ActivityManagerNative.java)
B, B: ActivityManagerService.activityPaused (ActivityManagerService.java)
B, B: ActivityManagerService.completePauseLocked (ActivityManagerService.java)
B, B: ActivityManagerService.resumeTopActivityLocked (ActivityManagerService.java)
B, B: ActivityManagerService.startSpecificActivityLocked (ActivityManagerService.java)
B, B: ActivityManagerService.startProcessLocked (ActivityManagerService.java) → creates the ProcessRecord
B, B: Process.start (ActivityManagerService.java)
B, B: Process.startViaZygote (Process.java)
B, B: Process.zygoteSendArgsAndGetPid (Process.java)
B, B: Process.openZygoteSocketIfNeeded (Process.java)
→ send a request using Zygote's local socket to fork a new child process

… at this point the call stacks of Zygote and Binder are executed.

startActivity II

Call Stack continued:

Processes A, B, C and D. M is the main thread, B is a Binder thread

… we go on at ActivityThread.main which is executed in the forked Zygote child process.

D, M: ActivityThread.main (ActivityThread.java)
- → creates the ActivityThread instance for this process
- → each ActivityThread contains the ApplicationThread instance for the process which manages activity and service lifecycles etc.
- → the ApplicationThread is responsible for managing the software component container (process) on behalf of the ActivityManagerService

D, M: ActivityThread.attach (ActivityThread.java)
D, M: RuntimeInit.setApplicationObject (RuntimeInit.java)
D, M: ActivityManagerNative.getDefault (ActivityManagerNative.java) → connects back to ActivityManagerService
D, M: ActivityManagerProxy.attachApplication (ActivityManagerNative.java)
→ registers the ActivityThread‘s ApplicationThread service object at the ActivityManagerService
B, B: ActivityManagerNative.onTransact(ATTACH_APPLICATION_TRANSACTION) (ActivityManagerNative.java)
→ gets an endpoint reference to the ActivityThreads‘s ApplicationThread service object (ApplicationThreadProxy)
B, B: ActivityManagerService.attachApplication (ActivityManagerService.java)
B, B: ActivityManagerService.attachApplicationLocked (ActivityManagerService.java)
B, B: ApplicationThreadProxy.bindApplication (ApplicationThreadNative.java)
B, B: ActivityManagerService.realStartActivityLocked (ActivityManagerService.java) → uses ProcessRecord and HistoryRecord
B, B: ApplicationThreadProxy.scheduleLaunchActivity (ApplicationThreadNative.java)

→ ApplicationThreadProxy.bindApplication and ApplicationThreadProxy.scheduleLaunchActivity are called next to each other. But B1 and B2 do NOT execute in parallel inside of process C. (Does Binder‘s kernel driver assure this behaviour?)
So B2 does not start before B1 is done with ActivityThread.ApplicationThread.bindApplication.

D, B1: ApplicationThreadNative.onTransact(BIND_APPLICATION_TRANSACTION) (ApplicationThreadNative.java)
D, B1: ActivityThread.ApplicationThread.bindApplication (ActivityThread.java)
D, M: ActivityThread.handleMessage(BIND_APPLICATION) (ActivityThread.java)
D, M: ActivityThread.handleBindApplication (ActivityThread.java)
D, M: ActivityThread.PackageInfo.makeApplication (ActivityThread.java)
→ creates the initial app‘s ApplicationContext
→ this ApplicationContext is returned by Activity.getApplicationContext
D, M: Instrumentation.newApplication(android.app.Application)

startActivity III

Call Stack continued:

Processes A, B, C and D. M is the main thread, B is a Binder thread

D, B2: ApplicationThreadNative.onTransact(SCHEDULE_LAUNCH_ACTIVITY_TRANSACTION) (ApplicationThreadNative.java)
D, B2: ActivityThread.ApplicationThread.scheduleLaunchActivity (ActivityThread.java)
D, M: ActivityThread.handleMessage(LAUNCH_ACTIVITY) (ActivityThread.java)
D, M: ActivityThread.handleLaunchActivity (ActivityThread.java)
D, M: ActivityThread.performLaunchActivity (ActivityThread.java)
D, M: Instrumentation.newActivity (Instrumentation.java) → creates the Activity object (e.g. vu.de.urpool.quickdroid.Quickdroid)
D, M: ActivityThread.PackageInfo.makeApplication (ActivityThread.java) → returns existing app object
D, M: create new ApplicationContext for the newly created Activity
D, M: Activity.attach (Activity.java) → gets the newly created ApplicationContext as argument
→ creates the Activity's window and attaches it to its local WindowManager
D, M: Instrumentation.callActivityOnCreate (Instrumentation.java)
D, M: Activity.onCreate (Activity.java) → Activity.setContentView
D, M: Activity.performStart (Activity.java)
D, M: Activity.onStart (Activity.java)
D, M: Instrumentation.callActivityOnRestoreInstanceState (Instrumentation.java)
D, M: Activity.onRestoreInstanceState (Activity.java)
D, M: Instrumentation.callActivityOnPostCreate (Instrumentation.java)
D, M: Activity.onPostCreate (Activity.java)
D, M: ActvityThread.handleResumeActivity (ActivityThread.java)
D, M: ActivityThread.performResumeActivity (ActivityThread.java)
D, M: Activity.performResume (Activity.java)
D, M: Instrumentation.callActivityOnResume (Instrumentation.java)
D, M: Activity.onResume (Activity.java)
D, M: Activity.onPostResume (Activity.java)
D, M: Window.LocalWindowManager.addView (Window.java)
→ attaches the Activity's view hierarchy to the WindowManagerService via the local WindowManager
D, M: Activity.makeVisible (Activity.java)